EINDHOVEN UNIVERSITY OF TECHNOLOGY

On local search and LP and SDP relaxations for k-set packing

Citation for published version (APA):
Oosterwijk, T. S. (2015). *On local search and LP and SDP relaxations for k-set packing*. (arXiv; Vol. 1507.07459 [math.CO]). s.n.

Document status and date:
Published: 01/01/2015

Document Version:
Publisher's PDF, also known as Version of Record (includes final page, issue and volume numbers)

Please check the document version of this publication:

• A submitted manuscript is the version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher's website.
• The final author version and the galley proof are versions of the publication after peer review.
• The final published version features the final layout of the paper including the volume, issue and page numbers.

Link to publication

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

• Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
• You may not further distribute the material or use it for any profit-making activity or commercial gain
• You may freely distribute the URL identifying the publication in the public portal.

If the publication is distributed under the terms of Article 25fa of the Dutch Copyright Act, indicated by the "Taverne" license above, please follow below link for the End User Agreement:

www.tue.nl/taverne

Take down policy
If you believe that this document breaches copyright please contact us at:

openaccess@tue.nl

providing details and we will investigate your claim.

Download date: 24. Sep. 2021

arXiv:1507.07459v1 [math.CO] 27 Jul 2015

Eindhoven University of Technology

On local search and LP and SDP relaxations for k-set packing

Tim Oosterwijk

A thesis submitted in partial fulfillment
of the requirements of the degree of

Master of Science

October 8, 2013

Important note

There is a mistake in the following line of Theorem 17: "As an induced subgraph of H with more edges than vertices constitutes an improving set". Therefore, the proofs of Theorem 17, and hence Theorems 19, 23 and 24, are false. It is still open whether these theorems are true.

Abstract

Set packing is a fundamental problem that generalises some well-known combinatorial optimization problems and knows a lot of applications. It is equivalent to hypergraph matching and it is strongly related to the maximum independent set problem.

In this thesis we study the k-set packing problem where given a universe $\mathcal{U}$ and a collection $\mathcal{C}$ of subsets over $\mathcal{U}$, each of cardinality k, one needs to find the maximum collection of mutually disjoint subsets. Local search techniques have proved to be successful in the search for approximation algorithms, both for the unweighted and the weighted version of the problem where every subset in $\mathcal{C}$ is associated with a weight and the objective is to maximise the sum of the weights. We make a survey of these approaches and give some background and intuition behind them. In particular, we simplify the algebraic proof of the main lemma for the currently best weighted approximation algorithm of Berman ([Ber00]) into a proof that reveals more intuition on what is really happening behind the math.

The main result is a new bound of $\frac{k}{3} + 1 + \varepsilon$ on the integrality gap for a polynomially sized LP relaxation for k-set packing by Chan and Lau ([CL10]) and the natural SDP relaxation [**NOTE: see page iii**]. We provide detailed proofs of lemmas needed to prove this new bound and treat some background on related topics like semidefinite programming and the Lovász Theta function.

Finally we have an extended discussion in which we suggest some possibilities for future research. We discuss how the current results from the weighted approximation algorithms and the LP and SDP relaxations might be improved, the strong relation between set packing and the independent set problem and the difference between the weighted and the unweighted version of the problem.

Acknowledgements

First of all, I would like to give a huge thanks to my supervisor Nikhil Bansal. I learned so much during (the research for) this thesis about subjects that interest me, more than this thesis could ever contain. You have only encouraged me to absorb all that knowledge and broaden my horizons. You showed me fascinating results on all related topics and problems and inspired me by researching all that might be interesting for my academic future. Thank you for everything you have done to make me a better researcher.

I would like to thank my future (co-)promotor Tjark Vredeveld; your interest in the thesis and me has contributed to my confidence as a researcher and my general feeling of well-being. Thank you, I'm looking forward to our cooperation in the years to come.

I also owe a thank you to my fellow student Annette Ficker, with whom I initially researched various interesting problems until we parted ways when I chose k-set packing as my subject and you chose 2-dimensional bin packing for your thesis. Thank you for the fruitful discussions about everything we encountered and of course for the very nice time we had together during university.

I also would like to thank some other people who discussed the problem with me, among which are authors of some papers I read. In particular I would like to thank Per Austrin, Fabrizio Grandoni, Konstantin Makarychev, Monaldo Mastrolilli, Viswanath Nagarajan and Ruben van der Zwaan for their time.

A big thank you to the teachers in the combinatorial optimization group for their interesting and inspiring classes: Nikhil Bansal, Cor Hurkens, Judith Keijsper, Rudi Pendavingh and Gerhard Woeginger. A special thanks to Jan Draisma and Gerhard Woeginger for forming the assessment committee for my thesis.

Also I want to thank some other fellow students, in particular Jorn van der Pol and Reint den Toonder, for their interest in the thesis, their general support and of course their friendly company for the past years.

Finally a thanks to my family and boyfriend for their support and faith. I would not be where I am today if it weren't for you. A huge thanks to all of you.

Contents

Abstract **v**

Acknowledgements **vii**

Table of contents **ix**

1 Introduction **1**

1.1 Definition of the problem 1

1.2 Terminology 2

1.3 Contribution 3

1.4 Outline of the thesis 4

2 Applications and related problems **7**

2.1 Applications 7

2.2 Related problems 8

2.3 Local search 11

3 Current results **13**

3.1 Unweighted approximations 13

3.2 Weighted approximations 17

3.3 Parameterized complexity 19

3.4 Hardness results 20

4 LP formulation **25**

4.1 The standard LP 25

4.2 Strengthening the LP formulation 26

4.3 A polynomially sized LP 31

5 SDP formulation **33**
5.1 Background on semidefinite programming 33
5.2 Background on the Lovász Theta function 34
5.3 An SDP for k-set packing 36

6 Weighted approximation **39**
6.1 Terminology 39
6.2 Two algorithms joining forces 41
6.3 Simplified proof 44

7 Discussion **47**
7.1 LP and SDP relaxations 47
7.2 Improving Berman's weighted approximation 49
7.3 Relation to the independent set problem 51
7.4 Unweighted versus weighted k-set packing 53

A Parameterized complexity **55**

Bibliography **57**

Chapter 1

Introduction

1.1 Definition of the problem

Set packing and k-set packing Set packing is one of Karp's 21 NP-complete problems [Kar72] and it has received a lot of attention during the past years. A lot of progress has been made on the complexity of this problem, even though under standard complexity assumptions algorithms for this problem require at least superpolynomial or perhaps even exponential running time. Set packing is a fundamental problem that generalises some well-known problems and thus knows a lot of applications. There has been a long line of research on this problem. In this thesis we will consider k-set packing, which is the following problem.

> **k-Set Packing (k-SP)**
> Given: a universe $\mathcal{U}$ of N elements and a collection $\mathcal{C} \subseteq 2^{\mathcal{U}}$ of n subsets over $\mathcal{U}$, each of cardinality k
> Find: a maximum collection of mutually disjoint subsets in $\mathcal{C}$.

Any collection of mutually disjoint subsets in $\mathcal{C}$ is called a packing and the goal is to find the largest packing. In k-set packing every subset in $\mathcal{C}$ contains at most k elements. Without loss of generality assume every subset contains exactly k elements: add some unique dummy elements to the subsets of less than k elements.

k-set packing is a special case of the optimization version of the set packing problem, where there is no restriction on the cardinality of every subset in $\mathcal{C}$.

> **(Maximum) Set Packing (SP)**
> Given: a universe $\mathcal{U}$ of N elements and a collection $\mathcal{C} \subseteq 2^{\mathcal{U}}$ of n subsets over $\mathcal{U}$
> Find: a maximum collection of mutually disjoint subsets in $\mathcal{C}$.

1.2 Terminology

We proceed with some notational conventions and definitions to make sure no confusion may arise. By $\mathcal{U}$ we mean the universe of elements over which the k-set system $\mathcal{C} \subseteq 2^{\mathcal{U}}$ has been defined. We write $|\mathcal{U}| = N$ and $|\mathcal{C}| = n$. By $I = (\mathcal{U}, \mathcal{C})$ a given instance for k-set packing is denoted in which the objective is to find the largest packing.

Definition 1. ***(Conflict graph)*** *Let an instance $I = (\mathcal{U}, \mathcal{C})$ for the k-set packing problem be given. The conflict graph of I is the graph G where every subset in $\mathcal{C}$ is represented by a vertex. Two vertices are adjacent if and only if the subsets in $\mathcal{C}$ these vertices correspond to intersect each other.*

Now let $\mathcal{A}$ be some solution to this instance, i.e. a collection of subsets in $\mathcal{C}$ that are mutually disjoint.

Definition 2. ***(Intersection graph)*** *Let an instance I for the k-set packing problem be given and let $\mathcal{A}$ and $\mathcal{B}$ be two packings. The intersection graph of $\mathcal{A}$ and $\mathcal{B}$ is the induced subgraph of their vertices in the conflict graph of I.*

The bipartite intersection graph of $\mathcal{A}$ and $\mathcal{B}$ thus contains a vertex for every set in $\mathcal{A}$ and $\mathcal{B}$ and two vertices are adjacent if and only if the subsets in the set packing instance are in conflict. Throughout this thesis calligraphic letters $\mathcal{A}$ will be used to denote collections of subsets and normal letters A to denote the set of vertices corresponding to $\mathcal{A}$ in the conflict or intersection graph. Denote $N_G(B, A) = N(B) \cap A$, the neighbours of B in A in their intersection graph G. Sometimes for brevity we just write $N(G, A)$ if the graph G is clear from the context. We can now succinctly define an improving set.

Definition 3. ***(Improving set)*** *Let $\mathcal{A}$ and $\mathcal{B}$ be two packings. $\mathcal{B}$ is called an improving set for $\mathcal{A}$ when $|A \cup B \setminus N_G(B, A)| > |A|$, i.e. when adding the sets of $\mathcal{B}$ to the current solution $\mathcal{A}$ and removing the sets of $\mathcal{A}$ that they intersect, leads to a solution of larger cardinality.*

The removal of the neighbourhood of B in A ensures that the new solution is also mutually disjoint. Now define the following.

Definition 4. ***(t-locally optimal solution)*** *Let $\mathcal{A}$ be a packing. $\mathcal{A}$ is said to be t-locally optimal when for every collection of mutually disjoint subsets $\mathcal{B}$ with $|\mathcal{B}| \leq t$ we have $|A \cup B \setminus N_G(B, A)| \leq |A|$. In other words, a solution is a t-locally optimal solution when there does not exist an improving set of size at most t.*

Finally, an algorithm is said to approximate a maximization problem within a factor $\rho > 1$ if for every instance of the problem $\frac{v(OPT)}{v(A)} \leq \rho$, where $v(A)$ is the objective value of the output of the algorithm and $v(OPT)$ is the best achievable objective value for that instance. We call ρ its approximation guarantee.

1.3 Contribution

1.3.1 Improved integrality gap

In this thesis we improve the integrality gap of a linear program of k-set packing called the intersecting family LP (see Section 4.2).

Theorem 5. *Let $\varepsilon > 0$. The integrality gap of the intersecting family LP is at most $\frac{k}{3} + 1 + \varepsilon$.*

Following the results from [CL10], this immediately implies the following two theorems.

Theorem 6. *Let $\varepsilon > 0$. There is a polynomially sized LP for k-set packing with integrality gap at most $\frac{k}{3} + 1 + \varepsilon$.*

Theorem 7. *Let $\varepsilon > 0$. There is a polynomially sized SDP for k-set packing with integrality gap at most $\frac{k}{3} + 1 + \varepsilon$.*

The previous bound on this integrality gap was $\frac{k+1}{2}$ [CL10]. This was a continuation of the work on the standard linear programming relaxation for k-set packing. We treat these results on the linear programs in Chapter 4. The result on the semidefinite programming relaxation is treated in Chapter 5, along with some background.

1.3.2 Simplified proof

We simplify the proof of the main lemma of the currently best weighted approximation algorithm from Berman [Ber00, Lemma 2]. The current proof is very clever but also very algebraic. We make the observation that the squared weight function somehow captures both the maximum weight and the sum of the weights of the neighbourhood of a vertex in the conflict graph of the instance. This allows us to avoid the algebraic proof and simplify it.

We treat this result in Chapter 6.

1.4 Outline of the thesis

Chapter 2 Set packing is one of the fundamental optimization problems and therefore there are numerous applications within mathematics and real life. It is highly related to some other well-known problems such as the hypergraph matching problem and the maximum independent set problem. In Chapter 2 these applications and related problems are considered.

Chapter 3 In Chapter 3 the current results on the k-set packing problem are discussed. First the unweighted approximation algorithms and the weighted approximation algorithms are considered in Sections 3.1 and 3.2. For the unweighted problem the best approximation algorithm currently achieves an approximation guarantee of $\frac{k+1}{3}+\varepsilon$ [Cyg13, FY13] and for the weighted problem the best result is a $\frac{k+1}{2}$-approximation [Ber00].

Then Section 3.3 continues with the parameterized algorithms for k-set packing, as this problem is fixed parameter tractable. There has been a long line of research in this area and Appendix A contains an overview of these algorithms.

Chapter 3 ends with some results on the inherent hardness of the problem in Section 3.4. Subsection 3.4.1 starts with some hardness results that apply to the general set packing problem. Subsections 3.4.2, 3.4.3 and 3.4.4 contain theorems on k-set packing specifically. It is NP-hard to approximate within a factor of $\Omega\left(\frac{k}{\log k}\right)$ [HSS06] and three other results on the limits of local search techniques for the problem are mentioned [Cyg13, FY13, SW13].

Chapter 4 Chapter 4 treats the results on the standard linear programming relaxation and the intersecting family LP and contains the proofs of Theorems 5 and 6.

Chapter 5 The proof of Theorem 7 is given in Chapter 5 along with some background about semidefinite programming and the Lovász Theta function.

Chapter 6 The topic of Chapter 6 is weighted k-set packing. Currently the best result is a $\frac{k+1}{2}$-approximation from Berman [Ber00]. He provides two algorithms. One is called SquareImp which uses a local search technique using the squared weight function. The second is called WishfulThinking which searches locally for structures which he calls nice claws (definitions are in Chapter 6). He links these two algorithms in a shrewd way, allowing him to proof both the approximation guarantee and the polynomial running time. At the end we give a simplified proof of the main lemma.

Chapter 7 Finally there is an extended discussion in Chapter 7 about possible improvements. The results on the LP and the SDP relaxation are discussed in Section 7.1 and whether these can be extended to the weighted case. We see why changing Berman's [Ber00] weighted $\frac{k+1}{2}$-approximation algorithm a bit does not yield an improvement in Section 7.2. In Section 7.3 the strong relation between set packing and the independent set problem (on bounded degree graphs) is considered and it is argued why the results for independent set are better. We discuss why the weighted problem asks for such different algorithms compared to the unweighted version of the problem in Section 7.4. Several suggestions for future research are given.

Chapter 2

Applications and related problems

2.1 Applications

Set packing has a lot of applications in capital budgeting, crew scheduling, cutting stock, facilities location, graphs and networks, manufacturing, personnel scheduling, districting, information systems, vehicle routing and timetable scheduling, see [Vem98] for a survey. In this section some real life applications are mentioned and in Section 2.2 the relation of set packing to other combinatorial problems is discussed. Section 2.3 then gives some background on local search as a background for Chapter 3.

Latin squares A nice application of set packing is the extension of partial Latin squares [GRS04, HJKS07]. A partial Latin square is an $n \times n$ array in which each cell is either empty or coloured with exactly one of the colours $\{1, \ldots, n\}$. A Latin square is a partial Latin square without empty cells where every colour occurs exactly once in every row and every column. The problem is given a partial Latin square to find a completion that colours as many empty cells as possible such that no rows or columns contain any colour more than once.

This can be modeled as a set packing problem as follows. Let the universe $\mathcal{U}$ consist of $3n^2$ elements of the form $\{e_i, r_j\}$, $\{e_i, c_k\}$ and $\{r_j, c_k\}$ for every combination of some colour i, some row j and/or some column k. Call an element $\{e_i, r_j\}$ or $\{e_i, c_k\}$ empty if respectively row j or column k does not contain colour i, and call $\{r_j, c_k\}$ empty if the cell in row j column k is empty. Now let the collection of subsets $\mathcal{C}$ consist of all triplets $\{\{e_i, r_j\}, \{e_i, c_k\}, \{r_j, c_k\}\}$ where all three elements are empty. This creates a set packing instance where every triplet contained in the solution indicates to colour the cell at row j column k with colour i.

This idea can be extended to the popular Sudoku puzzles. A Sudoku is a 9×9 Latin square with the additional constraints that in the nine 3×3 "boxes" every colour is allowed only once. Using quadruples rather than triplets, this problem can be translated to set packing in a similar fashion.

Other applications Three real life applications one could think of is the assignment of crew members to airplanes [Ski08], the generation of a coalition structure in multiagent systems [SLA+99] and determining the winners in combinatorial auctions to maximise the profit [GL00, San02, Vic61].

2.2 Related problems

Set packing is highly related to the hypergraph matching problem and the independent set problem. This section surveys these relations and some special cases of set packing.

2.2.1 Hypergraph matching

Set packing and hypergraph matching are really the same problem with different names. This subsection contains some background on hypergraphs and hypergraph matchings and relates these notions to set packing and k-set packing.

Hypergraphs A hypergraph H is a pair $H = (V, E)$ where V is the set of vertices and E is the set of hyperedges. A hyperedge $e \in E$ is a nonempty subset of the vertices. In a weighted hypergraph, every hyperedge $e \in E$ is associated with a weight $w(e)$. A hyperedge $e \in E$ is said to contain, cover or be incident to a vertex $v \in V$ when $v \in e$.

For a vertex v, $\delta(v)$ denotes the set of hyperedges incident to it. The degree of a vertex v is $|\delta(v)|$. The cardinality of a hyperedge is the number of vertices it contains. When every vertex has the same degree k, the hypergraph is called k-regular. When every hyperedge has the same cardinality k, the hypergraph is k-uniform. A graph is thus a 2-uniform hypergraph. In a graph, every edge is a subset of two vertices.

Matchings A hypergraph matching is a subset of the hyperedges $M \subseteq E$ such that every vertex is covered at most once, i.e. the hyperedges are mutually disjoint. This generalises matchings in graphs. From now on, with matching we mean a hypergraph matching. The cardinality of a matching is the number of hyperedges it contains. A matching is called maximum if it has the largest cardinality of all possible matchings. In the hypergraph matching problem, a hypergraph is given and one needs to find the maximum matching.

Hypergraph matching problem
Given: a hypergraph $H = (V, E)$
Find: a maximum collection of mutually disjoint hyperedges.

k-partite hypergraphs The notion of bipartiteness in graphs can be generalised in a hypergraph to the concept of k-partiteness. A hypergraph $H = (V, E)$ is called k-partite if the set of vertices V can be partitioned into k classes $V_1, \ldots, V_k$ such that every hyperedge $e \in E$ touches exactly one vertex from every vertex class. A k-partite hypergraph is k-uniform by definition. The hypergraph matching problem on a k-partite graph is also called the k-dimensional matching problem. This generalises the bipartite matching on graphs where $k = 2$ and it is a restricted version of hypergraph matching on a k-uniform hypergraph. In particular the 3-dimensional matching problem is well studied in literature.

Relation to set packing Let a set packing instance $S = (\mathcal{U}, \mathcal{C})$ be given. Define a hypergraph $H = (\mathcal{U}, \mathcal{C})$, calling the elements in $\mathcal{U}$ vertices and the subsets in $\mathcal{C}$ hyperedges. Then the set packing problem on S is exactly the same as the hypergraph matching problem on H. An instance of the k-set packing problem translates to an instance of the hypergraph matching problem on a k-uniform hypergraph, as every subset (hyperedge) contains exactly k elements.

Results for set packing thus immediately apply to hypergraph matching and vice versa.

2.2.2 Independent set

Relation to set packing Set packing is also closely related to the (maximum) independent set problem. Given a graph $G = (V, E)$, an independent set is a subset of vertices that are mutually non-adjacent, i.e. a set of vertices whose induced subgraph does not contain any edge. This subsection treats this problem and its relation to set packing.

Independent set problem
Given: a graph $G = (V, E)$
Find: a maximum collection of mutually non-adjacent vertices.

Let a set packing instance $I = (\mathcal{U}, \mathcal{C})$ be given. Create the conflict graph $G = (\mathcal{C}, E)$ for the instance I. Then the edge set E captures all the intersections between the subsets. Any packing of $\mathcal{C}$ now corresponds to an independent set in G: as a packing is mutually disjoint by definition, the corresponding vertices are non-adjacent. On the other hand, any independent set in G corresponds to a packing of $\mathcal{C}$. Finding a maximum packing is thus equivalent to finding a maximum independent set in the conflict graph.

Relation to k-set packing Now consider a k-set packing instance $(\mathcal{U}, \mathcal{C})$ where every subset in $\mathcal{C}$ contains exactly k elements from $\mathcal{U}$. Consider a fixed subset S and look at the set $N(S)$ of all subsets intersecting S. There can be at most k subsets in $N(S)$ that are mutually disjoint, when each subset intersects S in a distinct element. Thus within $N(S)$, the maximum packing has at most cardinality k.

Now consider the corresponding conflict graph G. S is a vertex in G and $N(S)$ is the set of all neighbours of S. By the previous reasoning, $N(S)$ contains an independent set of size at most k. Thus in the neighbourhood of any vertex of G there are at most k mutually non-adjacent vertices. In other words, the conflict graph is $k+1$-claw free: it does not contain $K_{1,k+1}$ as an induced subgraph, where $K_{n,m}$ is the complete bipartite graph on n and m vertices. A vertex can have any number of neighbours, but the maximum number of mutually non-adjacent neighbours is k.

Independent set on bounded degree graphs On the other hand, the maximum independent set problem in graphs G where all degrees are bounded by k is a special case of k-set packing. Map every vertex in G to a subset in $\mathcal{C}$ and map every edge to a distinct element in $\mathcal{U}$. Let a subset in $\mathcal{C}$ (a vertex in G) contain all elements in $\mathcal{U}$ (edges in G) that are incident to it. Now every subset in $\mathcal{C}$ contains at most k elements because every vertex in G had degree at most k. Again, by adding dummy elements to the subsets with less than k elements, every subset in $\mathcal{C}$ has exactly k elements. As such, the k-set packing problem generalises the maximum independent set problem on bounded degree graphs.

Section 7.3 discusses the relation between the results on set packing and independent set more thoroughly.

2.2.3 Special cases of set packing

Clique and triangle packing There are some special cases of set packing we would like to mention. Instead of studying the set packing problem with a bound on the cardinality of every subset, it is also possible to study the problem with certain given structures on the given sets. One example is the clique packing problem, where one is given some graph G. A packing of G is a collection of pairwise vertex-disjoint subgraphs of G, each of which is isomorphic to a clique. A packing is said to cover an edge of G if one of the subgraphs contains that edge. In the clique packing problem, one tries to maximise the number of covered edges [CMWY09]. The special case where every clique is of size 3 is called the triangle packing problem [CTW09, HR06]. The set packing problem generalises these problems.

Tree-like weighted set packing The tree-like weighted set packing problem is another subproblem with an additional structure on the sets that allows for better results that even extend to the weighted version. In this problem, the subsets in $\mathcal{C}$ can be organised into a forest F satisfying the following properties. Every vertex in F corresponds to one subset in $\mathcal{C}$. If a vertex Y is a child of a vertex X, then Y is a subset of X. If Y and Z are distinct children from X, then Y and Z are disjoint. And the roots of all the trees are pairwise disjoint. When the sets are structured like this, the problem can be solved exactly in cubic time using a dynamic programming algorithm [GT10].

k-dimensional matching Perhaps the best known special case of $k$set packing is the k-dimensional matching problem. As was mentioned in Subsection 2.2.1, this is the hypergraph matching problem on a k-partite hypergraph, which is a stronger structure than a k-uniform hypergraph. In particular 3-dimensional matching is well-known. All results for k-set packing immediately apply to k-dimensional matching.

2.3 Local search

Background A lot of the approximation algorithms for k-set packing are related to the notion of local search techniques. In this section we survey this related topic. Local search is a natural heuristic to tackle difficult problems. The most basic solution approaches for a discrete optimization problem are generating just one solution (e.g. using a priority-based heuristic), enumerating implicitly all possible solutions (e.g. with branch and bound techniques) and generating several solutions and choosing the best. Local search is a heuristic that finds from an initial solution a sequence of solutions that use the previous solution for the next solution, with the goal to increase the objective value from the initial solution to a better value. A neighbourhood structure is defined on the solutions that somehow resembles similar solutions. Until some stopping criterion is reached, in every iteration a solution from the neighbourhood of the current solution is chosen.

A local search heuristic usually consists of four main elements: a method to calculate an initial solution, a definition of the neighbourhood of a solution, a criterion to choose a solution from any neighbourhood, and a stopping criterion. The method to calculate an initial solution is needed to start a local search, but is usually not seen as a part of the local search technique itself. For example, Chapter 3 contains a lot of results stating that a locally optimal solution with respect to some neighbourhood structure achieves a certain approximation guarantee, without going into the details of how to find an initial solution or sometimes even how to iterate to the next solution.

Results Local search techniques are widely used in practice, the most common example being for the traveling salesman problem. However, there are not much positive results on local search in theory. In fact, a complexity class called $\mathcal{PLS}$ (polynomial time local search) has been defined [JPY88]. Problems that are $\mathcal{PLS}$-complete are as hard as the hardest local optimization problems. A lot of well-known heuristics are known to be $\mathcal{PLS}$-complete. There are even some problems for which it has been proved that there is no sequence, of less than exponential length, of improvements ending in a locally optimal solution [Hal95]. To make matters worse, there are popular heuristics with good average-case behaviour that perform quite bad at relatively easy problems. For instance, the algorithm at the heart of the well-known simulated annealing performs quite poor at finding a maximum matching (which is even in $\mathcal{P}$).

On the positive side, set packing does not fit this negative theoretic framework. Currently all approximation algorithms for both the weighted and the unweighted k-set packing problem use local search at their core. However, there are also some results on the limits of the approximation guarantees using local search. The next chapter proceeds with approximation algorithms for k-set packing and the bounds on what is achievable using this technique.

Chapter 3

Current results

3.1 Unweighted approximations

In this section we briefly mention the state of the art of unweighted approximations for the k-set packing problem. There has been a long line of work in this area and Subsection 3.1.1 starts with the first $\frac{k}{2} + \varepsilon$-approximation given already in 1989 by Hurkens and Schrijver [HS89]. We give a proof and some intuition for the case of a 2-locally optimal solution.

Subsection 3.1.2 continues with a $\frac{k+2}{3}$-approximation [Hal95] and a $\frac{k+1}{3}+\varepsilon$-approximation [CGM13]. These search for improving sets of size $O(\log n)$, so the improved approximation guarantee is at the cost of the running time, which is $n^{O(\log n)}$ for both algorithms (also called quasi-polynomial running time).

Only this year these algorithms were adapted to run in polynomial time [Cyg13, SW13]. These results are mentioned in Subsection 3.1.3, together with an adaptation of the $\frac{k+2}{3}$-approximation to turn it into another $\frac{k+1}{3}$-approximation [FY13]. These are currently the best results.

Next, Section 3.2 treats the current state of the art of the weighted approximation algorithms, followed by the parameterized algorithms and the hardness results in Sections 3.3 and 3.4. The results on the linear programming and semidefinite programming relaxations are postponed to the next chapters, as these will be treated in more detail and we will provide a new bound for their integrality gap.

3.1.1 The first approximation algorithm

The unweighted k-set packing problem can be solved in polynomial time for $k = 2$ [Min80], so from now on assume $k \geq 3$.

2-locally optimal solution Perhaps the easiest local search technique to try is to iteratively search for an improving set of size 2. This is either one subset in $\mathcal{C}$ that does not intersect the current solution $\mathcal{A}$ or a pair of subsets that intersect at most one subset of $\mathcal{A}$. Starting with the empty solution and iteratively adding improving sets of size 2 yields an algorithm that is $\frac{k+1}{2}$-approximate [BNR95, Hal95].

First approximation algorithm A natural extension of this search technique is to search for improving sets of larger cardinality. If instead of improving sets of size 2 improving sets of constant size s are searched for increasing values of s, it is possible to obtain a polynomial time approximation ratio of $\frac{k}{2}+\varepsilon$. This was discovered by Hurkens and Schrijver [HS89] and this was actually the first approximation algorithm for k-set packing. Even though the problem was well-studied in the years that followed, this remained the best polynomial time approximation for over 20 years.

To be precise, they proved the following.

Theorem 8. *([HS89, Theorem 1]) Let $E_1, \ldots, E_m$ be subsets of the set V of size n, such that:*

1. *Each element of V is contained in at most k of the sets $E_1, \ldots, E_m$;*
2. *Any collection of at most t sets among $E_1, \ldots, E_m$ has a system of distinct representatives*[1]*.*

Then, we have the following:

$$\frac{m}{n} \leq \frac{k(k-1)^r - k}{2(k-1)^r - k} \quad \text{if } t = 2r-1;$$
$$\frac{m}{n} \leq \frac{k(k-1)^r - 2}{2(k-1)^r - 2} \quad \text{if } t = 2r.$$

Proof for $t = 2$ They provide a very keen yet complicated proof. We will give the proof and some intuition for the case $t = 2$, which establishes that any 2-locally optimal solution is at most a factor of $\frac{k+1}{2}$ away from the optimal solution. Let $\mathcal{A}$ be any 2-locally optimal solution and $\mathcal{B}$ be any optimal solution. Denote by $\mathcal{B}_1$ and $\mathcal{B}_2$ the sets in $\mathcal{B}$ that intersect $\mathcal{A}$ in one set respectively at least two sets. As $\mathcal{A}$ is 2-locally optimal there are no sets in $\mathcal{B}$ that do not intersect any set of $\mathcal{A}$. Note that every set in $\mathcal{A}$ intersects with at most k sets in $\mathcal{B}$. So

$$|\mathcal{B}_1| + 2|\mathcal{B}_2| \leq k|\mathcal{A}|.$$

[1] A system of distinct representatives of sets $\{E_i\}$ is a set $\{e_i\}$ such that $e_i \in E_i$ for all i and $e_i \neq e_j$ for $i \neq j$.

Since $\mathcal{A}$ is 2-locally optimal, $|\mathcal{B}_1| \leq |\mathcal{A}|$ and hence

$$2|\mathcal{B}| = |\mathcal{B}_1| + 2|\mathcal{B}_2| + |\mathcal{B}_1| \leq k|\mathcal{A}| + |\mathcal{A}| = (k+1)|\mathcal{A}|,$$

and therefore $\frac{|\mathcal{B}|}{|\mathcal{A}|} \leq \frac{k+1}{2}$.

Intuition Intuitively the argument boils down to the following. Look at the intersection graph of the current 2-locally optimal solution $\mathcal{A}$ and some optimal packing $\mathcal{B}$. The sets in $\mathcal{B}$ that conflict with just one set in $\mathcal{A}$ are not particularly interesting as they do not form an improving set. Let us delete these sets from the intersection graph together with the sets in $\mathcal{A}$ they intersect. Because $\mathcal{A}$ is a 2-locally optimal solution, the sets in $\mathcal{B}$ conflicting with two sets in $\mathcal{A}$ still have degree at least 1, while the degrees of the other sets can be anything. That is fine, because in general when every degree is at most two we are done: the number of edges between $\mathcal{A}$ and $\mathcal{B}$ is then at most $2m$ and this number is upper bounded by nk and hence $2m \leq nk$. So the core of the argument really consists of small conflicts posing no problem and concentrating on the remainder of the graph. The iterative process of deleting these sets cannot go on forever, because there are only so many vertices in $\mathcal{A}$ and possibly much more in $\mathcal{B}$.

3.1.2 Quasi-polynomial time algorithms

A $\frac{k+2}{3}$-approximation Halldórsson [Hal95] was the first to find $\frac{k+2}{3}$-approximate solutions. His clever insight is that using a search space of improving sets of size $O(\log n)$ allows to improve the approximation guarantee from about $\frac{k}{2}$ to about $\frac{k}{3}$. However, in a straightforward implementation of an iterative algorithm with such a search space, the running time would be quasi-polynomial, i.e. $n^{O(\log n)}$. The deduction of this running time is straightforward: one can find an improving set of size $O(\log n)$ in time $n^{O(\log n)}$, and the number of iterations is trivially upper bounded by n.

A $\frac{k+1}{3} + \varepsilon$-approximation This year, Cygan, Grandoni and Mastrolilli [CGM13] improved upon this result and showed that the approximation guarantee of an $O(\log n)$-locally optimal solution can in fact be bounded by $\frac{k+1}{3} + \varepsilon$. The running time is still quasi-polynomial. Their analysis is more involved and they use a lemma from [BF94] as a black box. In Chapter 4 the same lemma is used to prove an improved bound on the integrality gap of an LP formulation for the k-set packing problem.

3.1.3 Polynomial time algorithms

A $\frac{k+2}{3}$-approximation Sviridenko and Ward [SW13] recently established an elegant polynomial time approximation algorithm that beats the approximation guarantee of $\frac{k}{2}$ and this is the first polynomial time improvement over the $\frac{k}{2}+\varepsilon$ result from [HS89]. Their approximation guarantee is $\frac{k+2}{3}$.

Their key insight is that using improving sets with a special structure in the analysis of Halldórsson [Hal95] allows to reduce the running time. They look at a graph with a vertex for every set in the current solution $\mathcal{A}$. Two vertices S, T are connected if there is a set in $\mathcal{C}\setminus\mathcal{A}$ that intersects $\mathcal{A}$ only in S and T. Possibly $S = T$, which gives rise to a loop; so every set that intersects with at most two sets in $\mathcal{A}$ corresponds to one edge. Then they define three canonical improvements, which are three types of connected graphs containing two distinct cycles, thus having more edges than vertices. They iteratively search for canonical improvements, and when one is found they remove the vertices from $\mathcal{A}$ and add the edges from $\mathcal{C} \setminus \mathcal{A}$ to the current solution.

Another key insight is then to combine this with the famous color-coding technique [AYZ95] and an application of dynamic programming, which allows them to efficiently search for these canonical improvements. The result is a new polynomial time approximation after more than 20 years.

A $\frac{k+1}{3}+\varepsilon$-approximation Only two months later the polynomial time approximation guarantee was cleverly improved to $\frac{k+1}{3}+\varepsilon$ by Cygan [Cyg13]. He adapted the quasi-polynomial $\frac{k+1}{3}$-approximation from [CGM13], also with the sharp insight that structured sets reduce the search space and the running time without losing the good approximation guarantee. His algorithm searches for elegant structures that he calls improving sets of bounded pathwidth. Again using the color-coding technique, the algorithm can search the space of $O(\log n)$ improving sets of bounded pathwidth in polynomial time.

Another $\frac{k+1}{3}+\varepsilon$-approximation Independently, another $\frac{k+1}{3}+\varepsilon$ polynomial time approximation algorithm was found four months later by Fürer and Yu [FY13]. This is an ingenious improvement of the polynomial time $\frac{k+2}{3}$-approximation of [SW13]. Let $\mathcal{A}$ denote the current solution again. Define a bipartite auxiliary graph $G_{\mathcal{A}}$ with a vertex for every set in $\mathcal{A}$ in one colour class and a vertex for every set in $\mathcal{C} \setminus \mathcal{A}$ in the other colour class. Connect two vertices from both colour classes if and only if the sets corresponding to them intersect. Indeed, this is the conflict graph of the instance with only a subset of the edges.

The algorithm starts by looking for improving sets of constant size as in [HS89]. Then they guess $O(\log n)$ sets $\mathcal{I}$ in $\mathcal{C} \setminus \mathcal{A}$ that might form an improving set. Now consider any collection of sets $\mathcal{I}_3$ within $\mathcal{I}$ of degree at least 3 in the graph $G_{\mathcal{A}}$. For such a collection, the algorithm looks for a sequence of replacements that swaps t sets in $\mathcal{A}$ with t sets outside $\mathcal{A}$. The idea is to increase the number of sets in $\mathcal{C} \setminus \mathcal{A}$ that intersect $\mathcal{A}$ in at most two sets. If the degree of the vertices in $\mathcal{I}_3$ now drops to 2 or less, they check whether these vertices together with sets of degree at most 2 in $\mathcal{I}$, form a canonical improvement as in [SW13]. If so, this canonical improvement is added to obtain a solution of larger cardinality.

Again the color-coding technique is exploited to find all structures in polynomial time. In comparison, the local improvements in this paper are less general than in the other $\frac{k+1}{3} + \varepsilon$ polynomial time approximation by Cygan. This results in the fact that this new analysis is simpler than the one in [Cyg13].

3.2 Weighted approximations

This section considers approximation algorithms for weighted k-set packing. In the weighted k-set packing problem, every set S in $\mathcal{C}$ is associated with a weight $w(S)$ and the objective is to maximise the total weight of the packing rather than its cardinality.

Also the weighted k-set packing problem can be solved in polynomial time for $k = 2$ [NT99], so from now on assume $k \geq 3$.

3.2.1 Three close to k-approximations

A k-approximation The first approximation algorithm for weighted k-set packing is due to Hochbaum [Hoc83] who achieved an approximation guarantee of k. His algorithm first preprocesses the input and then more or less greedily adds sets to the current solution.

A $k - 1 + \frac{1}{k}$-approximation A weighted $k - 1 + \frac{1}{k}$-approximation algorithm was found in [BNR95]. They show that the simple greedy algorithm approximates the weighted problem within a factor of k, where the greedy approach successively selects the subset with the highest weight from all subsets that do not intersect any selected subset so far. Using a similar local search analysis as in the unweighted case they are able to improve this bound to $k - 1 + \frac{1}{k}$.

A $k - 1 + \varepsilon$-approximation This result was slightly improved by Arkin and Hassin [AH97]. Their setting is more general in the following sense. They consider a bipartite graph $G = (U \cup V, E)$ where every vertex is associated with a weight. They use the

shorthand notation $w(X) = \sum_{x \in X} w(x)$ for a subset of the vertices X, and write E_v for the neighbourhood $N(v)$ of some vertex v. Then analogously to [HS89] they assume that $|E_v| \leq k$ for all $v \in V$ and that any subset $R \subseteq U$ of at most t nodes satisfies $w(R) \leq w\left(\bigcup_{u \in R} E_r\right)$.

Their main theorem is that $\frac{w(U)}{w(V)} \leq k - 1 + \frac{1}{t}$. The $k - 1 + \frac{1}{k}$-approximation from [BNR95] is the case where $k = t$, so this result is slightly more general. Effectively, they reached an approximation guarantee of $k - 1 + \varepsilon$.

3.2.2 A $\frac{4k+2}{5}$ and a $\frac{2k+2}{3}$-approximation

The algorithm Chandra and Halldórsson [CH99] were the first to beat this bound and obtain an approximation guarantee of $\frac{2k+2}{3}$. They combine the greedy algorithm to find an initial solution and local search techniques to improve upon this solution. Essentially they show that either the greedy solution is already good, or the local search improves it quite well. They reduce the problem to the independent set problem in a $k+1$-claw free graph G, and their local search finds an improving claw. This is a claw C whose center vertex v is in the current solution A in G, and hence the talons T_C of the claw (its other vertices forming an independent set) are not in A. They show that using local search to find any improving claw leads to an approximation guarantee of $\frac{4k+2}{5}$, while a local search for the best local improvement is $\frac{2k+2}{3}$-approximate. With the best improvement they mean the claw C with the maximum ratio between the sum of the weights of T_C and the sum of the weights of the neighbours of T_C in A.

Time complexity To show that the algorithm runs in polynomial time, they need to modify it a bit. First they find a solution A using the greedy approach. Then they rescale the weight function such that $w(A) = kn$. Then they keep searching for the best local improvement using the weight function $\lfloor w \rfloor$. Now each iteration increases $\lfloor w \rfloor(A)$ by at least one. Since $\mathcal{A}$ is at most a factor of k away from the optimal solution, the number of iterations is bounded by $k^2 n$. Since in every turn they only inspect a polynomial number of candidates, the algorithm now runs in polynomial time.

3.2.3 A close to $\frac{2k}{3}$-approximation

Berman and Krysta [BK03] improved the approximation guarantee slightly. For every k they find the optimal value for α such that any 2-locally optimal solution with respect to the "misdirected" weight function w^α achieves the best approximation guarantee. Surprisingly there are only three distinct values for α that cover all values of k.

There is an optimal value of α for $k = 3$ which yields an approximation guarantee of $\frac{2k}{3} \approx 0.66667k$, there is an optimal value for $k = 4$ which approximates the problem within $\frac{\sqrt{13}-1}{4}k \approx 0.65139k$ and there is a value that is optimal for $k \geq 5$, resulting in a $2^{-\log_3 2}k \approx 0.64576k$-approximation.

3.2.4 A $\frac{k+1}{2}$-approximation algorithm

Currently the best approximation algorithm for weighted k-set packing is from Berman [Ber00], also using another objective function w^2 rather than w. Its approximation guarantee is $\frac{k+1}{2}$. Berman defines a function charge(u, v) and searches for minimal claws that satisfy some condition on this charge function, called nice claws. He shows that if this algorithm terminates it achieves the desired approximation guarantee. To show that the algorithm terminates, he shows that every nice claw improves the square of the weight function. Similar to the running time analysis of [CH99], he shows that the number of iterations where w^2 improves is polynomially bounded. And when there is no more claw that improves w^2, there is no nice claw anymore, and hence the algorithm terminates. Therefore it runs in polynomial time and achieves the approximation guarantee $\frac{k+1}{2}$. Chronologically, this result came before the result from the previous subsection [BK03].

Chapter 6 dives into the details of this paper and we show a simplified and more intuitive proof of the fact that a nice claw improves w^2.

3.3 Parameterized complexity

Background Due to the limited results in the area of approximation algorithms for a long time, people started to study the parameterized problem where the cardinality of the solution is assumed to be m. In the k-set m-packing problem the goal is to find m disjoint sets, each of size k. The running time of exact algorithms for this problem can be written in the form $f(m)poly(n)$, moving the exponential dependency to the parameter m instead of n. There have been huge improvements on the running time of these exact algorithms. This is in spite of the fact that the k-set m-packing problem is $W[1]$-complete with respect to the parameter m [DF99] (see Subsection 3.4.3 for details about $W[1]$-completeness). These results contain some algorithms especially designed for the case $k = 3$. Table A.1 in Appendix A gives an overview of the results on the 3-set m-packing, and we refer to Table A.2 in Appendix A for the more general k-set m-packing problem. Some results also extend to the weighted case, this is mentioned in the last column. We follow the convention from parameterized algorithms to let $O^*(f(m))$ denote $f(m)n^{O(1)}$.

Improvements Some remarks in the last column of these tables require some explanation. Koutis' [Kou05] original result was an $O^*(2^{O(m)})$ time algorithm. In [CLSZ07, LLCS06] it was pointed out that the constants are huge in this approximation, showing the bound is at least $O^*(32000^{3m})$ when $k = 3$. Like many results in this area, Koutis derandomised his algorithm using the color-coding technique from Alon, Yuster and Zwick [AYZ95]. In [CLSZ07, LLCS06] a new perfect hashing technique was introduced, with which Koutis' result could be improved to $O^*(25.6^{mk})$. A similar result is true for the deterministic algorithm by Fellows et al [FKN+08]. Its original bound was $exp(O(mk))$, which was actually $O^*((12.7D)^{3m})$ for some $D \geq 10.4$ (when $k = 3$). This is improved to $O^*(13.78^{mk})$ using the new perfect hash function.

The first deterministic result of the form $O^*(g(m, k))$ for k-set m-packing is due to Jia, Zhang and Chen [JZC04]. Currently the best complexity result (randomised) result for k-set m-packing is $O^*(f(m, k))$ by Björklund et al [BHKK10]. This function is not very readable or insightful, but behaves well for small k. For example, when $k = 3$, we have $O^*(f(3, m)) \approx O^*(1.493^{3m})$, which is the result mentioned in Table A.1. The function is strictly smaller than $O^*(2^{mk})$.

Hardness Some of the fastest parameterized algorithms rely on group algebra theory. A variable is introduced for every element and a subset is the product of its variables. A packing then corresponds to a multilinear polynomial of degree mk. Koutis and Williams [KW09] showed that detecting such a polynomial cannot be done in their model in time faster than $O^*(2^{mk})$. These results still hold if the color-coding method is used ([AYZ95]) or the randomised divide-and-conquer approach ([CKL+09, CLSZ07]).

This is the only result on the limit of the time complexity of these parameterized algorithms. There are more hardness results known for the approximation algorithms of non-parameterized k-set packing. The next section captures an outline of these hardness results.

3.4 Hardness results

Subsection 3.4.1 considers some results on the general set packing problem. Subsections 3.4.2, 3.4.3 and 3.4.4 consider hardness results specifically for k-set packing. These are respectively a result on the hardness of approximation, on the non-existence of certain algorithms and on the limits of local search techniques for this problem.

Covering-packing dualities	
Minimum set cover	Maximum set packing
Minimum vertex cover	Maximum matching
Minimum edge cover	Maximum independent set

Table 3.1: Every horizontal pair of problems are each other's LP-duals.

3.4.1 Hardness of set packing

Set packing is one of the standard packing problems, which are closely related to covering problems. Table 3.1 lists some of these problems. Set packing is the most general of these packing problems, together with its LP-dual minimum set cover. We refer to [ADP80] for a survey of these problems.

The following results on the hardness of the problems also apply to the set packing problem. Arora et al showed that, unless $\mathcal{P} = \mathcal{NP}$, the vertex cover does not admit a polynomial time approximation scheme, even on bounded degree graphs [ALM+98]. Moreover, also assuming that $\mathcal{P} \neq \mathcal{NP}$, it has been shown that there is no constant-ratio polynomial time approximation scheme for independent set [GJ79, PS82]. Håstad proved that set packing cannot be approximated within $n^{1-\varepsilon}$, where n is the number of sets, unless $\mathcal{NP} = \mathcal{ZPP}$ [Hås96]. On the positive side, set packing can be approximated within a factor of $\sqrt{N}$ [HKT00] (recall that N is the number of elements in $\mathcal{U}$). Håstad's result also implies this is the best possible assuming $\mathcal{NP} \neq \mathcal{ZPP}$.

3.4.2 Hardness of approximation

As noted in Subsection 3.1.3, currently the best approximation guarantee for unweighted k-set packing is $\frac{k+1}{3} + \varepsilon$. Hazan, Safra and Schwarz [HSS06] showed the following hardness of approximation result. There is still a gap to bridge between the currently best approximation guarantee and this bound.

Theorem 9. *([HSS06]) It is NP-hard to approximate k-set packing in polynomial time within a factor of $O\left(\frac{k}{\log k}\right)$.*

Here is a rough outline of their argument. Define gap-P-$[a, b]$ to be the following decision problem: decide on an instance of the decision problem P whether there exists a fractional solution of size at least b or whether every solution of the given instance is of fractional size smaller than a. Then if gap-P-$[a, b]$ is NP-hard, so is approximating P within a factor smaller than $\frac{b}{a}$.

Define MAX-3-LIN-q as the optimization problem where a set of linear equations over $GF(q)$ is given, each depending on 3 variables, and one needs to find an assignment maximising the number of satisfied equations. Håstad [Hås01] proved that gap-MAX-3-LIN-q-$[\frac{1}{q}+\varepsilon, 1-\varepsilon]$ is NP-hard, which was a central result in the theory of hardness of approximation.

In [HSS06] they provide a polynomial time reduction from MAX-3-LIN-q to k-uniform hypergraph matching (i.e. to k-set packing). They use what they call a (q,δ)-Hyper-Graph-Edge-Disperser as a gadget (see [RTS00] for background on dispersers and extractors). They construct such a gadget for every variable occurring in the equations. The set of all these gadgets is then the set of vertices of a hypergraph H. By a clever construction of the hyperedges they relate the satisfied equations to a packing in this hypergraph. This enables them to show that gap-k-SP-$[\frac{4}{q^3}+\varepsilon, \frac{1}{q^2}-1]$ is NP-hard. As they have $k=\Theta(q\log q)$, they find a bound for the inapproximability factor for k-set packing of $\Omega(\frac{k}{\log k})$.

3.4.3 Non-existence of certain algorithms

There are also some other hardness results known, more specifically on the existence of certain algorithms or on the limits of what is achievable using local search techniques. Cygan [Cyg13] proved the following.

Theorem 10. *([Cyg13, Theorem 1.1]) It is $W[1]$-hard to search the whole space of improving sets of size r efficiently.*

More formally, unless $FPT = W[1]$, there is no $f(r)poly(n)$ time algorithm that given a family $\mathcal{C} \subseteq 2^{\mathcal{U}}$ of sets of size 3 and a disjoint subfamily $\mathcal{A} \subseteq \mathcal{C}$ either finds a bigger disjoint family $\mathcal{B} \subseteq \mathcal{C}$ or verifies that there is no disjoint family $\mathcal{B} \subseteq \mathcal{C}$ such that $|\mathcal{A}\backslash\mathcal{B}|+|\mathcal{B}\backslash\mathcal{A}| \leq r$.

Fixed parameter tractability This theorem requires some explanation (see e.g. [CHKX05, DF99]). FPT is the set of fixed parameter tractable problems, which are problems with input size n that can be solved in time $f(k)n^{O(1)}$ for some function f. FPT thus classifies problems according to multiple parameters rather than a single parameter, where it is crucial that functions $f(n,k)$ like n^k are not allowed. If the value of k is fixed, the problem is said to be parameterized, which is the setting of Subsection 3.3 with parameter m.

W-hierarchy of FPT The W-hierarchy $\bigcup_{t\geq 0} W[t]$ has been introduced to formalise the level of intractability of problems. We have $FPT = W[0]$ and $W[i] \subseteq W[j]$ whenever $i \leq j$. We do not go into the details and refer the interested reader to [CHKX05, DF99].

Theorem 10 assumes $FPT \neq W[1]$, which is a widely believed assumption. In particular, this assumption is equivalent to the famous Exponential Time Hypothesis (ETH) [IP99]. If $FPT = W[1]$ then the ETH fails and vice versa [Cyg13, CHKX05]. When a problem is $W[1]$-complete (i.e. not solvable in polynomial time unless $FPT = W[t]$, here for $t = 1$), this is strong evidence that the problem is probably not fixed parameter tractable. For example, the clique problem and the independent set problem are $W[1]$-complete and the dominating set problem and the set cover problem are $W[2]$-complete.

The assumption in Theorem 10 is thus plausible, and if this is true then there is no polynomial time algorithm which searches the whole space of size r improving sets.

3.4.4 Limits of local search

Next to this result on the existence of such an algorithm, there is also a lower bound on the approximation guarantee that algorithms based on local search techniques can achieve. Sviridenko and Ward [SW13] proved the following.

Theorem 11. *([SW13, Theorem 6.1]) The locality gap of a t-locally optimal solution is at least $\frac{k}{3}$, even when t is allowed to grow on the order of n.*

More formally, let $c = \frac{9}{2e^5 k}$ and suppose that $t \leq cn$ for all sufficiently large n. Then there exist 2 pairwise disjoint collections of k-sets $\mathcal{A}$ and $\mathcal{B}$ with $|\mathcal{A}| = 3n$ and $|\mathcal{B}| = kn$ such that any collection of $a \leq t$ sets in $\mathcal{B}$ intersects with at least a sets in $\mathcal{A}$.

This result shows that even local search algorithms that are allowed to examine some exponential number of possible improvements at each stage cannot achieve an approximation guarantee better than $\frac{k}{3}$. This suggests that local search algorithms, which currently achieve an approximation guarantee of $\frac{k+1}{3} + \varepsilon$, are not a good approach to beat the approximation guarantee much further. Fürer and Yu [FY13] extended this result.

Theorem 12. *([FY13, Theorem 7]) There is an instance for k-set packing with locality gap $\frac{k+1}{3}$ such that there is no local improvement of size up to $O(n^{\frac{1}{5}})$.*

More formally, for any $t \leq \left(\frac{3e^3 n}{k}\right)^{\frac{1}{5}}$ there exist two disjoint collections of k-sets $\mathcal{A}$ and $\mathcal{B}$ with $|\mathcal{A}| = 3n$ and $|\mathcal{B}| = (k+1)n$ such that any collection of t sets in $\mathcal{A}$ intersects with at least t sets in $\mathcal{B}$.

So local search has reached its limits for all practical purposes. To achieve an approximation guarantee beating the order of $\frac{k}{3}$, other approaches to the problem are necessary to consider. Another approach one could take is the LP and SDP relaxations for k-set packing. The next chapters treats these two approaches and show improved bounds on their integrality gap.

Chapter 4

LP formulation

This chapter treats the current results on the LP relaxation of k-set packing. Section 4.1 starts with the results for the standard LP relaxation. Then Section 4.2 shows how to strengthen this LP and we prove a new bound for the integrality gap of this strengthened LP. Finally Section 4.3 presents how to decrease the size of this LP to a polynomial size.

4.1 The standard LP

LP-formulation View the set packing problem as the hypergraph matching problem like in Subsection 2.2.1. Introduce a variable x_e for every hyperedge e that indicates whether e is included in the matching or not. The objective is to maximise $\sum_{e \in E} x_e$ such that every vertex is covered only once. For convenience we introduce the following notation.

$$x(F) := \sum_{e \in F} x_e.$$

Hence the natural linear program looks like the following, denoted by (LP). As before, $\delta(v)$ denotes the set of hyperedges incident to v.

$$\begin{aligned} \max \quad & x(E) \\ \text{s.t.} \quad & x(\delta(v)) \leq 1, \quad \forall v \in V \\ & 0 \leq x_e \leq 1, \quad \forall e \in E \end{aligned} \tag{LP}$$

Results Füredi, Kahn and Seymour [FKS93] have shown that the integrality gap of (LP) equals $k - 1 + \frac{1}{k}$ for k-uniform hypergraphs. They also show that in the case of a k-partite hypergraph the integrality gap equals $k-1$, but both proofs are non-algorithmic.

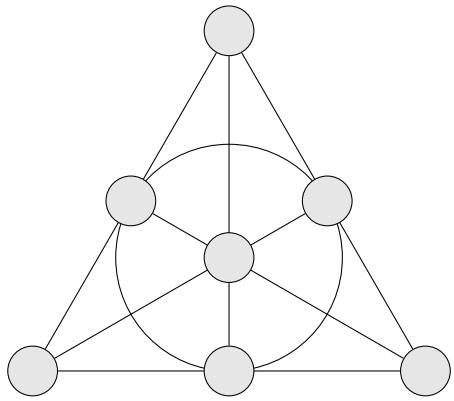

Figure 4.1: The Fano plane. A hyperedge is represented by a line connecting three vertices.

Chan and Lau [CL10] gave an algorithmic proof of these facts and showed that the bounds are tight. Both results also extend to the weighted case.

Tight example As a tight example for the non-k-partite case, consider the projective plane of order $k-1$. This is a hypergraph that is k-uniform (every hyperedge has cardinality k), k-regular (every vertex has degree k), in which every pair of hyperedges intersects in one vertex, and in which every pair of vertices is contained in exactly one hyperedge. Equivalently it is the Steiner system $S(2, k, k^2 - k + 1)$. A projective plane of order k exists if k is a prime power and the conjecture that this is also a sufficient condition is a long standing open question. The projective plane of order 2 (thus corresponding to 3-set packing) is the well-known Fano plane. Figure 4.1 depicts a Fano plane, where a hyperedge is represented by a line connecting three vertices.

To see that the projective plane of order $k-1$ is a tight example, note that the integral solution to this k-set packing instance equals 1 as every hyperedge intersects every other hyperedge. But fractionally, it is possible to set $x_e = \frac{1}{k}$ for every hyperedge because the hypergraph is k-regular. This is a feasible solution to (LP) and since the hypergraph has $k^2 - k + 1$ hyperedges the integrality gap equals $\frac{1}{k}(k^2 - k + 1) = k - 1 + \frac{1}{k}$.

4.2 Strengthening the LP formulation

4.2.1 The intersecting family LP

The LP formulation can be strengthened by adding extra local constraints. Call a family of hyperedges intersecting if every two of them overlap in at least one vertex. In the conflict graph of an instance, an intersecting family $\mathcal{F}$ would be a clique F. Obviously,

from every intersecting family only one hyperedge can be picked. For every intersecting family thus an extra constraint can be added to obtain the following strengthened LP. Let $\mathcal{K}$ denote the collection of all intersecting families.

$$\begin{aligned} \max \quad & x(E) \\ \text{s.t.} \quad & x(\delta(v)) \leq 1, \quad \forall v \in V \\ & x(K) \leq 1, \quad \forall K \in \mathcal{K} \\ & 0 \leq x_e \leq 1, \quad \forall e \in E \end{aligned} \tag{Intersecting family LP}$$

This is called the interesting family LP. Note that this LP in general has exponentially many constraints and hence it is not solvable in polynomial time. In Section 4.3 we will show that for each k the LP can be rewritten into an LP with a number of constraints that is polynomial in n. In [CL10], they proved the following theorem about this intersecting family LP.

Theorem 13. *([CL10, Theorem 4.1]) The ratio between any LP solution to the intersecting family LP and any 2-locally optimal solution is at most $\frac{k+1}{2}$. Thus the integrality gap of the intersecting family LP is at most $\frac{k+1}{2}$.*

We omit the proof, because we will show an improved bound in Theorem 17. In order to proof this, the lemmas from [BF94] treated in the next subsections are necessary.

4.2.2 A lemma on multigraphs

Lemma 14. *([BF94, Lemma 3.1]) Assume that every vertex in a multigraph[1] $G = (V, E)$ has degree at least 3. Then every vertex $v \in V$ belongs to a connected induced subgraph $G[X]$ with strictly more edges than vertices, of at most $4 \log_2 n - 1$ vertices.*

Proof. This is a slightly more detailed proof than in [BF94]. Let $G = (V, E)$ be a multigraph where every vertex has degree at least 3 and let $v \in V$ be an arbitrary vertex. Consider a breadth-first search tree T of G rooted at v. The distance of a vertex in T is the length of the shortest path from the vertex to v.

Let $m > 0$ and suppose every vertex in T at a distance less than m has at least two children. Then T has at least 2^m vertices. Since T has n vertices, $m \leq \log_2 n$. Since v has degree at least 3, in fact $m < \log_2 n$. This implies that there must be a vertex u at a distance of at most $\log_2 n - 1$ having at most one child.

[1] A multigraph is a graph where multiple edges and loops are allowed.

Since u has at least degree 3 but at most one child, one of the edges incident to u in T must be a cross edge, a loop or a multiple edge. Let's denote this edge by $e = \{u, w\}$, where possibly $w = u$. Then the tree paths from v to u and from v to w together with the edge e form an induced subgraph $G[Y]$ of at least as many edges as vertices, and this induced subgraph has at most $2\log_2 n$ vertices.

If the number of edges in $G[Y]$ is already strictly larger than the number of vertices in $G[Y]$, then the proof is finished. Otherwise the number of edges equals the number of vertices. In this case, shrink $G[Y]$ to one single vertex y. In this modified graph G', every vertex still has degree at least 3. Therefore G' contains another induced subgraph induced by some $Z \cup \{y\}$ with at least as many edges as vertices, of size at most $2\log_2 n$. But then $G[Y] \cup G[Z]$ contains strictly more edges than vertices in G and its size is at most $4\log_2 n - 1$, because y is not a vertex in G. So $G[Y \cup Z]$ is an induced subgraph of G that satisfies the required properties, as $G[Y \cup Z]$ has at least as many edges as $G[Y] \cup G[Z]$. □

4.2.3 Another lemma on multigraphs

Using this lemma following lemma can be proved, which is the one really needed for Theorem 17.

Lemma 15. *([BF94, Lemma 3.2]) For any integer $h \geq 1$, any undirected multigraph $G = (V, E)$ with $|E| \geq \frac{h+1}{h}|V|$ contains a set X of less than $4h\log_2 n$ vertices, such that in $G[X]$ there are more edges than vertices.*

Proof. Also this is a slightly more detailed proof than in [BF94]. Let h be any positive integer. For convenience, for any undirected multigraph $H = (V(H), E(H))$ denote $|V(H)| = V_H$, $|E(H)| = E_H$. H is said to satisfy Property $(*)$ if $E_H \geq \frac{h+1}{h}V_H$. Given a set of vertices U in a multigraph H, denote $V_U := V_{H[U]}$, $E_U := E_{H[U]}$. U is said to satisfy Property $(*)$ if its induced subgraph $H[U]$ satisfies Property $(*)$.

Let $G = (V, E)$ be an undirected multigraph that satisfies Property $(*)$, so $|E| \geq \frac{h+1}{h}|V|$. We need to show that T contains a set X of less than $4h\log_2 n$ vertices such that in $G[X]$ there are more edges than vertices.

Let $U \subseteq V$ be the smallest set of vertices in G that satisfies Property $(*)$. Because U is a minimal set, $G[U]$ cannot have a vertex u of degree 1: otherwise $U \setminus \{u\}$ would be a smaller set satisfying Property $(*)$.

Now consider a maximal chain of degree 2 vertices C in $G[U]$ and denote the two vertices adjacent to C, its endpoints in $G[U] \setminus C$, by x and y. First we show that C has less than h vertices. Assume the contrary and suppose C has exactly h vertices.

We claim that $U \setminus C$ is a smaller set of vertices that satisfies Property $(*)$ which is a contradiction. Note that $E_{U\setminus C} = E_U - h - 1$ because $G[C]$ contains $h-1$ edges and is connected to the rest of U by 2 other edges. Also $V_{U\setminus C} = V_U - h$. So we need to show that $E_U - h - 1 \geq \frac{h+1}{h}(V_U - h)$. But this is true: subtract $h+1$ on both sides from the known fact that $E_U \geq \frac{h+1}{h}V_U$ to get to this equation. So if C contains h vertices, U is not the smallest set of vertices satisfying Property $(*)$, which is a contradiction. Note that the argument still holds if C has $p > h$ vertices: then on the left-hand side $p+1$ is subtracted while on the right-hand side $\frac{h+1}{h}p = p + \frac{p}{h} > p+1$ is subtracted, so the left-hand side is indeed still larger than the right-hand side. We conclude that C has strictly less than h vertices.

Now modify the graph $G[U]$. Replace every maximal chain of degree 2 vertices C by a single edge connecting its endpoints x and y. Since any two such chains do not intersect (except possibly at their endpoints), every such chain is replaced independently from another. Denote the resulting graph, obtained from $G[U]$ by contracting all such chains, by G'. Then G' does not have any vertices of degree 2 as they are all contracted.

Therefore all vertices in G' have degree at least 3, and Lemma 14 applies. Hence G' contains some induced subgraph $G'[X']$ of size $m \leq 4\log_2 n - 1$ with strictly more edges than vertices. Now select $m+1$ edges in $G'[X']$ and replace the edges that were chains in $G[U]$ by their respective chains of degree 2 vertices. Denote the resulting graph, obtained from $m+1$ edges from $G'[X']$ by expanding the chains, by $G[X]$ (note that $G[X]$ is indeed an induced subgraph of G so this notation is valid).

Note that in the expanding of the chains, at most $m+1$ chains are expanded and every chain has size less than h. So $G[X]$ has in total at most $m + (m+1)(h-1) = (m+1)h - 1$ vertices. As $m \leq 4\log_2 n - 1$ we see that $V_X = (m+1)h - 1 < 4h\log_2 n$ vertices and the proof is complete. □

The following lemma is an immediate consequence.

Lemma 16. *(Special case of Lemma 15) Let $\varepsilon > 0$. If in an undirected multigraph $G = (V, E)$ there is no subset of vertices X of size at most $4(1+\frac{1}{\varepsilon})\log_2 n$ such that in $G[X]$ there are more edges than vertices, then $|E| \leq (1+\varepsilon)|V|$.*

4.2.4 A new bound on the integrality gap

Using this last lemma, the following improved bound on the integrality gap of the intersecting family LP can be established.

Theorem 17. *Let $\varepsilon > 0$. The ratio between any LP solution to the intersecting family LP and any $4(1+\frac{1}{\varepsilon})\log_2 n$-locally optimal solution is at most $\frac{k}{3}+1+\varepsilon$. Thus the integrality gap of the intersecting family LP is at most $\frac{k}{3}+1+\varepsilon$.* ***[NOTE: see page iii]***

Proof. Let M be a $4(1+\frac{1}{\varepsilon})\log_2 n$-locally optimal matching. Let x be a feasible solution to the intersecting family LP, and let $\mathcal{F}$ be the set of hyperedges with $x_e > 0$. To prove the theorem it suffices to prove that $x(\mathcal{F}) \leq \left(\frac{k}{3}+1+\varepsilon\right)|M|$. Denote by $\mathcal{F}_1$, $\mathcal{F}_2$ and $\mathcal{F}_{3+}$ the subsets of $\mathcal{F}$ in which every hyperedge in intersects exactly one, exactly two or at least three hyperedges in M, respectively.

Note that M is in particular a 1-local and a 2-locally optimal matching. Since M is a 1-locally optimal matching, each hyperedge e in $\mathcal{F}$ intersects at least one hyperedge in M: otherwise $M \cup \{e\}$ would be a larger matching. Hence $\mathcal{F} = \mathcal{F}_1 \cup \mathcal{F}_2 \cup \mathcal{F}_{3+}$. We will now proceed to bound $x(\mathcal{F}_1)$ and $x(\mathcal{F}_2)$ in terms of $|M|$.

Consider a hyperedge e in M and define $\mathcal{F}_1(e) := \{f \in \mathcal{F}_1 \mid f \cap e \neq \emptyset\}$. Then $\mathcal{F}_1(e)$ is an intersecting family. For suppose this is false, then there are two disjoint hyperedges f_1, f_2 in $\mathcal{F}_1$. These are disjoint from all other hyperedges in $M \setminus \{e\}$ because $f_1, f_2 \in \mathcal{F}_1$. Therefore $M - e + f_1 + f_2$ is a larger matching than M, but this is in contradiction with the fact that M is a 2-locally optimal matching. Hence $\mathcal{F}_1(e)$ is an intersecting family. Thus by the intersecting family constraint $x(\mathcal{F}_1(e)) \leq 1$. Now summing over all hyperedges $e \in M$ yields

$$x(\mathcal{F}_1) \leq |M|. \tag{4.1}$$

For the purpose of the analysis of the bound on $x(\mathcal{F}_2)$, consider an auxiliary multigraph H with a vertex for every set in M. Two vertices $m_1, m_2 \in M$ are adjacent in H if and only if there is a set in $\mathcal{F}_2$ that intersects both sets corresponding to m_1 and m_2. Note that H consists of exactly $|M|$ vertices and $|\mathcal{F}_2|$ edges, potentially some of them are parallel. By assumption, M cannot be improved by a set of size at most $4(1+\frac{1}{\varepsilon})\log_2 n$. As an induced subgraph of H with more edges than vertices constitutes an improving set, H cannot contain an induced subgraph of size at most $4(1+\frac{1}{\varepsilon})\log_2 n$ with more edges than vertices. Hence, by Lemma 16 we infer that $|E(H)| \leq |V(H)|(1+\varepsilon)$. Consequently, $|\mathcal{F}_2| \leq (1+\varepsilon)|M|$. As $x(\mathcal{F}_2) \leq |\mathcal{F}_2|$ it follows that

$$x(\mathcal{F}_2) \leq (1+\varepsilon)|M|. \tag{4.2}$$

Now that $x(\mathcal{F}_1)$ and $x(\mathcal{F}_2)$ are upper bounded in terms of $|M|$, the result follows easily.

Note that there are $k|M|$ vertices in M. Then the degree constraint yields

$$\begin{aligned} k|M| &\geq x(\mathcal{F}_1) + 2x(\mathcal{F}_2) + 3x(\mathcal{F}_{3+}) \\ &= 3x(\mathcal{F}) - x(\mathcal{F}_2) - 2x(\mathcal{F}_1). \end{aligned}$$

Now plug in the bound from (4.1) and (4.2) to find

$$k|M| \geq 3x(\mathcal{F}) - (1+\varepsilon)|M| - 2|M|,$$

which can be rewritten as

$$x(\mathcal{F}) \leq \frac{k+3+\varepsilon}{3}|M| \leq \left(\frac{k}{3} + 1 + \varepsilon\right)|M|. \qquad \square$$

4.3 A polynomially sized LP

A new bound on the integrality gap of the intersecting family LP has now been established, but the LP might have exponentially many constraints and thus not solvable in polynomial time. This section treats the result from [CL10] that for constant k only a polynomial number of constraints can be added to the standard LP formulation such that every intersecting family has a fractional value of at most 1. To this end the definition of a kernel is needed. Intuitively, for an intersecting family K its kernel is a subset U of the vertices covered by K such that all hyperedges restricted to U still form an intersecting family. More formally, for each hyperedge e define $e_U = e \cap U$, and for a collection of hyperedges K define $K_U = \{e_u \mid e \in K\}$. Then U is a kernel for an intersecting family K if K_U is an intersecting family.

We proceed with the following result from [CK64].

Lemma 18. *([CK64]) For every k there exists an $f(k)$ such that for every k-uniform intersecting family K there is a kernel S of cardinality at most $f(k)$.*

The point of this lemma that will be exploited is that the size of this kernel $f(k)$ is independent of n or the number of vertices of the hypergraph. This was a well-studied topic and the interested reader may read [AF87, EM72, EL73, FF86, Fra78, Tuz85]. Now we will repeat the result from [CL10] that shows how to apply this lemma to prove the following theorem, with our improved bound on the integrality gap.

Theorem 19. *Let $\varepsilon > 0$. There is a polynomially sized LP for k-set packing with integrality gap at most $\frac{k}{3} + 1 + \varepsilon$.* ***[NOTE: see page iii]***

Proof. We will prove it is possible to find a polynomially sized LP that still captures all the constraints of the intersecting family LP. This immediately implies the LP has the claimed integrality gap by Theorem 17. We follow the proof from [CL10].

Let $G = (V, E)$ be the hypergraph and create a variable x_U for every subset $U \subseteq V$ that is a subset of some hyperedge $e \in E$. To enforce that x_U represents the fractional value of U, add the constraint $x_U = \sum_{e \supseteq U} x_e$. Each new variable U is a subset of some hyperedges in G, and U is said to be contained in a subset S if $U \subseteq S$.

Now enumerate all possible subsets $S \subseteq V$ with $|S| \leq f(k)$. For each such subset S, enumerate all possible intersecting families K_S formed by the new variables contained in S. For each such intersecting family K_S write the following kernel constraint.

$$\sum_{U \in K_S} x_U \leq 1.$$

There are $\sum_{i=1}^{f(k)} \binom{n}{i} \leq n^{f(k)+1}$ possible kernels. For each kernel S with $|S| = t$ there are at most 2^t new variables contained in S, because there are at most 2^t subsets of S. Hence there are at most 2^{2^t} intersecting families K_S induced in S, because there are at most 2^{2^t} hypergraphs in S. Every such intersecting family corresponds to one constraint, so there are no more than $n^{f(k)+1} 2^{2^{f(k)}}$ kernel constraints. So when k is a constant this is a number polynomial in n.

By Lemma 18 each intersecting family has a kernel constraint and hence each intersecting family has at most a fractional value of 1. So indeed, the intersecting family LP can be rewritten into a polynomially sized LP for every constant k. □

Chapter 5

SDP formulation

The first two sections treat background on semidefinite programming and the Lovász Theta function. These concepts are needed for the proof in Section 5.3 of the theorem that there exists a polynomially sized SDP with the improved integrality gap from Theorem 17.

5.1 Background on semidefinite programming

A semidefinite program is a more general form of a linear program. General references for a thorough background about SDPs are [BTN01, Fre04, Lov95, Pis06, VB94].

In a linear program, the objective is to maximise a linear function over a convex polyhedron. In a semidefinite program the numbers are substituted by vectors and the dot product of two vectors is used instead of the multiplication of two numbers. A semidefinite program can be written in the following form.

$$\begin{aligned} \max \quad & c^T x \\ \text{s.t.} \quad & x_1 A_1 + \ldots + x_n A_n - B \succeq 0 \end{aligned}$$

Here $x \in \mathbb{R}^n$ is the vector of decision variables one needs to assign values to in order to maximise the inner product $c^T x$ with the given vector $c \in \mathbb{R}^n$. $A_1, \ldots, A_n, B \in \mathbf{S}^m$ are given symmetric $m \times m$ matrices, so one can think of $X := x_1 A_1 + \ldots + x_n A_n - B$ as a matrix whose entries are linear functions over the variables x_i. The constraint $X \succeq 0$ means X needs to be positive semidefinite, which is equivalent to $y^T X y$ being nonnegative for all $y \in \mathbb{R}^m$ or to X having only nonnegative eigenvalues. When $X \succeq 0$ for some $x \in \mathbb{R}^n$ we say x is a feasible solution. Since both the objective function and the constraints are convex in x, a semidefinite program is a convex optimization problem. In contrast to a linear program, its feasible region is in general not a a polyhedron.

5.2 Background on the Lovász Theta function

This section gives some background on the famous Lovász Theta function introduced in [Lov79]. We introduce the Lovász Theta function via orthogonal representations. In order to do that some background about the stable set polytope is first given in Subsection 5.2.1. Subsection 5.2.2 will talk about orthogonal representations and then the Lovász Theta function is introduced in Subsection 5.2.3. General references for this section are [BTN01, GLS88, Knu94, Lov95].

5.2.1 The stable set polytope

Definition Stable set is another name for an independent set. Given a graph $G = (V, E)$, $\alpha(G)$ denotes the size of the maximum independent set in G. For every subset of the vertices $S \subseteq V$ its incidence vector is denoted by $\chi^S \in \mathbb{R}^V$, i.e. for all $i \in V$, $\chi_i^S = 1$ if $i \in S$ and $\chi_i^S = 0$ otherwise. Now define the stable set polytope STAB(G) as follows.

$$\text{STAB}(G) = \text{conv.hull}(\chi^S \in \mathbb{R}^V \mid S \text{ is an independent set in } G)$$

Properties So STAB(G) is the smallest convex set in $\mathbb{R}^V$ containing the incidence vectors of all independent sets. Since all extreme points of this polytope are $0, 1$-vectors, there is a system of linear inequalities describing this convex set. Theoretically it is possible to find $\alpha(G)$ by optimising the linear objective function $\sum_i x_i$ over STAB(G). However, the number of constraints is generally exponential in $|V|$ so this is not an efficient approach to find $\alpha(G)$, which should be expected as determining $\alpha(G)$ is NP-hard. What one can do, however, is to find extra inequalities for the stable set polytope and find upper bounds for $\alpha(G)$.

The clique constrained stable set polytope With the intersecting family LP in mind it is natural to start with the following inequalities for $x \in \mathbb{R}^V$.

$$x_i \geq 0 \quad i \in V, \tag{5.1}$$

$$\sum_{i \in V(Q)} x_i \leq 1 \quad \text{for all cliques } Q \text{ in } G. \tag{5.2}$$

Now define the clique constrained stable set polytope as follows.

$$\text{QSTAB}(G) = \text{conv.hull}(x \in \mathbb{R}^V \mid \text{Constraints (5.1) and (5.2) hold}).$$

Now any independent set in G corresponds to an integral vertex in QSTAB(G) and vice versa. The clique constrained stable set polytope is the first approximation of the stable set polytope that is considered here, but to formally define the Lovász Theta function another polytope is introduced in the next subsection.

5.2.2 Orthogonal representations

Definition This subsection introduces the Lovász Theta function via orthogonal representations. Let $G = (V, E)$ be a graph and let $\overline{E} = \{\{i, j\} \in V \times V \mid \{i, j\} \notin E\}$ be the complement of E. Formally, an orthogonal representation of G is a mapping (labeling) $u : V \to \mathbb{R}^d$ for some d such that $u_i^T u_j = 0$ for all $\{i, j\} \in \overline{E}$. In other words we need to assign a vector u_v to every vertex v such that the vectors of any two non-adjacent vertices are perpendicular to each other. Such a mapping always exists, in fact, there are two trivial mappings: map all vertices to 0, or map the vectors to a set of mutually orthogonal vectors in $\mathbb{R}^V$.

Orthogonal constrained stable set polytope An orthogonal representation $(u_i \mid i \in V)$ with $u_i \in \mathbb{R}^d$ is called orthonormal when every vector has unit length, i.e. when $\|u_i\| = 1$ for all $i \in V$. Let c be some vector in $\mathbb{R}^d$ with $\|c\| = 1$ (for example, take $c = e_1$). Then for any stable set $S \subseteq V$ its vectors $\{u_i \mid i \in S\}$ are mutually orthonormal as the vertices are non-adjacent, and hence

$$\sum_{i \in S} (c^T u_i)^2 \leq 1.$$

This is true because the left-hand side is the squared length projection of c onto the subspace spanned by the u_i. The length of this projection is at most the length of c which is 1. In fact, note that $\sum_{i \in V} (c^T u_i)^2 \chi^S = \sum_{i \in S} (c^T u_i)^2$, which yields that the following inequality holds for the incidence vector χ^S of any stable set $S \subseteq V$. It is called the orthogonality constraint.

$$\sum_{i \in V} (c^T u_i)^2 x_i \leq 1. \tag{5.3}$$

Similar like before, we can now define the orthogonal constrained stable set polytope as follows.

$$\text{TSTAB}(G) = \text{conv.hull}(x \in \mathbb{R}^V \mid \text{Constraints (5.1) and (5.3) hold}).$$

5.2.3 The Lovász Theta function

The most interesting property of TSTAB(G) is the fact that one can optimise linear functions over it in polynomial time [GLS88, Theorem 9.3.30]. We can now succinctly define the Lovász Theta function.

$$\vartheta(G) = \max\left\{\sum_i x_i \mid x \in \mathrm{TSTAB}(G)\right\}.$$

The following is equivalent by writing out the definitions. Let ONR denote an orthonormal representation. Denote the following LP by (ϑ-LP).

$$\begin{aligned} \vartheta(G) = \max \quad & \sum_i x_i \\ \text{s.t.} \quad & \sum_{i\in V}(c^T u_i)^2 x_i \leq 1 \quad \forall c : \|c\| = 1 \quad \forall \mathrm{ONR}\{u_i\} \\ & x_i \geq 0 \qquad\qquad\qquad \forall i \in V \end{aligned} \tag{ϑ-LP}$$

There are a lot of alternative and equivalent definitions for the Lovász Theta function. The semidefinite program for k-set packing uses the following equivalent definition, known in the literature as $\theta_3(G)$. An orthogonal representation $\{b_i\}$ is called normalised if $\sum_i \|b_i\|^2 = 1$, and define $\overline{G} = (V, \overline{E})$.

$$\vartheta_3(G) = \max\left\{\sum_{u,v} b_u b_v \mid b \text{ is a normalised orthogonal representation of } \overline{G}\right\}.$$

5.3 An SDP for k-set packing

This section contains the main theorem of this chapter. As before, view k-set packing as the independent set problem in a $k+1$-claw free graph and consider the following clique LP.

$$\begin{aligned} \max \quad & \sum_i x_i \\ \text{s.t.} \quad & x \in \mathrm{QSTAB}(G) \end{aligned} \tag{Clique LP}$$

This is optimising the size of an independent set over the clique constrained stable set polytope.

Lemma 20. (Clique LP) *is equivalent to the intersecting family LP of Chapter 4.*

Proof. The nonnegativity constraints $x_i \geq 0$ for (Clique LP) obviously match the same constraints in the intersecting family LP. The clique constraints $\sum_{i\in Q} x_i \leq 1$ for cliques

of size 1 imply the bound $x_i \leq 1$. Evidently they also imply the intersecting family constraints $x(K) \leq 1$ for $Q = K$. The constraints that $x(\delta(v)) \leq 1$ are also implied by the clique constraints: all hyperedges covering element v form a clique in the conflict graph. The other way around is similar. □

Note that replacing QSTAB(G) by TSTAB(G) yields $\vartheta(G)$. These are related as follows.

Lemma 21. *([CL10, Lemma 4.3]) Any feasible solution to* (ϑ-LP) *is a feasible solution to* (Clique LP).

Proof. It suffices to show that the orthogonality constraints (5.3) imply the clique constraints (5.2). Let Q be a clique in G and map all vertices of Q to c and all other vertices to mutually orthogonal vectors that are also orthogonal to c. Then the orthogonality constraint for Q implies its clique constraint. □

In other words, for every graph G

$$\mathrm{STAB}(G) \subseteq \mathrm{TSTAB}(G) \subseteq \mathrm{QSTAB}(G).$$

Finally all linear and semidefinite programs can be linked.

Lemma 22. *$\vartheta_3(G)$ is a stronger relaxation than the intersecting family LP.*

Proof. By Lemma 21 (ϑ-LP) is a stronger relaxation than the clique LP. Hence by Lemma 20 (ϑ-LP) is also stronger than the intersecting family LP. (ϑ-LP) is equivalent to $\vartheta(G)$, which is equivalent to $\vartheta_3(G)$. Hence $\vartheta_3(G)$ is a stronger relaxation than the intersecting family LP. □

$\vartheta_3(G)$ can be written as follows.

$$\begin{aligned} \vartheta_3(G) = \max \quad & \sum_{i,j \in V} u_i u_j \\ \text{s.t.} \quad & u_i u_j = 0, \qquad \forall (i,j) \in E \\ & \sum_{i=1}^{n} u_i^2 = 1 \\ & u_i \in \mathbb{R}^d, \qquad \forall i \in V \end{aligned} \tag{ϑ_3-LP}$$

This is a semidefinite program which is a stronger relaxation than the intersecting family LP. Then by Lemma 22 and Theorem 17 the following is true.

Theorem 23. *Let $\varepsilon > 0$. (ϑ_3-LP) is an SDP relaxation for k-set packing with integrality gap at most $\frac{k}{3} + 1 + \varepsilon$.* ***[NOTE: see page iii]***

This is Theorem 1.5 from [CL10] with our improved bound on the integrality gap. Since this semidefinite program has a polynomial size, the following theorem is also true, similar to Theorem 19.

Theorem 24. *Let $\varepsilon > 0$. There is a polynomially sized SDP for k-set packing with integrality gap at most $\frac{k}{3} + 1 + \varepsilon$.* ***[NOTE: see page iii]***

Chapter 6

Weighted approximation

In this chapter we give a simplified proof of the main lemma of Berman's paper [Ber00] containing the currently best approximation algorithm for weighted k-set packing with approximation guarantee $\frac{k+1}{2}$. First the necessary terminology is introduced in Section 6.1. Then the algorithm is discussed in Section 6.2 and Section 6.3 provides a new proof of the main lemma.

6.1 Terminology

Claws Consider the following setting from [Ber00] for this chapter. For a graph, define a k-claw C as a subgraph isomorphic to $K_{1,k}$, the complete bipartite graph on 1 and k vertices. For convenience, define a 1-claw to be a singleton set C with $T_C = C$. A claw is a k-claw for some k. Define the single vertex connected to all other vertices of the claw to be the center Z_C. The other vertices of the claw (forming an independent set by definition) are called the talons T_C of the claw. A k-claw has k talons and one center vertex. Write $C = Z_C \cup T_C$ for a claw C with center vertex Z_C and talons T_C.

Approximation guarantee Let $G = (V, E)$ be a k-claw free graph with a weight $w(v)$ for every vertex $v \in V$. The main theorem of [Ber00] is a $\frac{k}{2}$-approximation algorithm for maximum independent set in such a graph. This yields a $\frac{k+1}{2}$-approximation algorithm for weighted k-set packing because any packing corresponds to an independent set in a $k+1$-claw free graph. The algorithm searches for claws satisfying certain properties and then adds the talons of the claw to the current independent set A and removes the neighbours of the talons in A.

Notation We will use the following notational conventions. Define for a vertex $v \in V$ its open neighbourhood (or just its neighbourhood) $N(v) = \{w \in V \mid \{v, w\} \in E\}$ and its closed neighbourhood $N[v] = N(v) \cup \{v\}$. For a subset of the vertices $W \subseteq V$ define its closed neighbourhood $N[W] = \bigcup_{w \in W} N[w]$ and write $N(W) = N[W] \setminus W$ for the (open) neighbourhood of W.

Definition 25. *For two given subsets of the vertices U and A write $N(U, A) = N(U) \cap A$, i.e. the neighbourhood of U in A.*

$N(U, A)$ is called the A-neighbourhood of U and we refer to the vertices in $N(U, A)$ as the A-neighbours of U. Write $N(u, A)$ for $N(\{u\}, A)$ for some vertex u and some subset of the vertices A.

Definition 26. *By $n(u, A)$, denote a vertex $v \in N(u, A)$ with maximum weight.*

If this is not uniquely defined, simply choose a random vertex of the different possibilities.

For convenience we introduce the notation $w(U) = \sum_{v \in U} w(v)$ for some subset of the vertices $U \subseteq V$. Even shorter, we write the following.

Definition 27. *$w(U, A) = w(N(U, A))$ and $w(u, A) = w(N(u, A))$.*

These are the sums of the weights of the A-neighbours of a subset of the vertices U respectively one vertex u. These notations extend to different weight functions such as w^2, in particular note that $w^2(U, A) = \sum_{v \in N(U,A)} w^2(v) \neq (w(U, A))^2$. Now define the following function as in [Ber00].

$$charge(u, v) = \begin{cases} w(u) - \frac{1}{2}w(u, A), & \text{if } v = n(u, A); \\ 0, & \text{otherwise.} \end{cases}$$

Now define the following.

Definition 28. *Let A be an independent set in a graph $G = (V, E)$. Define a good claw $C = Z_C \cup T_C$ to be a claw satisfying either of the following properties.*

1. *$N(T_C, A) = \emptyset$, i.e. adding T_C to A to obtain another independent set does not require the removal of any sets in A; or*

2. *The center vertex Z_c is in A and $\sum_{u \in T_C} charge(u, v) > \frac{1}{2}w(v)$.*

A claw C is called a nice claw if it is a minimal set forming a good claw, i.e. if there is no strict subset of C forming a smaller good claw.

6.2 Two algorithms joining forces

6.2.1 The algorithms SquareImp and WishfulThinking

In this setting, define the following algorithm:

SquareImp
$A \leftarrow \emptyset$
While there exists a claw C such that T_C improves $w^2(A)$
  $A \leftarrow A \cup T_C \setminus N(T_C, A)$

Now define the following algorithm:

WishfulThinking
$A \leftarrow \emptyset$
While there exists a nice claw C
  $A \leftarrow A \cup T_C \setminus N(T_C, A)$

These two algorithms can now be linked in the following way.

1. Every nice claw improves $w^2(A)$, so a run of WishfulThinking forms the initial part of a run of SquareImp. See Section 6.3.
2. Consequently, WishfulThinking cannot make more iterations than SquareImp.
3. When SquareImp terminates it yields an independent set A for which no claw improves $w^2(A)$. Hence there is no more nice claw, so WishfulThinking terminates.
4. If WishfulThinking terminates, its approximation guarantee is $\frac{k}{2}$. See Subsection 6.2.2.

The proof of the approximation guarantee is repeated in Subsection 6.2.2 to get some insight into the non-intuitive definitions of the charge function and good claws. The main lemma is the fact that every nice claw improves $w^2(A)$, for which we give a simplified proof in Section 6.3.

6.2.2 The approximation guarantee

We repeat the following proof of [Ber00].

Lemma 29. *([Ber00, Lemma 1]) Assume that* WishfulThinking *has terminated and that A^* is an independent set. Then* $\dfrac{w(A^*)}{w(A)} \leq \dfrac{k}{2}$.

Proof. Let $G = (V, E)$ be the graph and let A be the independent set that has been found using WishfulThinking. Let A^* be any independent set in G (in particular it could be the maximum independent set). We will distribute $w(A^*)$ among the vertices of A such that no vertex $v \in A$ receives more than $\frac{k}{2}w(v)$. This immediately implies the claimed result. The distribution consists of two phases.

In the first phase, every vertex $u \in A^*$ sends to each of its A-neighbours $v \in N(u, A)$ a portion of its weight equal to $\frac{1}{2}w(v)$. Note that $N(u, A) \neq \emptyset$ because otherwise $\{u\}$ would be a nice 1-claw and these do not exist when WishfulThinking has terminated.

In this first phase, every vertex u sends a portion of its weight equal to $\frac{1}{2}w(u, A)$. By the definition of the *charge* function, the portion of its weight that is not distributed yet equals $charge(u, n(u, A))$. In the second phase u sends $charge(u, n(u, A))$ to $n(u, A)$.

Now consider some vertex $v \in A$ in the receiving side of this distribution. In the first phase v gets $\frac{1}{2}w(v)$ from all its neighbours in A^*. Because A^* is an independent set and the graph is k-claw free, v has at most $k - 1$ neighbours in A^*. Thus v gets at most $(k-1)\frac{1}{2}w(v) = \frac{k}{2}w(v) - \frac{1}{2}w(v)$ in the first phase.

In the second phase, v receives exactly $\sum_{u \in N(v,A^*)} charge(u, v)$. By the definition of a good claw, this can be at most $\frac{1}{2}w(v)$: otherwise the vertices in A^* sending positive *charge*s to v form the talons of a good claw with $v \in A$ at its center.

Hence every vertex $v \in A$ receives at most $\frac{k}{2}w(v) - \frac{1}{2}w(v) + \frac{1}{2}w(v) = \frac{k}{2}w(v)$, and hence the weight of A^* is at most $\frac{k}{2}$ times as much as the weight of A. □

As noted, this yields a $\frac{k}{2}$-approximation for the weighted independent set problem in k-claw free graphs, which results in a $\frac{k+1}{2}$-approximation for weighted k-set packing.

Instructive example Berman [Ber00] proceeds with an example of an instance where an iteration of WishfulThinking in fact decreases $w(A)$, which is the function it is in fact trying to maximise. This is contra-intuitive: we are using a local search technique and from a given solution we might move to a next solution with a worse objective value. However, as we will prove in Section 6.3, an iteration of WishfulThinking always increases $w^2(A)$ and that suffices for the analysis.

Here is the example, see Figure 6.1. Let S be a subset of the current independent set A depicted at the bottom of Figure 6.1 and let T be a subset of the vertices in $V \setminus A$ depicted at the top. Write $S = \{s_1, \ldots, s_5\}$ with $w(s_i) = 10$ for $i = 1, \ldots, 5$, and $T = \{t_1, t_2\}$ with $w(t_1) = w(t_2) = 18$. Let $n(t_i, A) = s_3$ for $i = 1, 2$.

We claim that $\{s_3\} \cup \{t_1, t_2\}$ is a nice claw. To see this, note that $charge(t_i, s_3) = w(t_i) - \frac{1}{2}w(t_i, A) = 18 - \frac{1}{2}(10+10+10) = 3$. So for s_3 we have $\sum_{t_i} charge(t_i, s_3) = 3+3 = 6$,

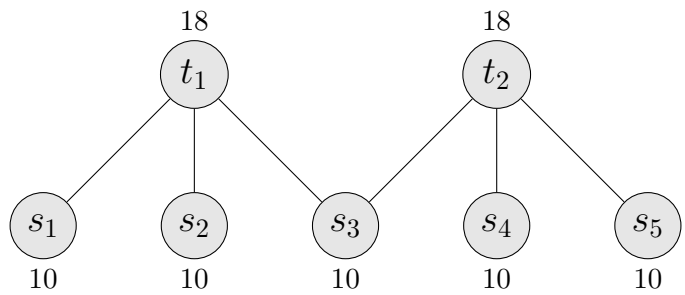

Figure 6.1: An example where an iteration of WishfulThinking actually decreases $w(A)$. The bottom vertices are in $S \subseteq A$ and the top vertices are in $T \subseteq V \setminus A$.

which is larger than $\frac{1}{2}w(s_3) = 5$. So all conditions are satisfied and $\{s_3\} \cup \{t_1, t_2\}$ is a good claw, and because it is minimal it is also a nice claw.

However, adding T and removing $N(T, S) = S$ means adding two sets of weight 18 and removing 5 sets of weight 10. Hence $w(A)$ decreases by 14. But the squared weight function increases: the gain is twice 18^2 and the loss is five times 10^2, so it increases by $648 - 500 = 148$. In fact, elementary calculus shows that w^c increases in this example for $c > \frac{\log 5 - \log 2}{\log 18 - \log 10} \approx 1.56$, or in a more general setting, for $c > \frac{\log |S| - \log |T|}{\log w(s_i) - \log w(t_i)}$. See also [BK03] for results on using the misdirected weight function w^c for some $c \neq 1$.

6.2.3 A new observation

For the independent set problem in k-claw free graphs the use of w^2 rather than w can be advantageous due to the following observation. Let A be some independent set and let u be some vertex not in A.

$$\sum_{t \in N(u,A)} w^2(t) \leq \sum_{t \in N(u,A)} \left(w(t) \left(\max_{t \in N(u,A)} w(t) \right) \right) = \left(\max_{t \in N(u,A)} w(t) \right) \sum_{t \in N(u,A)} w(t).$$

This proofs the following observation.

Observation 30. $w^2(u, A) \leq n(u, A) w(u, A)$.

So the squared weight function of a set of vertices is capable of capturing information not only about the sum of the weights but also about the maximum weight. Using this simple observation Berman's proof can be simplified in the next section.

6.3 Simplified proof

Here is a simplified proof of the main lemma from [Ber00] proving that adding a nice claw to the current independent set A improves $w^2(A)$. We believe this proof gives some more insight into what is really happening behind the math thanks to Observation 30.

Lemma 31. *If C is a nice claw, then T_C improves $w^2(A)$.*

Proof. Let A be the current independent set and let $C = Z_C \cup T_C = \{v\} \cup T$ be a nice claw. We need to show that the weight of what is added (T) is more than the weight of what is lost ($N(T, A)$), so we need to show that

$$w^2(T) > w^2(T, A), \tag{6.1}$$

or equivalently,

$$w^2(T) - w^2(T, A - \{v\}) > w^2(v). \tag{6.2}$$

To proof that for a nice claw (6.2) holds, we proof the following claim.

Claim 1. *Let $C = \{v\} \cup T$ be a nice claw. Then*

$$w^2(T, A - \{v\}) \leq \sum_{u \in T} \left(w^2(u, A) - w^2(v) \right).$$

Proof. By definition, $w^2(T, A - \{v\})$ is the sum of the squared weights of all neighbours of the talons T in A excluding v. Summing over T rather than the neighbourhood of T, this can be bounded by $\sum_{u \in T} w^2(u, A - \{v\})$. This is an upper bound, because in this expression vertices that are neighbours of more than one vertex in T are counted multiple times. Therefore, $w^2(T, A - \{v\}) \leq \sum_{u \in T} w^2(u, A - \{v\})$. Noting that $w^2(u, A - \{v\})$ equals $w^2(u, A) - w^2(v)$, the claim follows. □

For the first term in (6.2), write $w^2(T) = \sum_{u \in T} w^2(u)$. Now by Claim 1, the following implies (6.2).

$$\sum_{u \in T} \left(w^2(u) - w^2(u, A) + w^2(v) \right) > w^2(v). \tag{6.3}$$

To show that (6.3) holds when C is a nice claw, we proceed to the second claim.

Claim 2. *Let $C = \{v\} \cup T$ be a nice claw. Then $v = n(u, A)$ for all u in T and*

$$w(v) < 2 \sum_{u \in T} charge(u, v). \tag{6.4}$$

Proof. Equation (6.4) just follows from the definition stating $\sum_{u \in T} charge(u, v) > \frac{1}{2}w(v)$. Also, as C is a nice claw, it is minimal, implying that every term on the right-hand side of (6.4) is positive. By the definition of *charge*, this is true only if v is the maximum weight neighbour of u within A. □

So proving that (6.3) holds when C is a nice claw has now been reduced by Claim 2 to showing that whenever $charge(u, v) > 0$ we have

$$w^2(u) - w^2(u, A) + w^2(v) \geq 2w(v)charge(u, v).$$

Here we plugged in (6.4) only once in the right-hand side. By the definition of *charge*, this boils down to proving that

$$w^2(u) - w^2(u, A) + w^2(v) \geq 2w(u)w(v) - w(v)w(u, A) \tag{6.5}$$

holds whenever v is the maximum weight neighbour of u.

From this point on we will deviate from the proof of Berman [Ber00]. He now scales the quantities, makes a case distinction and rewrites the equations algebraically until it is clear they are indeed true. However, (6.5) can be shown more easily using Observation 30. Plugging this into (6.5) yields

$$w^2(u) - w(v)w(u, A) + w^2(v) \geq 2w(u)w(v) - w(v)w(u, A).$$

The terms $w(v)w(u, A)$ now cancel and what is left is

$$w^2(u) + w^2(v) \geq 2w(u)w(v), \tag{6.6}$$

which is obviously true as this is equivalent to

$$(w(u) - w(v))^2 \geq 0.$$

We have now proved that when C is a nice claw, (6.1) holds, and thus T_C improves $w^2(A)$. □

Chapter 7

Discussion

In this chapter we will discuss possible directions for further research. The LP and the SDP relaxations for the problem are discussed in Section 7.1. Section 7.2 considers possible ways in which the weighted approximation algorithm of Berman [Ber00] could be lightly changed. Section 7.3 continues with a discussion about the difference in the results on the independent set problem in bounded degree graphs and the current results on k-set packing. Finally Section 7.4 discusses the problems arising when one tries to generalise the unweighted approximation algorithms with weights. Sections 7.1, 7.3 and 7.4 suggest future research directions and contain some conjectures.

7.1 LP and SDP relaxations

Here is a summary of the current results.

1. The standard LP relaxation for k-set packing has integrality gap $k-1+\frac{1}{k}$. In the case of k-dimensional matching (i.e. when the hypergraph is k-partite) the integrality gap equals $k-1$. Chan and Lau [CL10] gave algorithms for these cases. These results and algorithms also extend to the weighted case.

2. The intersecting family LP for k-set packing has integrality gap at most $\frac{k}{3}+1+\varepsilon$ (Theorem 17). It is not known whether this result also extends to the weighted case.

3. By the results of [CL10], there also exists a polynomially sized LP for k-set packing with integrality gap at most $\frac{k}{3}+1+\varepsilon$. We don't know whether this is also true for the weighted version either.

4. Also by [CL10], the Lovász Theta function is at least as strong as the intersecting family and therefore this SDP relaxation has integrality gap at most $\frac{k}{3}+1+\varepsilon$.

7.1.1 Extending results to the weighted case

Unweighted versus weighted While the results for the standard LP extend to the weighted case, the results for the intersecting family LP do not extend in an obvious way. This is for the same reason why the unweighted approximation algorithms for k-set packing do not easily generalise to the weighted case (c.f. Section 7.4): the local search technique is relying crucially on cardinality. For example, $\mathcal{F}_1(e)$ in the proof of Theorem 17 need not be an intersecting family in the weighted case: the objective function might increase even by adding less sets than are removed. We elaborate on this in Section 7.4.

Possible research directions Perhaps there is another way to partition the hyperedges rather than in $\mathcal{F}_1$, $\mathcal{F}_2$ and $\mathcal{F}_{3+}$ that does provide a way to extend the result to the weighted case. For the weighted problem the setting of $\mathcal{F}_1$, $\mathcal{F}_2$ and $\mathcal{F}_{3+}$ does not make sense and nothing can be proved. In the weighted case it seems to make sense to define a t-locally optimal solution as a solution where adding at most t new sets and losing any number of sets does not yield an improvement (instead of losing less than t sets in the unweighted case). But perhaps in the weighted case it will prove to be worthwhile to consider the integral optimum solution rather than a t-locally optimum.

The proof of the existence of a polynomially sized LP in Theorem 19 depends on the existence of small kernels for every intersecting family. If a new result on the integrality gap of some LP for weighted k-set packing relies on intersecting families, this result still holds. The SDP relaxation still captures all intersecting family constraints in the weighted case (one can just add weights to the objective function and nonnegativity constraints on them), so if one proves a result for weighted k-set packing using these intersecting families, that result immediately extends to the polynomially sized LP and SDP relaxation.

Problem 1. *Narrow the gap between the integrality gap for relaxations of the weighted and the unweighted k-set packing problem.*

7.1.2 Smaller bound on integrality gap

Another way to improve upon the current results is to further decrease the upper bound on the integrality gap on LP relaxations like the intersecting family. This seems likely to be possible, because all unweighted approximation algorithms achieve bounds of $\frac{k+2}{3} + \varepsilon$ or $\frac{k+1}{3} + \varepsilon$ while the current result is $\frac{k}{3} + 1 + \varepsilon$. Perhaps some ideas from these algorithms can be extended to the integrality gap of the LP relaxation.

Problem 2. *Improve the integrality gap of relaxations for the unweighted k-set packing to $\frac{k+2}{3} + \varepsilon$ or better.*

Towards a gap of $\frac{k+1}{3} + \varepsilon$ In particular we would like to point out that it may be worthwhile to see if the idea from the quasi-polynomial time $\frac{k+1}{3} + \varepsilon$ from [CGM13] can be extended. The proof of this approximation guarantee also depends on Lemma 15 as did our new bound of $\frac{k}{3} + 1 + \varepsilon$. Their algorithm uses slightly more crafted ideas, but these do not seem to extend to the integrality gap of the LP in a straightforward way. Roughly speaking they are able to bound $\mathcal{F}_1$ by $\varepsilon|M|$ rather than $|M|$. This is then added twice like in the proof of Theorem 17. This effectively decreases the bound by $\frac{2}{3}|M|$ which is exactly the current difference between the results. Perhaps by altering the argument a little bit, one could use the idea of this proof to improve the bound for the integrality gap for the intersecting family LP to, say, $\frac{k+1}{3} + \varepsilon$.

Towards a gap of $\frac{k+2}{3} + \varepsilon$ Also the idea from the $\frac{k+2}{3}$-approximation by Sviridenko and Ward [SW13] might be interesting to take a closer look at. Very roughly speaking they bound $\mathcal{F}_2$ by $2|M \setminus \mathcal{F}_1|$. This cancels against the bound for $\mathcal{F}_1$ and hence they obtain $3|\mathcal{F}| \leq k|M| + 2|M|$. A similar idea could perhaps be used to bound the LP-values of the sets $\mathcal{F}_1$ and $\mathcal{F}_2$.

Since the local search techniques have reached their limits in terms of their approximation guarantee, other techniques have to be sought to improve the approximation guarantee. The results on the LP and SDP relaxations for k-set packing are scarce and we believe more research in this area could turn out to be fruitful. We make the following conjecture.

Conjecture 3. *The integrality gap of relaxations for the unweighted k-set packing can be bounded by $\frac{k+1}{3} + \varepsilon$.*

7.2 Improving Berman's weighted approximation

7.2.1 Generalising *charge* and claws

The analysis as presented in Section 6.3 shows that slight changes to the algorithm do not simply constitute an improvement of the approximation guarantee to $\frac{k}{c}$ for some $c > 2$. Let us replace the $\frac{1}{2}$ in the definition of $charge(u, v)$ to $\frac{1}{c}$ and the $\frac{1}{2}$ in the definition of a nice (good) claw to $\frac{1}{c}$.

Approximation guarantee Lemma 29 can now be easily adapted to hold for any $c > 2$. The proof remains exactly the same and WishfulThinking now has an approximation guarantee of $\frac{k}{c}$.

Nice claw improves $w^2(A)$ However, Lemma 31 is no longer true and this can be seen from our simplified proof. In the proof that the talons of a nice claw improve $w^2(A)$ when the definitions of $charge(u, v)$ and of a nice claw have been altered, the analysis remains exactly the same except for the fact that in Equation (6.6), the factor 2 in the right-hand side changes into a c:

$$w^2(u) + w^2(v) \geq cw(u)w(v), \tag{7.1}$$

However, Equation (7.1) is only true for all u and v for the values $c = 0$ and $c = 2$. Since $c = 0$ is not possible because c occurs in the denominator in the definition of $charge(u, v)$ and of a nice claw, $w^2(A)$ improves only when $c = 2$ when following the current analysis.

It is possible to incorporate an extra constraint in the definition of a nice claw to make sure that Equation (7.1) holds for some $c > 2$. Then Lemma 31 is true because of the altered definition. However, the extra constraint now causes trouble in the proof of Lemma 29, which is then not necessarily true anymore.

7.2.2 Generalising the weight function

Intuitively, what is the crucial point why w^2 might behave differently than w? In general, looking at the squared weight function (or at w^c for any $c > 1$) is slightly more biased towards larger weights. When a vertex has some weight m and we add 1 to its weight, w increases by 1 while w^2 increases by $2m + 1$. Also the square of the weight of one vertex might be more than the weight of two vertices. w^2 prefers one vertex of weight 3 to two vertices of weight 2, while w prefers the two vertices of weight 2 to the single vertex of weight 3. So by guiding the search by w^2 rather than w, an iteration might decrease the real objective function but it is more difficult to get stuck in an inferior locally optimal solution and hence a better result might be achieved in the end.

One could also try to improve w^p for some $2 < p \in \mathbb{N}$ next to the use of the parameter c rather than 2. Following the same analysis as in Section 6.3, Equation (6.6) then translates to

$$w^p(u) + w^p(v) \geq cw(u)w^{p-1}(v).$$

This is trivially true for $c = 0$ and even p, for $p = c = 1$ and for $p = c = 2$. Again the

first is not a real option, the second option gives an approximation guarantee of $\frac{k}{1}$ which is not an improvement and the third option is the one that leads to Berman's result. This shows that using the current analysis this is the best possible result, so to improve the approximation guarantee one really needs another analysis or another approach. See also [BK03] for other reasoning about generalising Berman's algorithm.

7.3 Relation to the independent set problem

Here is a summary of the relation between k-set packing and the independent set problem.

1. Finding a maximum set packing is equivalent to finding a maximum independent set in the conflict graph of the set packing instance.

2. Finding a maximum k-set packing can be reduced to finding a maximum independent set in the $k+1$-claw free conflict graph of the k-set packing instance.

7.3.1 Results on the independent set problem

General graphs The following results are known for the maximum independent set problem in general. Let n be the number of vertices in the graph.

1. The problem is NP-hard and it is hard to approximate within $n^{1-\varepsilon}$ unless NP-hard problems have randomised polynomial time algorithms [Hås96].

2. There is an approximation algorithm that achieves an approximation guarantee of $\Theta\left(\frac{n}{\log^2 n}\right)$ [BH92].

Bounded degree graphs When the maximum degree of every vertex is assumed to be bounded by some Δ, approximating the problem becomes considerably easier. The following results are known for the maximum independent set problem on bounded degree graphs.

1. Assuming the Unique Games Conjecture [Kho02] it is hard to approximate within $O\left(\frac{\Delta}{\log^2 \Delta}\right)$ [AKS09].

2. The greedy algorithm is an obvious Δ-approximation. Hochbaum [Hoc83] was the first to give an approximation algorithm with approximation guarantee $\frac{\Delta}{2}$, which was improved to $\frac{\Delta+2}{3}$ [HR94a, HL97]. Berman and Fürer [BF94] give a $\frac{\Delta+3}{5}+\varepsilon$-approximation for even Δ and a $\frac{\Delta+3.25}{5}+\varepsilon$-approximation for odd Δ, which

was slightly improved in [BF95]. Then a major jump came with an $O\left(\frac{\Delta}{\log\log\Delta}\right)$-approximation, a $\frac{\Delta}{6}(1+o(1))$-approximation, a proof that greedy achieves $\frac{\Delta+2}{3}$ and that Berman and Fürer really achieved $\frac{\Delta+3}{4}$ [HR94b]. The currently best result is an $O\left(\frac{\Delta\log\log\Delta}{\log\Delta}\right)$-approximation in polynomial time that also extends to the weighted case [Hal02, Hal99, Vis96].

7.3.2 From bounded degree to claw-free graphs

Comparison The results on the maximum independent set problem in bounded degree graphs are much stronger than the results on k-set packing. For example, the gap between its hardness ($\Omega\left(\frac{\Delta}{\log^2\Delta}\right)$) and its best approximation guarantee ($O\left(\frac{\Delta\log\log\Delta}{\log\Delta}\right)$) is much smaller ($\Omega(\frac{k}{\log k})$ versus $\frac{k+1}{3}+\varepsilon$). The constraint that the degree is bounded apparently allows much more arguments than the constraint that the size of an independent set in the neighbourhood of every vertex is bounded.

Problem 4. *Narrow the gap between the approximation guarantee of k-set packing and the independent set problem on bounded degree graphs.*

Mimicking the bounded degree algorithm As an example, consider the $O\left(\frac{\Delta\log\log\Delta}{\log\Delta}\right)$-approximation for weighted independent set in bounded degree graphs by Halperin [Hal02, Section 5]. In a nutshell this solves a semidefinite programming relaxation and partitions the resulting vectors in sets S_0, S_1 and S_2. It then uses the greedy approach on S_0 to find an independent set I_0, projects and normalises the vectors in S_1 and selects some of them to find I_1, and just sets $I_2 = S_2$. It then returns the largest weight independent set from I_0, I_1 and I_2. By a good choice of a parameter the approximation guarantee is achieved.

If one tries to use the same approach on the independent set problem in $k+1$-claw free graphs it is the set S_0 that is causing trouble. In the bounded degree case it is trivially true that the greedy algorithm produces an independent set I_0 of total weight at least $\frac{w(S_0)}{\Delta+1}$ where $w(S_0)$ is the sum of the weights of the vertices in S_0. Intuitively S_0 is already a good structured set so it is not needed to do anything smarter than the greedy algorithm.

However, in the $k+1$-claw free case the greedy algorithm does not have such a sufficient performance guarantee. It has a performance guarantee of k, so the value of the greedy solution can be compared to the optimal independent set size in S_0. But it can not be compared to the weight (or the cardinality) of S_0.

Intuition and research direction Morally it seems there should not be such a difference between the independent set problem in bounded degree graphs and claw-free graphs.

Look at a vertex v in a claw-free graph and at its neighbours $N(v)$. As the maximum size of an independent set in $N(v)$ is at most k, one could look at $N(v)$ as the union of k cliques (in relation to the set packing instance: one clique for the sets that all share one of the k elements, modulo some duplicates). And for the independent set problem, a clique is not that different from a vertex: it is only possible to pick one of the vertices. It would be an interesting research direction to see where exactly the analogy with the bounded degree graphs stops, or to somehow change the algorithm for the bounded degree graphs and achieve an improved approximation guarantee for claw-free graphs.

The intuition in this paragraph inclines us to believe the approximation guarantee for k-set packing can be brought down further. Perhaps it it not $\frac{k}{\log k}$, which is the best known hardness bound, but we make the following conjecture.

Conjecture 5. *The approximation guarantee for k-set packing can be bounded by some function strictly smaller than $\frac{k}{3}$.*

7.4 Unweighted versus weighted k-set packing

General extension For most problems the weighted version is not much more difficult than the unweighted version. For example, also weighted matchings can be found in graphs in polynomial time, and for a lot of problems adding weights to the LP-formulation does not change anything significantly. However, for k-set packing the difference between the weighted version and the unweighted version is nontrivial.

Extending k-set packing As noted in Section 3.1, the analysis of the unweighted case hugely depends on the cardinality of every set. This is because the algorithms rely on local search and analyse a locally optimal solution. However, adding more sets than you remove from your current solution may not be advantageous in the weighted case, as the total weight might decrease while the cardinality of the solution increases. And in the weighted case, it could be the case that the total weight increases when you add less sets than you remove from your current solution. This proves hard to be incorporated in the analysis, and the weights of the sets cannot be handled in a straightforward way. This is why the results for the unweighted case do not easily extend to the weighted case. There have been less results on the weighted case and the algorithms do not immediately follow from the unweighted results, although they all use local search techniques.

Problem 6. *Improve the approximation guarantee for weighted k-set packing.*

Research direction The best weighted approximation algorithms reduce the problem to the independent set problem on $k+1$-claw free graphs. There are no indications so far that better results can be obtained in the weighted k-set packing problem by holding back from this reduction, so this seems a good way to look at the problem. These weighted approximation algorithms achieve approximation guarantees of approximately $\frac{2k}{3}$ [CH99] and $\frac{k}{2}$ [Ber00]. Section 7.2 considered possible ways to improve upon this last algorithm by changing it a little bit, aided by the simplified proof from Chapter 6. Berman and Krysta [BK03] considered what values of w^α are the best for every k when one searches a 2-locally optimal solution and achieve an approximation guarantee of about $\frac{2k}{3}$. A natural extension of this would be to use this alternate weight function in a search for t-locally optimal solutions for some $t > 2$. Considering such larger improving sets proved to work for the unweighted problem and it trivially works for the weighted case when one searches for improving sets of size n; the question is, how large do the improving sets have to be to get an improved approximation guarantee of, say, about $\frac{k}{3}$? Perhaps a value of $t = O(k)$ or $t = O(\log n)$ works. We think it is fruitful to investigate this possibility and either find a better approximation algorithm or an indication that this might not be helpful after all.

We believe a better approximation guarantee could be obtained by increasing the search space and we make the following conjecture.

Conjecture 7. *The approximation guarantee for weighted k-set packing can be bounded by $\frac{k+c}{3} + \varepsilon$ for some fixed $c \geq 0$.*

Appendix A

Parameterized complexity

Complexity	Reference			Remarks
$O^*((5.7m)^m)$	Jia, Zhang and Chen [JZC04]	R	3	
$O^*(10.88^{3m})$	Koutis [Kou05]	R	k	
$O^*(16^m)$	Chen et al [CLSZ07]	R	k	Also for weighted problem
$O^*(2^{3m})$	Koutis [Kou08]	R	k	
$O^*(1.493^{3m})$	Björklund et al [BHKK10]	R	k	
$O^*(2^{O(m)}(3m)!)$	Downey and Fellows [DF99]	D	3	
$O^*(25.6^{3m})$	Koutis [Kou05]	D	k	Originally $O^*(2^{O(mk)}) \geq O^*(32000^{3m})$ ([CLSZ07, LLCS06])
$O^*(13.78^{3m})$	Fellows et al [FHR+04]	D	k	Originally $exp(O(mk)) = O^*(12.7D)^{3m}$, $D \geq 10.4$ ([CLSZ07, LLCS06])
$O^*(12.8^{3m})$	Chen [Che13]	D	k	
$O^*(12.8^{3m})$	Liu, Chen and Wang [LCW07]	D	k	Also for weighted problem
$O^*(12.8^{3m})$	Chen et al [CLSZ07]	D	k	Also for weighted problem
$O^*(7.56^{3m})$	Wang and Feng [WF08a]	D	3	Also for weighted problem
$O^*(5.44^{3m})$	Chen and Chen [CC13]	D	k	
$O^*(4.61^{3m})$	Liu et al [LLCS06]	D	3	
$O^*(4^{3m})$	Chen et al [CKL+09]	D	k	Also for weighted problem
$O^*(3.523^{3m})$	Wang and Feng [WF08b]	D	3	
$O^*(32^m)$	Feng et al [FLLW09]	D	k	Also for weighted problem

Table A.1: Parameterized complexity results of randomised (R) and deterministic (D) algorithms for $3-SP$. Column 4 indicates whether the result only applies to $3-SP$ (3) or whether it is derived from a more general result from $k-SP$ (k).

Complexity	Reference		Remarks
$O^*(10.88^{mk})$	Koutis [Kou05]	R	
$O^*(4^{(k-1)m})$	Chen et al [CKL+09]	R	Also for weighted problem
$O^*(2^{mk})$	Koutis [Kou08]	R	
$O^*(f(m,k))$	Björklund et al [BHKK10]	R	$f(m,k) \approx \left(\frac{0.11 \cdot 2^m (1-\frac{1.64}{m})^{1.64-m} m^{0.68}}{(m-1)^{0.68}}\right)^k$
$O^*(g(k,m))$	Jia, Zhang and Chen [JZC04]	D	
$O^*(25.6^{mk})$	Koutis [Kou05]	D	Originally $O^*(2^{O(mk)})$ ([CLSZ07, LLCS06])
$O^*(13.78^{mk})$	Fellows et al [FKN+08]	D	Originally $exp(O(mk))$ ([CLSZ07, LLCS06])
$O^*(12.8^{mk})$	Chen [Che13]	D	
$O^*(12.8^{mk})$	Liu, Chen and Wang [LCW07]	D	Also for weighted problem
$O^*(5.44^{mk})$	Chen and Chen [CC13]	D	
$O^*(4^{mk})$	Chen et al [CKL+09]	D	Also for weighted problem
$O^*(2^{(2k-1)m})$	Feng et al [FLLW09]	D	Also for weighted problem

Table A.2: Parameterized complexity results of randomised (R) and deterministic (D) algorithms for $k - SP$.

Bibliography

[ADP80] G. Ausiello, A. D'Atri, and M. Protasi. Structure preserving reductions among convex optimization problems. *Journal of Computer and System Sciences*, 21(1):136 – 153, 1980.

[AF87] N. Alon and Z. Füredi. On the kernel of intersecting families. *Graphs and Combinatorics*, 3(1):91–94, 1987.

[AH97] Esther M. Arkin and Refael Hassin. On local search for weighted k-set packing. In *Proceedings of the 5th Annual European Symposium on Algorithms*, ESA '97, pages 13–22, London, UK, UK, 1997. Springer-Verlag.

[AKS09] Per Austrin, Subhash Khot, and Muli Safra. Inapproximability of vertex cover and independent set in bounded degree graphs. In *Proceedings of the 2009 24th Annual IEEE Conference on Computational Complexity*, CCC '09, pages 74–80, Washington, DC, USA, 2009. IEEE Computer Society.

[ALM+98] Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. Proof verification and the hardness of approximation problems. *J. ACM*, 45(3):501–555, May 1998.

[AYZ95] Noga Alon, Raphael Yuster, and Uri Zwick. Color-coding. *J. ACM*, 42(4):844–856, July 1995.

[Ber00] Piotr Berman. A $d/2$ approximation for maximum weight independent set in d-claw free graphs. *Nordic J. of Computing*, 7(3):178–184, September 2000.

[BF94] Piotr Berman and Martin Fürer. Approximating maximum independent set in bounded degree graphs. In *Proceedings of the fifth annual ACM-SIAM symposium on Discrete algorithms*, SODA '94, pages 365–371, Philadelphia, PA, USA, 1994. Society for Industrial and Applied Mathematics.

[BF95] Piotr Berman and Toshihiro Fujito. On approximation properties of the independent set problem for degree 3 graphs. In *In Proc. of Workshop on Algorithms and Data Structures*, pages 449–460. Springer, 1995.

[BH92] Ravi Boppana and Magnús M. Halldórsson. Approximating maximum independent sets by excluding subgraphs. *BIT*, 32(2):180–196, May 1992.

[BHKK10] Andreas Björklund, Thore Husfeldt, Petteri Kaski, and Mikko Koivisto. Narrow sieves for parameterized paths and packings. *CoRR*, abs/1007.1161, 2010.

[BK03] Piotr Berman and Piotr Krysta. Optimizing misdirection. In *Proceedings of the fourteenth annual ACM-SIAM symposium on Discrete algorithms*, SODA '03, pages 192–201, Philadelphia, PA, USA, 2003. Society for Industrial and Applied Mathematics.

[BNR95] Vineet Bafna, Babu O. Narayanan, and R. Ravi. Non-overlapping local alignments (weighted independent sets of axis parallel rectangles). In *Proceedings of the 4th International Workshop on Algorithms and Data Structures*, WADS '95, pages 506–517, London, UK, UK, 1995. Springer-Verlag.

[BTN01] Aharon Ben-Tal and Arkadiaei Semenovich Nemirovskiaei. *Lectures on modern convex optimization: analysis, algorithms, and engineering applications*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2001.

[CC13] Shenshi Chen and Zhixiang Chen. Faster deterministic algorithms for packing, matching and t-dominating set problems. *CoRR*, abs/1306.3602, 2013.

[CGM13] Marek Cygan, Fabrizio Grandoni, and Monaldo Mastrolilli. How to sell hyperedges: The hypermatching assignment problem. In *SODA*, pages 342–351, 2013.

[CH99] Barun Chandra and Magnús Halldórsson. Greedy local improvement and weighted set packing approximation. In *Proceedings of the tenth annual ACM-SIAM symposium on Discrete algorithms*, SODA '99, pages 169–176, Philadelphia, PA, USA, 1999. Society for Industrial and Applied Mathematics.

[Che13] Shenshi Chen. Monomial testing and applications. *CoRR*, abs/1303.0478, 2013.

[CHKX05] Jianer Chen, Xiuzhen Huang, Iyad A. Kanj, and Ge Xia. W-hardness under linear fpt-reductions: structural properties and further applications. In *Proceedings of the 11th annual international conference on Computing and Combinatorics*, COCOON'05, pages 975–984, Berlin, Heidelberg, 2005. Springer-Verlag.

[CK64] M. Calczyńska-Karlowicz. Theorem on families of finite sets. *Bulletin de l'Académie Polonaise Des Sciences. Série Des Sciences Mathématiques, Astronomiques et Physiques*, 12:87–89, 1964.

[CKL+09] J. Chen, J. Kneis, S. Lu, D. Mï¿½lle, S. Richter, P. Rossmanith, S. Sze, and F. Zhang. Randomized divide-and-conquer: Improved path, matching, and packing algorithms. *SIAM Journal on Computing*, 38(6):2526–2547, 2009.

[CL10] Yuk Hei Chan and Lap Chi Lau. On linear and semidefinite programming relaxations for hypergraph matching. In *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '10, pages 1500–1511, Philadelphia, PA, USA, 2010. Society for Industrial and Applied Mathematics.

[CLSZ07] 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.

[CMWY09] F. Chataigner, G. Manić, Y. Wakabayashi, and R. Yuster. Approximation algorithms and hardness results for the clique packing problem. *Discrete Appl. Math.*, 157(7):1396–1406, April 2009.

[CTW09] Zhi-Zhong Chen, Ruka Tanahashi, and Lusheng Wang. Note: An improved randomized approximation algorithm for maximum triangle packing. *Discrete Appl. Math.*, 157(7):1640–1646, April 2009.

[Cyg13] Marek Cygan. Improved approximation for 3-dimensional matching via bounded pathwidth local search. *CoRR*, abs/1304.1424, 2013.

[DF99] Rodney G. Downey and Michael R. Fellows. *Parameterized Complexity.* Springer-Verlag, 1999. 530 pp.

[EL73] Paul Erdös and László Lovász. Problems and results on 3-chromatic hypergraphs and some related questions. *Colloquia Mathematica Societatis János Bolyai*, 1973.

[EM72] Andrzej Ehrenfeucht and Jan Mycielski. Interpolation of functions over a measure space and conjectures about memory. *Approximation Theory*, 9:218–236, 1972.

[FF86] P. Frankl and Z. Füredi. Finite projective spaces and intersecting hypergraphs. *Combinatorica*, 6(4):335–354, December 1986.

[FHR+04] Mike Fellows, Pinar Heggernes, Frances Rosamond, Christian Sloper, and Jan Arne Telle. Finding k disjoint triangles in an arbitrary graph. In *Proceedings of the 30th international conference on Graph-Theoretic Concepts in Computer Science*, WG'04, pages 235–244, Berlin, Heidelberg, 2004. Springer-Verlag.

[FKN+08] M. R. Fellows, C. Knauer, N. Nishimura, P. Ragde, F. Rosamond, U. Stege, D. M. Thilikos, and S. Whitesides. Faster fixed-parameter tractable algorithms for matching and packing problems. *Algorithmica*, 52(2):167–176, August 2008.

[FKS93] Z. Füredi, J. Kahn, and P.D. Seymour. On the fractional matching polytope of a hypergraph. *Combinatorica*, 13(2):167–180, 1993.

[FLLW09] Qilong Feng, Yang Liu, Songjian Lu, and Jianxin Wang. Improved deterministic algorithms for weighted matching and packing problems. In *Proceedings of the 6th Annual Conference on Theory and Applications of Models of Computation*, TAMC '09, pages 211–220, Berlin, Heidelberg, 2009. Springer-Verlag.

[Fra78] Peter Frankl. On intersecting families of finite sets. *Journal of Combinatorial Theory, Series A*, 24(2):146 – 161, 1978.

[Fre04] Robert M. Freund. Introduction to semidefinite programming. Technical report, 2004.

[FY13] Martin Fürer and Huiwen Yu. Approximate the k-set packing problem by local improvements. *pre-print*, 2013.

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

[GL00] Rica Gonen and Daniel Lehmann. Optimal solutions for multi-unit combinatorial auctions: branch and bound heuristics. In *Proceedings of the 2nd ACM conference on Electronic commerce*, EC '00, pages 13–20, New York, NY, USA, 2000. ACM.

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

[GRS04] Carla P. Gomes, Rommel G. Regis, and David B. Shmoys. An improved approximation algorithm for the partial latin square extension problem. *Oper. Res. Lett.*, 32(5):479–484, September 2004.

[GT10] Mehmet Gulek and Ismail Hakki Toroslu. A dynamic programming algorithm for tree-like weighted set packing problem. *Inf. Sci.*, 180(20):3974–3979, October 2010.

[Hal95] Magnús M. Halldórsson. Approximating discrete collections via local improvements. In *Proceedings of the sixth annual ACM-SIAM symposium on Discrete algorithms*, SODA '95, pages 160–169, Philadelphia, PA, USA, 1995. Society for Industrial and Applied Mathematics.

[Hal98] Magnús Halldórsson. Approximations of independent sets in graphs. In Klaus Jansen and José Rolim, editors, *Approximation Algorithms for Combinatiorial Optimization*, volume 1444 of *Lecture Notes in Computer Science*, pages 1–13. Springer Berlin Heidelberg, 1998.

[Hal99] Magnús M. Halldórsson. Approximations of weighted independent set and hereditary subset problems. In *Proceedings of the 5th annual international conference on Computing and combinatorics*, COCOON'99, pages 261–270, Berlin, Heidelberg, 1999. Springer-Verlag.

[Hal02] Eran Halperin. Improved approximation algorithms for the vertex cover problem in graphs and hypergraphs. *SIAM J. Comput.*, 31(5):1608–1623, May 2002.

[Hås96] J. Håstad. Clique is hard to approximate within $n^{1-\varepsilon}$. In *Proceedings of the 37th Annual Symposium on Foundations of Computer Science*, FOCS '96, pages 627–, Washington, DC, USA, 1996. IEEE Computer Society.

[Hås01] Johan Håstad. Some optimal inapproximability results. *J. ACM*, 48(4):798–859, July 2001.

[HJKS07] Iman Hajirasouliha, Hossein Jowhari, Ravi Kumar, and Ravi Sundaram. On completing latin squares. In *Proceedings of the 24th annual conference on Theoretical aspects of computer science*, STACS'07, pages 524–535, Berlin, Heidelberg, 2007. Springer-Verlag.

[HKT00] Magnús M. Halldórsson, Jean Kratochvíl, and Jean Arne Telle. Independent sets with domination constraints. *Discrete Appl. Math.*, 99(1-3):39–54, February 2000.

[HL97] Magnús M. Halldórsson and Hoong Chuin Lau. Low-degree graph partitioning via local search with applications to constraint satisfaction, max cut, and coloring. *JOURNAL OF GRAPH ALGORITHMS AND APPLICATIONS*, 1(3):1–13, 1997.

[Hoc83] Dorit S. Hochbaum. Efficient bounds for the stable set, vertex cover and set packing problems. *Discrete Applied Mathematics*, 6(3):243 – 254, 1983.

[HR94a] Magnús Halldórsson and Jaikumar Radhakrishnan. Greed is good: approximating independent sets in sparse and bounded-degree graphs. In *Proceedings of the twenty-sixth annual ACM symposium on Theory of computing*, STOC '94, pages 439–448, New York, NY, USA, 1994. ACM.

[HR94b] Magnús M. Halldórsson and Jaikumar Radhakrishnan. Improved approximations of independent sets in bounded-degree graphs via subgraph removal. *Nordic J. of Computing*, 1(4):475–492, December 1994.

[HR06] Refael Hassin and Shlomi Rubinstein. An approximation algorithm for maximum triangle packing. *Discrete Appl. Math.*, 154(6):971–979, April 2006.

[HS89] C. A. J. Hurkens and A. Schrijver. On the size of systems of sets every t of which have an sdr, with an application to the worst-case ratio of heuristics for packing problems. *SIAM J. Discret. Math.*, 2(1):68–72, February 1989.

[HSS06] Elad Hazan, Shmuel Safra, and Oded Schwartz. On the complexity of approximating k-set packing. *Comput. Complex.*, 15(1):20–39, May 2006.

[IP99] Russell Impagliazzo and Ramamohan Paturi. The complexity of k-sat. In *Proceedings of the Fourteenth Annual IEEE Conference on Computational Complexity*, COCO '99, pages 237–, Washington, DC, USA, 1999. IEEE Computer Society.

[JPY88] David S. Johnson, Christos H. Papadimitriou, and Mihalis Yannakakis. How easy is local search? *J. Comput. Syst. Sci.*, 37(1):79–100, August 1988.

[JZC04] Weijia Jia, Chuanlin Zhang, and Jianer Chen. An efficient parameterized algorithm for m-set packing. *J. Algorithms*, 50(1):106–117, January 2004.

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

[Kho02] Subhash Khot. On the power of unique 2-prover 1-round games. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, STOC '02, pages 767–775, New York, NY, USA, 2002. ACM.

[Knu94] Donald E. Knuth. The sandwich theorem. *ELECTRONIC J. COMBINATORICS*, 1:1, 1994.

[Kou05] Ioannis Koutis. A faster parameterized algorithm for set packing. *Inf. Process. Lett.*, 94(1):7–9, April 2005.

[Kou08] Ioannis Koutis. Faster algebraic algorithms for path and packing problems. In *Proceedings of the 35th international colloquium on Automata, Languages and Programming, Part I*, ICALP '08, pages 575–586, Berlin, Heidelberg, 2008. Springer-Verlag.

[KW09] Ioannis Koutis and Ryan Williams. Limits and applications of group algebras for parameterized problems. In *Proceedings of the 36th International Colloquium on Automata, Languages and Programming: Part I*, ICALP '09, pages 653–664, Berlin, Heidelberg, 2009. Springer-Verlag.

[LCW07] Yunlong Liu, Jianer Chen, and Jianxin Wang. Parameterized algorithms for weighted matching and packing problems. In *Proceedings of the 4th international conference on Theory and applications of models of computation*, TAMC'07, pages 692–702, Berlin, Heidelberg, 2007. Springer-Verlag.

[LLCS06] Yang Liu, Songjian Lu, Jianer Chen, and Sing-Hoi Sze. Greedy localization and color-coding: improved matching and packing algorithms. In *Proceedings of the Second international conference on Parameterized and Exact Computation*, IWPEC'06, pages 84–95, Berlin, Heidelberg, 2006. Springer-Verlag.

[Lov79] L. Lovász. On the shannon capacity of a graph. *Information Theory, IEEE Transactions on*, 25(1):1–7, 1979.

[Lov95] László Lovï¿½sz. Semidefinite programs and combinatorial optimization (lecture notes), 1995.

[Min80] George J. Minty. On maximal independent sets of vertices in claw-free graphs. *J. Comb. Theory, Ser. B*, 28(3):284–304, 1980.

[NT99] Daishin Nakamura and Akihisa Tamura. A revision of minty's algorithm for finding a maximum weight stable set of a claw-free graph, 1999.

[Pis06] David Pisinger. Semidefinite programming - an introduction. nov 2006.

[PS82] Christos H. Papadimitriou and Kenneth Steiglitz. *Combinatorial optimization: algorithms and complexity*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1982.

[RTS00] Jaikumar Radhakrishnan and Amnon Ta-Shma. Bounds for dispersers, extractors, and depth-two superconcentrators. *SIAM J. Discret. Math.*, 13(1):2–24, January 2000.

[San02] Tuomas Sandholm. Algorithm for optimal winner determination in combinatorial auctions. *Artif. Intell.*, 135(1-2):1–54, February 2002.

[Ski08] Steven S. Skiena. *The Algorithm Design Manual*. Springer Publishing Company, Incorporated, 2nd edition, 2008.

[SLA+99] Tuomas Sandholm, Kate Larson, Martin Andersson, Onn Shehory, and Fernando Tohmé. Coalition structure generation with worst case guarantees. *Artif. Intell.*, 111(1-2):209–238, July 1999.

[SW13] Maxim Sviridenko and Justin Ward. Large neighborhood local search for the maximum set packing problem. *CoRR*, abs/1302.4347, 2013.

[Tuz85] Zsolt Tuza. Critical hypergraphs and intersecting set-pair systems. *Journal of Combinatorial Theory, Series B*, 39(2):134 – 145, 1985.

[VB94] Lieven Vandenberghe and Stephen Boyd. Semidefinite programming. *SIAM Review*, 38:49–95, 1994.

[Vem98] R.R. Vemuganti. Applications of set covering, set packing and set partitioning models: A survey. In *Handbook of combinatorial optimization*, volume 1, pages 573–746. Boston: Kluwer Academic Publishers, 1998.

[Vic61] William Vickrey. Counterspeculation, auctions, and competitive sealed tenders. *The Journal of Finance*, 16(1):8–37, 1961.

[Vis96] Sundar Vishwanathan. Personal communication to Magnús Halldórsson, cited in [Hal98], 1996.

[WF08a] Jianxin Wang and Qilong Feng. Improved parameterized algorithms for weighted 3-set packing. In *Proceedings of the 14th annual international conference on Computing and Combinatorics*, COCOON '08, pages 130–139, Berlin, Heidelberg, 2008. Springer-Verlag.

[WF08b] Jianxin Wang and Qilong Feng. An $o^*(3.523k)$ parameterized algorithm for 3-set packing. In *Proceedings of the 5th international conference on Theory and applications of models of computation*, TAMC'08, pages 82–93, Berlin, Heidelberg, 2008. Springer-Verlag.

This figure "Example.jpg" is available in "jpg" format from:

http://arxiv.org/ps/1507.07459v1

This figure "FanoPlane.jpg" is available in "jpg" format from:

http://arxiv.org/ps/1507.07459v1

This figure "WheelGraph.jpg" is available in "jpg" format from:

http://arxiv.org/ps/1507.07459v1