University of Warsaw
Faculty of Mathematics, Informatics and Mechanics

Paweł Rzążewski

Exact algorithms for graph-theoretic frequency assignment problems

PhD dissertation

Supervisor
prof dr hab. Zbigniew Lonc

Auxilary supervisor
dr Konstanty Junosza-Szaniawski

Faculty of Mathematics
and Information Science
Warsaw University of Technology

October 2014

Abstract

Many real-life problems have very natural graph-theoretic models. One of such problems is the frequency assignment problem. It asks for an assignment of channels of frequency to transmitters in a broadcast network, so that no pair of transmitters interfere with each other. Depending on a way how we define the interference, this leads to many well-studied graph theoretic problems.

For example, an $L(2,1)$-labeling of a graph G is a mapping from the vertex set of G to the set of non-negative integers, such that the labels assigned to adjacent vertices differ by at least 2 and labels assigned to vertices with a common neighbor are different. The span of such a labeling is the maximum label used and the $L(2,1)$-labeling problem asks for the minimum span of an $L(2,1)$-labeling of G.

Other models considered in this context are the graph coloring problem, the channel assignment problem, the T-coloring problem etc.

Since all the problems mentioned above are NP-hard, there is little hope to solve them exactly in a polynomial time. Thus in this dissertation we are mostly interested in designing exact algorithms, whose complexities are bounded by exponential functions of the size of the input.

We describe and analyze a general framework, which can be used to solve some type of graph labeling problems. More precisely, we design an algorithm, which solves the so-called generalized list T-coloring problem, which is a common generalization of all the problems mentioned above. We also show how to improve the complexity analysis if the input graph has some specific structure.

Then we show how a refined version of this general algorithm works for the $L(2,1)$-labeling problem. The time and space complexity of our refined algorithm is $\mathcal{O}(2.6488^n)$.

We also present how to adapt the general algorithm to obtain algorithms determining the existence of a homomorphism and a locally injective homomorphism from a graph G to a graph H, both working in time and space $\mathcal{O}^*((b+2)^n)$, where b is the bandwidth of the complement of H.

Using a different approach, we construct an exact exponential algorithm for the $L(2,1)$-labeling problem, working in time $\mathcal{O}(7.4920^n)$ and using only polynomial space. We also modify this algorithm to determine if the input graph has an $L(2,1)$-labeling with the span at most k (for a constant k). This approach proves significantly better than the basic algorithm for $k \leq 31$.

Estimating time complexities of exact exponential algorithms often requires solving some problems from extremal graph theory. In our case, we give some upper and lower bounds for the maximum number of 2-packings in a graph (i.e. independent sets in a square of the graph) and some related graph-theoretic objects.

KEYWORDS: exact algorithm, graph labeling, frequency assignment problem, $L(2,1)$-labeling

ACM SUBJECT CLASSIFICATION: F.2.2, G.2.1, G.2.2

Streszczenie

Wiele problemów napotykanych w przemyśle czy biznesie daje się w naturalny sposób modelować za pomocą grafów. Jednym z takich problemów jest problem przydziału częstotliwości. Polega on na takim przypisaniu kanałów częstotliwości nadajnikom sieci radiowej, aby uniknąć zakłóceń w nadawaniu sygnału. W zależności od tego, jak zdefiniujemy zakłócenia (i, co za tym idzie, konflikty między nadajnikami), możemy otrzymać wiele znanych problemów grafowych.

Na przykład etykietowanie $L(2,1)$ grafu G to przypisanie wierzchołkom G nieujemnych liczb całkowitych w taki sposób, że etykiety sąsiadujących wierzchołków różnią się o co najmniej 2, zaś etykiety wierzchołków o wspólnym sąsiedzie są różne. Rozpiętością takiego etykietowania jest number największej użytej etykiety. Problem etykietowania $L(2,1)$ polega na znalezieniu najmniejszej możliwej rozpiętości etykietowania $L(2,1)$ grafu.

Inne problemy rozważane w tym kontekście to: kolorowanie grafów, problem przydziału kanałów częstotliwości (ang. *channel assignment problem*), problem T-kolorowania itp.

Ponieważ wszystkie wspomniane problemy są NP-trudne, szanse na znalezienie ich dokładnych rozwiązań w czasie wielomianowym są niewielkie. Dlatego w niniejszej pracy zajmujemy się projektowaniem algorytmów dokładnych o złożonościach ograniczonych pewną wykładniczą funkcją rozmiaru zadania.

Opisujemy ogólną metodę, która może zostać zastosowana do rozwiązania pewnego typu problemów etykietowania grafów. Dokładniej, prezentujemy algorytm rozwiązujący problem tzw. uogólnionego listowego T-kolorowania (ang. *generalized list T-coloring problem*), będący wspólnym uogólnieniem wszystkich problemów wymienionych powyżej. Pokazujemy również, jak zmienia się złożoność algorytmu dla grafów o pewnej określonej strukturze.

Następnie pokazujemy ulepszoną wersję tego algorytmu działającą dla problemu $L(2,1)$-etykietowania grafów. Złożoność obliczeniowa i pamięciowa naszego algorytmu wynosi $\mathcal{O}(2.6488^n)$.

Ponadto, pokazujemy jak zmodyfikować ogólny algorytm, by otrzymać algorytmy weryfikujące istnieje homomorfizmu i lokalnie różnowartościowego homomorfizmu (ang. *locally injective homomorphism*) grafu G w graf H. Złożoność obliczeniowa i pamięciowa obu tych algorytmów wynosi $\mathcal{O}^*((b+2)^n)$, gdzie b oznacza *bandwidth* dopełnienia grafu H.

Używając innej metody, konstruujemy algorytm dokładny dla problemu etykietowania $L(2,1)$, działający w czasie $\mathcal{O}(7.4920^n)$ i pamięci wielomianowej. Następnie pokazujemy pewną modyfikację tej metody pozwalającą odpowiedzieć na pytanie, czy dany graf ma etykietowanie $L(2,1)$ o rozpiętości co najwyżej k (gdzie k jest stałą). To podejście jest bardziej efektywne niż ogólny algorytm dla $k \leq 31$.

Szacowanie złożoności dokładnych algorytmów wykładniczych często wymaga rozwiązania pewnych problemów teorii grafów ekstremalnych. W naszym przypadku, pokazujemy górne i dolne ograniczenia na maksymalną liczbę zbiorów 2-niezależnych w grafie (czyli zbiorów niezależnych w kwadracie grafu) oraz innych, powiązanych, obiektów kombinatorycznych.

Słowa kluczowe: algorytm dokładny, etykietowanie grafów, problem przydziału częstotliwości, etykietowanie $L(2,1)$

Klasyfikacja Tematyczna ACM: F.2.2, G.2.1, G.2.2

Acknowledgements

I am sincerely grateful to dr. Konstanty Junosza-Szaniawski, my co-supervisor and my friend, for introducing me to research in general and to exact exponential algorithms in particular, and for his help and guidance throughout the years.

I also acknowledge the help of my supervisor, prof. Zbigniew Lonc. I am grateful for useful advice and also his patience and extreme meticulousness.

I am thankful to my Parents, my Wife and my Family for their constant support and encouragement.

Finally, I would like to express my gratitude to the co-authors of papers, which are the part of this dissertation: Jan Kratochvíl, Peter Rossmanith, Mathieu Liedloff, and to my colleagues from the 5th floor, especially Tomasz Brengos, Krzysztof Kaczmarski, Paweł Naroski and Michał Stronkowski. Thank you for everything you have taught me.

Contents

Chapter 1

Introduction

Among graph-theoretic problems, a considerable attention from the discrete mathematics and theoretical computer science community has been received by various graph labeling models. They are extensively studied both for their practical motivations, and for their interesting theoretical and structural properties.

One of the typical applications of graph labeling models is the frequency assignment problem. It is the problem of assigning channels (usually represented by non-negative integers) to each radio transmitter in a network so that no pair of transmitters interfere with each other. Hale [42] first formulated this problem in terms of so-called T-coloring of graphs. Since then, many other, related models have also arisen.

1.1 Graph labeling problems

In this section we present some of the the problems we are going to investigate in this dissertation. The descriptions of problems in this section are somewhat informal. The formal definitions will be provided later on, in Section 4.1.

Graph coloring. Although the graph coloring problem dates back to the 19th century, it still receives considerable attention of many researchers. We refer the reader to the book by Jensen and Toft [52] to get some information about the history and many still open problems in graph coloring. In the classical graph coloring problem, we want to assign colors (usually represented by natural numbers) to the vertices of the graph in such a way, that no two adjacent vertices receive the same color. The minimum number of colors needed to color a graph G in this way is called the *chromatic number* of G and denoted by $\chi(G)$.

The list version of this problem has been introduced independently by Vizing [85] and by Erdős, Rubin and Taylor [25]. Here each vertex has a list of colors and its color has to be chosen from this list (a non-list coloring can be seen as a special case of the list coloring with all lists equal).

$L(p,q)$-labeling. The notion of $L(p,q)$-labeling (for integers $p \geq q \geq 1$) is inspired by the frequency assignment problem in telecommunications. We look for such a labeling of the vertices G with integers, in which the labels of adjacent vertices differ by at least p and the labels of the vertices with a common neighbor differ by at least q. The most widely studied case is the $L(2,1)$-labeling, which was introduced by Roberts (according to [41]). In this model the vertices of the input graph correspond to transmitters of the network and the edges indicate which pairs of transmitters are too close to each other so that interference could occur even if the broadcasting channels were just one apart. The second condition follows from the requirement that no transmitter should have two or more close neighbors transmitting on the same frequency.

The *span* of an $L(2,1)$-labeling is the difference between the largest and smallest labels used. By $\lambda(G)$ we denote the $L(2,1)$-*span* of a graph G, which is the smallest span over all $L(2,1)$-labelings of G. A considerable attention has been given to bounding the value of $\lambda(G)$ by some function of G.

Griggs and Yeh [41] proved that $\lambda(G) \leq \Delta^2 + 2\Delta$ (where Δ denotes the largest vertex degree in G) and conjectured that $\lambda(G) \leq \Delta^2$ for every graph G. There are several results supporting this conjecture, for example Gonçalves [40] proved that $\lambda(G) \leq \Delta^2 + \Delta - 2$ for graphs with $\Delta \geq 3$. Lu [70] showed that the conjecture is satisfied for graphs with at most $(\lfloor \Delta/2 \rfloor + 1)(\Delta^2 - \Delta + 1) - 1$ vertices. Havet *et al.* [46] settled the conjecture in affirmative for graphs with $\Delta \geq 10^{69}$. For graphs with smaller Δ, the conjecture still remains open.

Yet it is interesting to note that the Petersen and Hoffmann-Singleton graphs are the only two known graphs that satisfy equality in this bound (for maximum degree greater than 2). Recently Lu [70] presented an infinite family of graphs with span $\Delta^2 - \Delta + 1$, which is the largest value for any known infinite family.

We refer the reader to surveys on the $L(p,q)$-labeling problem by Yeh [88] and Calamoneri [14]. The list version of this problem has been studied for example by Fiala and Škrekovski [35].

It is worth mentioning that the case of the $L(0,1)$-labeling has also received some attention (see or example Fiala, Golovach and Kratochvíl [26]). We may define the $L(0,1)$-labeling of G as the coloring of $G^2 - G$ (for a graph $G = (V, E)$, by G^2 we mean the graph, whose vertex set is V and its edge set is $E \cup \{uv\colon u \text{ and } v \text{ have a common neighbor in } G\}$).

Channel assignment. The channel assignment problem is a generalization of $L(p,q)$-labeling. Here every edge uv has a weight $\omega(uv)$ and we require that the difference of labels given to vertices u and v is at least $\omega(uv)$. We refer the reader to the survey by Král' [64] for more information about this problem.

T-coloring. In this problem we have a set T of forbidden differences and we ask for such a labeling of vertices of a graph, that the difference of labels given to the endvertices of any edge is not in T. Unlike the channel assignment problem, T-coloring allows the case when forbidden differences do not form a set of consecutive integers. It is interesting to mention that for $T = \{0, 7, 14, 15\}$ we obtain the model for interferences for the UHF transmitters (see McDiarmid [72]). The list version of this problem has been studied for example by Alon and Zaks [5].

Generalized (list) T-coloring. In the multitude of various labeling problems, a natural tendency is to look for the similarities between different models and try to unify and generalize them. In this spirit, Fiala, Král' and Škrekovski [28] defined and studied the so-called generalized list T-coloring problem, which unifies the channel assignment problem and the T-coloring problem. Here each edge has its own set of forbidden differences (which does not have to be an interval).

Graph homomorphism (or H-coloring). Graph homomorphism problem (or H-coloring, as it is sometimes called) is a natural generalization of a well-known graph coloring problem. For graphs G and H we say that $\varphi\colon V(G) \to V(H)$ is a homomorphism from G to H if $\varphi(v)\varphi(u) \in E(H)$

for any $uv \in V(G)$. In other words, a homomorphism is an edge-preserving mapping from $V(G)$ to $V(H)$. Thus the problem of coloring a graph with k colors is equivalent to the problem of finding a homomorphism to the complete graph K_k. We refer the reader to the monography by Hell and Nešetřil [49] for more information about graph homomorphisms.

Locally injective graph homomorphism. Some special cases of graph homomorphisms, namely locally constrained graph homomorphisms, have also received a considerable attention (see the survey by Fiala and Kratochvíl [32] for more information about the topic). We say that the homomorphism φ from G to H is *locally injective* (*locally surjective*; *locally bijective*) if the neighborhood of $v \in V(G)$ is mapped injectively (resp.: surjectively; bijectively) to the neighborhood of $\varphi(v)$. We will be mostly interested in locally injective homomorphisms. They can be seen as homomorphisms from G to H, in which no two vertices from G with a common neighbor are mapped to the same vertex of H.

1.2 Complexity results

In this section we list some results concerning the computational complexity of the mentioned problems. We are mostly interested in NP-hard cases.

Graph coloring. The problem of deciding 3-colorability of an input graph is one of Karp's 21 NP-complete problems listed in [61]. The problem remains NP-complete even if the input graph is planar and has maximum degree at most 4 (see Garey, Johnson and Stockmeyer [38]).

$L(2,1)$-labeling. Griggs and Yeh [41] proved that computing $\lambda(G)$ is an NP-hard problem. Fiala *et al.* [27] later proved that the k-$L(2,1)$-labeling problem (i.e. deciding if $\lambda(G) \leq k$) remains NP-complete for every fixed $k \geq 4$ (for $k \leq 3$ the problem is polynomial). NP-completeness for planar inputs was proved by Bodlaender *et al.* [10] for $k = 8$, by Janczewski *et al.* [51] for $k = 4$ and finally by Eggeman *et al.* [23] for all $k \geq 4$.

The problem is also NP-hard for regular graphs (see Fiala, Kratochvíl [30]). The fact that distance constrained labeling is a more difficult task than ordinary coloring is probably most strikingly documented by the NP-completeness of deciding if $\lambda(G) \leq k$ for series-parallel graphs (see Fiala *et* *al.* [36]) – here of course k is part of the input. On the other hand, the chromatic number of a series-parallel graph can be easily decided in linear time (see for example Seymour [80]).

Griggs and Yeh [41] raised the question of computational complexity of determining $\lambda(G)$ for trees. It was answered by Chang and Kuo [15] by constructing a polynomial time algorithm. Their result was later improved to a linear time algorithm by Hasunuma *et al.* [43].

Graph homomorphism and locally constrained graph homomorphism. Graph homomorphisms are also interesting from the computational point of view. In their celebrated theorem, Hell and Nešetřil [48] showed that determining if G has a homomorphism to H is polynomial if H is bipartite and NP-complete otherwise. For a locally surjective homomorphism Fiala and Paulusma [34] showed that determining the existence of a locally surjective homomorphism from G to a connected graph H is polynomial if H has at most 2 vertices and NP-complete otherwise. They also showed a full dichotomy for the case when H is disconnected, but the description of polynomial cases is more complicated. There is no similar characterization for the case of locally injective homomorphisms and locally bijective homomorphisms, but still we can find some partial results (see for example [29, 31, 33] for some results on locally injective homomorphisms and [1, 67] for locally bijective homomorphisms).

1.3 Exact exponential algorithms

When dealing with an NP-hard problem, there is little hope to design an exact algorithm, running in polynomial time. Therefore a recent trend in algorithmic research is designing exact exponential time algorithms for NP-hard problems, while trying to minimize the constant which is the base of the exponential running time function.

Graph coloring. Many such algorithms have been presented for the most widely studied of the mentioned problems, i.e. the graph coloring (see for example Lawler [68], Eppstein [24], Byskov [13]). The currently best exact algorithm for graph coloring was presented by Björklund *et al.* [9] and runs in time $\mathcal{O}^*(2^n)^\dagger$, using exponential space. If only polynomial space is available, one can use the same framework with Wahlström's [86] algorithm for counting independent sets as a subroutine. Then the time complexity is $\mathcal{O}(2.2377^n)$.

It is worth mentioning that algorithms for graph coloring work also for the $L(1,1)$-labeling and the $L(0,1)$-labeling problem for a graph G, since they are equivalent to coloring G^2 and $G^2 - G$, respectively.

Channel assignment. The best algorithm for the general case of the channel assignment problem, was proposed by McDiarmid [73] and runs in time $\mathcal{O}^*(n!)$. Much attention has been given to the so-called ℓ-bounded instances of the problem (for an integer ℓ). We say that an instance of the channel assignment problem is *ℓ-bounded* if the maximum weight of an edge is at most ℓ. The first non-trivial exact algorithm for the ℓ-bounded channel assignment problem with time complexity $\mathcal{O}^*((2\ell+1)^n)$ was given by McDiarmid [73]. Then Král' [63] presented an algorithm running in time $\mathcal{O}^*((\ell+2)^n)$. This bound was beaten by Cygan and Kowalik [21], who showed an algorithm with time complexity $\mathcal{O}^*((\ell+1)^n)$. Their approach uses a well-known inclusion-exclusion principle and fast zeta transform.

Very recently, Kowalik and Socała [62] published an algorithm for the ℓ-bounded channel assignment problem working in time $\mathcal{O}(2^n(\ell+2)^{n/2} \cdot n^2)$, using the so-called *meet-in-the-middle* approach. It still remains a great challenge to design an exact algorithm for the channel assignment problem with time complexity bounded by $\mathcal{O}^*(c^n)$ for c being a constant (or to prove that there is no such algorithm, under some standard complexity assumptions).

Another interesting branch of research is to prove the hardness of problems under some reasonable complexity assumptions. The assumption usually used in such results is the Exponential Time Hypothesis (ETH), introduced by Impagliazzo and Paturi [50]. It says that there is no algorithm for the 3-SAT problem working in time $2^{o(n)}$, where n is the number of variables in the input formula. Very recently Socała [81] proved that, assuming the ETH, the channel assignment problem cannot be solved in time $2^{o(n \log n)}$. Note that McDiarmid's algorithm works in time $\mathcal{O}^*(n!) = 2^{\mathcal{O}(n \log n)}$, so this bound is asymptotically tight.

$L(2,1)$-labeling. The first exact exponential algorithm dedicated for the $L(2,1)$-labeling problem, whose complexity does not depend on the number

†In the $\mathcal{O}^*$ notation we suppress polynomially bounded terms.

of labels, was presented by Havet *et al.* [45] and works in time $\mathcal{O}(3.8739^n)$.

This algorithm has been modified and refined by Junosza-Szaniawski and Rzążewski [56, 57], obtaining time complexity $\mathcal{O}(3.2361^n)$. A lower-bound of $\Omega(3.0731^n)$ for the worst-case running-time of this algorithm has also been provided.

Breaking the barrier of $\mathcal{O}^*(3^n)$ still seemed hardly attainable. The breakthrough came with the publication of the $\mathcal{O}(2.6488^n)$ algorithm by Junosza-Szaniawski *et al.* [53]. The algorithm uses the framework first invented by Rossmanith [77]. Moreover, the analysis of this algorithm required defining and solving some additional extremal combinatorial problems. The details of the algorithm are described in Section 4.3 of this thesis.

All the algorithms mentioned above use exponential amount of memory. Havet *et al.* [45] presented a branching algorithm which determines if $\lambda(G) \leq k$ in time $O^*((k-2.5)^n)$ and polynomial space. This bound was improved for small k. Havet *et al.* [45] designed an algorithm for the 4-$L(2,1)$-labeling problem with complexity $\mathcal{O}(1.3006^n)$ and Couturier *et al.* [20] showed that 5-$L(2,1)$-labeling problem can be solved in time $\mathcal{O}(1.7990^n)$, when the input graphs is cubic.

Havet *et al.* [44] and independently Junosza-Szaniawski and Rzążewski [55] applied the well-known *divide and conquer* paradigm to construct a polynomial-space exact algorithm for the $L(2,1)$-labeling problem, whose complexity does not depend on the number of labels used. Both groups obtained (but did not publish) the same algorithm with time complexity $\mathcal{O}((9+\epsilon)^n)$ for arbitrarily small positive constant ϵ. Finally, the refined algorithm with time complexity of $\mathcal{O}(7.4920^n)$ was published by Junosza-Szaniawski *et al.* [54]. The authors also adapted their method to obtain a significantly better time complexity if the number k of available labels is a small fixed integer. This approach outperforms the general $\mathcal{O}(7.4920^n)$ algorithm for $k \leq 31$. To obtain such complexity bounds, it was necessary to consider some extremal combinatorial problems. The details are described in Chapter 5.

Generalized list T-coloring. Although the algorithm by Cygan and Kowalik [21] is designed for the channel assignment problem, it can be be adapted to solve the generalized T-coloring problem. Its time complexity is then $\mathcal{O}^*((\tau+2)^n)$, where τ is the maximum forbidden difference. Junosza-Szaniawski and Rzążewski [59] obtained the same time complexity, by adapting the algorithm for the $L(2,1)$-labeling problem [53]. They were also able to improve the time complexity of the algorithm for graphs having some special structure. The graphs considered were: bounded degree graphs, $K_{1,d}$-free graphs (for integer d) and graphs having a clique factor, i.e. a spanning subgraph whose every connected component is a clique with at least 2 vertices. The results are described in Section 4.2.

Assuming the ETH, Kowalik and Socała [62] proved that the non-list version of the generalized T-coloring problem for a graph with n vertices cannot be solved in time $2^{2^{o(\sqrt{n})}} \cdot r^{\mathcal{O}(1)}$ (where r is the size of the instance in bits). This implies that there is no algorithm with complexity $\mathcal{O}^*(c^n)$ for any constant c or even with complexity $\mathcal{O}(n!)$.

Graph homomorphism and locally constrained graph homomorphism. When designing exact exponential algorithms to determine the existence of a homomorphism from G to H, we usually try to express the basis of the exponential factor in a complexity bound as a function of some invariant of H. Fomin *et al.* [37] presented an algorithm for determining the existence of a homomorphism from G to H, working in time $\mathcal{O}^*((2\,\mathrm{tw}(H)+1)^n)$, where $\mathrm{tw}(H)$ denotes a *treewidth* of the graph H (see Diestel's book [22] for some information about the treewidth of graphs) and n is the number of vertices of G. For a locally injective homomorphisms, Havet *et al.* [45] presented an algorithm working in time $\mathcal{O}^*((\Delta(H)-1)^n)$. To the best of our knowledge there are no similar results for the locally surjective and the locally bijective graph homomorphism problems.

Recently, Rzążewski [78] presented an algorithm determining the existence of a homomorphism from G to H, working in time $\mathcal{O}^*((b+2)^n)$, where b denotes the *bandwidth* of the complement of H and n is the number of vertices of G (see for example [16, 18] for more information about bandwidth). The algorithm uses the same framework as the algorithm for the $L(2,1)$-labeling problem by Junosza-Szaniawski *et al.* [53]. Moreover, it can be adapted to solve the locally injective graph homomorphism problem within the same time complexity bound. The details of this algorithm are presented in Section 4.4.

1.4 Related combinatorial problems

The complexity of exact exponential algorithms is often determined by the number of combinatorial objects (e.g. sets, sequences, tuples of sets) that need to be enumerated and analyzed. For example, the bounds for the maximum number of maximal independent sets (all such sets or only the ones with given cardinality) play a crucial role in the analysis of many exact algorithms for the graph coloring problem (see Lawler [68], Eppstein [24] or Byskov [13]). Independent sets are important to the graph coloring, because every color induces an independent set. In the case of the $L(2,1)$-labeling (and, in general, $L(p,q)$-labeling) each label induces a set containing no two vertices in distance at most 2 from each other. Such sets are called *2-packings*. We can think of 2-packings in a graph G as of independent sets in the graph G^2.

A notion of 2-packings (and more generally, k-packings, having no two vertices at distance at most k) has been introduced by Meir and Moon [74]. Some authors refer to 2-packings as 2-independent sets (see [12]) or 2-stable sets (see [15]).

The notion of 2-packings has been extensively studied. Meir an Moon [74] showed that the size of the largest 2-packing in a tree equals the size of the maximum dominating set. For general graphs, it is NP-complete to decide if those two values are equal (see Kratochvíl [65]). Also determining the size of the largest 2-packing is NP-hard, even for cubic graphs (see Kratochvíl [66]).

The question how large the number of 2-packings in a graph can be arises naturally in the analysis of an exacts algorithm for the $L(2,1)$-labeling problem by Havet *et al.* [45]. The authors showed that $u_k(n)$, being the maximum number of k-element 2-packings over all connected graphs on n vertices does not exceed $\binom{n/2}{k}2^k$.

Junosza-Szaniawski and Rzążewski [56] improved this result by showing that $u_k \leq \binom{n-k+1}{k}$. If we are interested in the number of all 2-packings in a connected graph on n vertices (denoted by $u(n)$), this bound gives us $u(n) = \mathcal{O}(1.6181^n)$ (we add up the bounds on $u_k(n)$ over all values of k). This was again improved by Junosza-Szaniawski and Rzążewski [58], who showed that $u(n) = \mathcal{O}(1.5400^n)$. Both bounds, along with some other auxilary combinatorial results are described in Chapter 3 of this dissertation.

Observe that if we drop the connectivity restriction, the question about the maximum number of 2-packings becomes trivial – in a graph with n vertices and no edges the number of 2-packings is equal to 2^n, where n denotes the number of vertices in this graph.

1.5 Main results and organization of the dissertation

The main results presented in this dissertation are as follows. Chapter 3 is purely combinatorial. It presents the following main results.

- We give new upper and lower bounds for the maximum number of k-element 2-packings in a connected graph (Section 3.1).
- We also provide new upper and lower bounds for the maximum number of all 2-packings in a connected graph (Section Section 3.2). We also consider some special classes of graphs (regular graphs, claw-free graphs).

  Moreover, we provide some auxilary combinatorial results, which will be useful later, in the analysis of the complexity of our algorithms.
- We give upper and lower bounds for the the maximum number of pairs of disjoint sets, one of which is a 2-packing (we call such pairs *proper*) – see Section 3.3.
- In Section 3.4 we define the notion of so-called *red-black graphs*, which are basically graphs with two kinds of edges – black and red. We give upper and lower bounds for the maximum number of proper pairs generalized to such graphs.
- We provide some bounds concerning the problem of partitioning and covering a graph with graphs with a bounded number of vertices and some specific structure (connected graphs, stars, cliques) – see Section 3.5.

In Chapter 4 we present exact exponential algorithms, using the framework first invented by Rossmanith [77].

- We describe the framework and present an exact algorithm for the generalized list T-coloring problem (Section 4.2.1) along with a detailed complexity analysis (Section 4.2.2).
- We show how to adapt the algorithm to obtain a refined algorithm for the $L(2,1)$-labeling problem (Section 4.3).

- We also adapt the algorithm for both the graph homomorphism problem and the locally injective graph homomorphism problem (Section 4.4).

The algorithms described in Chapter 4 require exponential memory. In Chapter 5 we focus on algorithms using polynomial space.

- We present an exact algorithm for the $L(2,1)$-labeling problem, using polynomial memory and its detailed complexity analysis (Section 5.1).
- We adapt this method to obtain an algorithm for the k-$L(2,1)$-labeling problem, which is faster than the general algorithm for $k \leq 31$ (Section 5.2).

In Chapter 6 we conclude the dissertation by pointing out some open problems and possible directions of further research.

Chapter 2

Preliminaries and basic definitions

By $\mathbb{N}$ we denote the set $\{1, 2, ..\}$ of natural numbers. We also define $\mathbb{N}_0 := \mathbb{N} \cup \{0\}$. For a set X and integer k, by $\binom{X}{k}$ we denote the family of all k-element subsets of X. By 2^X we mean the family of all subsets of X.

A pair $G = (V, E)$ is called a *graph* if V is a finite set of *vertices* and $E \subseteq \binom{V}{2}$ is the set of *edges*. For a graph $G = (V, E)$ we refer to V and E by $V(G)$ and $E(G)$, respectively. We consider only finite undirected graphs without multiple edges or loops.

The number of vertices of a graph G is called the *order* of G and denoted by $|G|$. In most cases we shall use the letter n to denote the order of currently considered graph. For two vertices u, v, the edge $\{u, v\}$ will be denoted shortly as uv.

By the *open neighborhood* of a vertex u in G we mean the set $\{v \colon uv \in E(G)\}$ and denote it by $N_G(u)$. The set $N_G[u] = N_G(u) \cup \{u\}$ denotes the *closed neighborhood* of u. The *open neighborhood* of a set X of vertices in G is denoted by $N_G(X) = \bigcup_{v \in X} N_G(v)$ and its *closed neighborhood* is denoted by $N_G[X] = N_G(X) \cup X$. The *degree* $\deg_G v$ of the vertex v is the number of its neighbors, i.e. $|N(v)|$.

A graph H is a *spanning subgraph* of G if $V(H) = V(G)$ and $E(H) \subseteq E(G)$. Let X be a subset of vertices of $G = (V, E)$ and let E' be a subset of edges. We denote the *subgraph of G induced by the vertices in X* by $G[X]$, i.e. $G[X] = (X, \{e \in E \colon e \subseteq X\})$. By $G[E']$ we denote the *subgraph induced by the set of edges E'*, i.e. $G[E'] = (\bigcup_{e \in E'} e, E')$.

By $G - X$ we denote the graph $(V \setminus X, \{e \in E \colon e \subseteq V \setminus X\})$. For X being a singleton (i.e. $X = \{x\}$), we shall write $G - x$ instead of $G - \{x\}$.

The distance $\mathrm{dist}_G(u, v)$ between two vertices u and v in a graph G is the length of a shortest path joining u and v. The *diameter* of the graph G, denoted by $\operatorname{diam}(G)$, is the maximum distance between vertices in G, i.e., $\operatorname{diam}(G) = \max_{u,v \in V(G)} \operatorname{dist}_G(u,v)$.

If the graph G is clear from the context, we shall omit the subscript G and simply write $N(v), \deg v, \operatorname{dist}(u,v)$ etc.

For a graph $G = (V,E)$, its *square graph* is the graph $G^2 = (V, \{uv \in \binom{V}{2} \colon \operatorname{dist}_G(u,v) \leq 2\})$. In an analogous way we define the *k-th power of* G (for any $k \geq 1$) as $G^k = (V, \{uv \in \binom{V}{2} \colon \operatorname{dist}_G(u,v) \leq k\})$. By $\overline{G}$ we denote the *complement* of G, i.e. the graph $(V, \{uv \in \binom{V}{2} \colon uv \notin E\})$.

The terminology and notation used in this thesis is standard for graph theory and can be found in e.g. a comprehensive book by Diestel [22].

Chapter 3

Combinatorial estimates

This chapter is purely graph-theoretic. We present here some upper and lower bounds for the maximum number of certain combinatorial objects in graphs. Some of the results in this chapter may seem artificial at a first glance. However, they will be used in the next sections to estimate the complexity of some algorithms.

Particular sections contain mostly joint results from the following papers:

Section 3.1:	a paper co-authored with K. Junosza-Szaniawski [57],
Section 3.2:	a paper co-authored with K. Junosza-Szaniawski [58],
Section 3.3:	a paper co-authored with K. Junosza-Szaniawski, J. Kratochvíl, M. Liedloff and P. Rossmanith [53],
Section 3.4:	a paper co-authored with K. Junosza-Szaniawski, J. Kratochvíl and M. Liedloff [54],
Section 3.5:	a paper co-authored with K. Junosza-Szaniawski, J. Kratochvíl, M. Liedloff and P. Rossmanith [53], and a preprint with K. Junosza-Szaniawski [59].

Definition 3.1. A *2-packing* in a graph G is a subset X of vertices, in which no two vertices are adjacent or have a common neighbor.

Sometimes 2-packings are called 2-independent sets or 2-stable sets. We can also see 2-packings as independent sets in the square of a graph. We shall use the following easy observations many times.

Observation 3.2. *If H is a subgraph of G and X is a 2-packing in G, then $X \cap V(H)$ is a 2-packing in H.*

Observation 3.3. *Let G be a graph with two vertices v_1, v_2 of degree 1, having a common neighbor v_3. Every 2-packing in G is also a 2-packing in the graph H obtained from G by removing the edge v_1v_3 and adding the edge v_1v_2 (see Figure 3.1).*

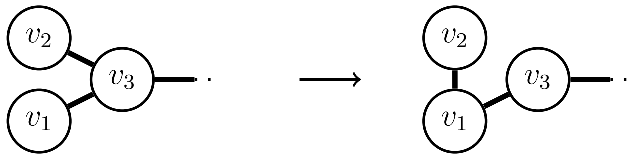


Figure 3.1: Transformation of two leaves with a common neighbor.

3.1 Maximum number of k-element 2-packings

Let $u_k(G)$ denote the number of k-element 2-packings in a graph G. By $u_k(n)$ we denote the maximum number of k-element 2-packings in a connected graph on n vertices. Havet *et al.* [45] showed the following upper bound for $u_k(n)$ (in fact, they only required that the graph does not have isolated vertices and we shall state their theorem in this form).

Theorem 3.4 (Havet *et al.* [45]). *The maximum number of k-element 2-packings in a graph with no isolated vertices does not exceed $\binom{n/2}{k}2^k$.*

Remark 3.5. Notice that in the theorem above we did not assume that n is even. In fact, the actual bound should be stated as $\binom{\lceil n/2 \rceil}{k}2^k$. However, as we are only interested in asymptotic behavior of such bounds, we shall by convention omit floors and ceilings, where it does not lead to a misunderstanding.

Observe that this bound it tight – it suffices to consider a graph consisting of $n/2$ disjoint edges. In this section we improve this bound for connected graphs and prove the following theorem.

Theorem 3.6 ([57]). *The maximum number of k-element 2-packings in a connected graph on n vertices does not exceed $\binom{n-k+1}{k}$. Moreover, all such 2-packings can be enumerated in time $\mathcal{O}^*\left(\binom{n-k+1}{k}\right)$, using polynomial space.*

Proof. Fix some k and let G be a connected graph with with $n \geq 3$ vertices and exactly $u_k(n)$ 2-packings. By Observations 3.2 and 3.3, we can assume that G is a tree (otherwise we can take a spanning tree of G) with no two leaves having a common neighbor. Let P be a longest path in G. Let v be an end-vertex of the path P and u be its neighbor on P. By Observation 3.3 we have that $\deg(u) = 2$ (recall that $n \geq 3$). We can partition the set of all 2-packings into two subsetes – one containing 2-packings X in which $v \notin X$ and the other one in which $v \in X$. If $v \notin X$, then X is a k-element 2-packing in $G - v$. On the other hand, if $v \in X$, then $X \setminus \{v\}$ is a $(k-1)$-element 2-packing in $G - \{v, u\}$ (note that the other neighbor of u on P also cannot belong to X, but we do not delete it to keep our graph connected). Since the graphs $G - v$ and $G - \{v, u\}$ are connected, we obtain the following recursion:

$$u_k(n) \leq \underbrace{u_k(n-1)}_{v \notin X} + \underbrace{u_{k-1}(n-2)}_{v \in X} \text{ for } n \geq 3 \text{ and } k \geq 1,$$

with the boundary conditions:

$$\begin{cases} u_0(n) = 1 \\ u_1(1) = 1 \\ u_1(2) = 2 \\ u_2(2) = 0 \\ u_k(n) = 0 & \text{for } k > n \end{cases}$$

It is easy to verify that this inequality implies that $u_k(n) \le \binom{n-k+1}{k}$.

Observe that the proof yields a simple branching algorithm for enumerating all k-element 2-packings, whose complexity is bounded by $\binom{n-k+1}{k} \cdot n^{\mathcal{O}(1)} = \mathcal{O}^*\left(\binom{n-k+1}{k}\right)$. $\square$

Now let us provide some lower bounds for $u_k(n)$. Consider a graph P_n, which is a path with n vertices. Notice that $u_k(P_n)$ is equal to the number of binary sequences of length n with exactly k ones, such that there are at least two zeros between every pair of ones. Observe that there are exactly $\binom{n-2k+2}{k}$ such sequences.

Now let n be odd and consider a graph S_n, obtained from a matching of size $\frac{n-1}{2}$ by adding one extra vertex and joining it with exactly one vertex from every edge of the matching (see Figure 3.2).

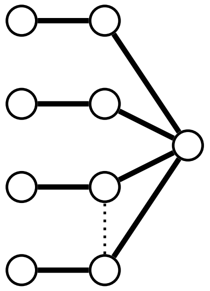


Figure 3.2: The graph S_n.

It is easy to see that:

$$u_k(S_n) = \begin{cases} n & \text{for } k = 1 \\ (k+1)\binom{\frac{n-1}{2}}{k} & \text{for } k \neq 1. \end{cases}$$

Observe that $u_2(P_n) > u_2(S_n)$ (for $n > 5$), while $u_{\frac{n-1}{2}}(S_n) > u_{\frac{n-1}{2}}(P_n) = 0$ (for $n > 7$). This gives us the following theorem.

Theorem 3.7 ([58]). *The maximum number of k-element 2-packings in a connected graph on n vertices is at least* $\max(\binom{n-2k+2}{k}, (k+1)\binom{\lfloor \frac{n-1}{2} \rfloor}{k})$.

3.2 Maximum number of 2-packings

Let $u(G)$ denote the number of 2-packings in a graph G. By $u(n)$ we denote the maximum number of 2-packings in a connected graph on n vertices.

3.2.1 General graphs

Clearly we have $u(n) \leq \sum_{k=0}^{n} u_k(n)$. When we add the bounds for u_k obtained in Theorem 3.6 over all possible values of k, we obtain the following corollary.

Corollary 3.8 ([57]). *The number of all 2-packings in a connected graph on n vertices does not exceed $Fib_{n+1} = \mathcal{O}^*((\frac{1+\sqrt{5}}{2})^n) = \mathcal{O}(1.6181^n)$*[†].

However, we can obtain a significantly better bound, using slightly more complicated branching rules. The following theorem is the main result of this section.

Theorem 3.9 ([58]). *The maximum number of 2-packings in a connected graph on n vertices is $u(n) = \mathcal{O}(1.5400^n)$. Moreover, all such 2-packings can be enumerated within this time bound, using polynomial space.*

Proof. Let G be a connected graph with $n \geq 3$ vertices and exactly $u(n)$ 2-packings. By Observations 3.2 and 3.3 we can assume that G is a tree with no two leaves having a common neighbor.

Let P be a longest path in G. Let v be an end-vertex of P, u be its neighbor on P, and c be a neighbor of u on P other that v (the third vertex on P). Recall that $\deg(u) = 2$ by Observation 3.3. Consider the following cases.

(A) Let $\deg(c) \leq 2$. We can partition the set of all 2-packings into two subsets: one containing 2-packings X in which $v \notin X$ and the other one in which $v \in X$ (see Figure 3.3).

If $v \in X$, then none of the vertices $\{u, c\}$ belongs to X. Since the graphs $G - v$ and $G - \{v, u, c\}$ are connected, we obtain the following recursive inequality:

$$u(n) = u(G) \leq u(G - v) + u(G - \{v, u, c\}) \leq u(n-1) + u(n-3) \quad (3.1)$$

[†] Fib_n denotes the n-th Fibonacci number.

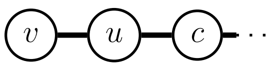
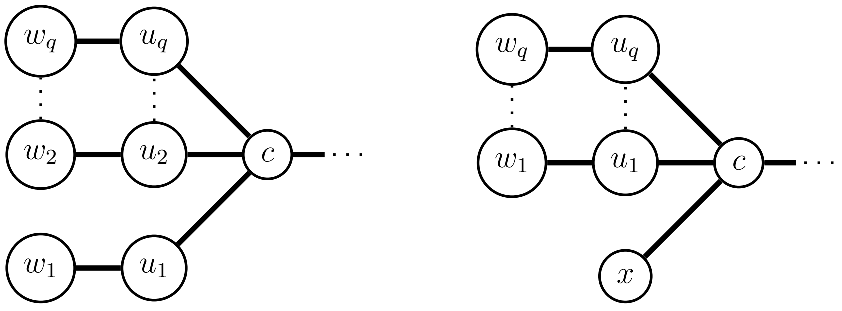


Figure 3.3: The case (A) with $\deg(c) \leq 2$.

(B) Let $\deg(c) > 2$. We denote by d the neighbor of c on P other than u. Let $U = N_G(c) \setminus \{d\}$ and let $W = N_G(U) \setminus \{c\}$. Then all vertices in W are leaves of G (since otherwise P is not the longest path) and all vertices in U except at most one (which may be a leaf) are of degree 2, because by Observation 3.3 there are no two leaves with a common neighbor. Let q be the number of vertices in U, which are not leaves. One of the following two subcases occurs:

(B0) No vertex from U is a leaf in G (see Figure 3.4(a), $q \geq 2$)).

(B1) There exists a vertex $x \in U$ which is a leaf in G (see Figure 3.4(b), $q \geq 1$).



(a) The case (B0) with $\deg(c) > 2$ and no neighbor of c being a leaf.

(b) The case (B1) with $\deg(c) > 2$ and one neighbor of c being a leaf.

Figure 3.4: Subcases (B0) and (B1).

We can partition the set of all 2-packings into two subsets: one containing 2-packings with non-empty intersection with $W \cup U$ and the other containing the remaining 2-packings. Let X be a 2-packing in G. If $X \cap (W \cup U) = \emptyset$, then X is just a 2-packing in $G - (W \cup U)$. If $\widehat{X} := X \cap (W \cup U) \neq \emptyset$, then $\widehat{X}$ must be a 2-packing in $G[W \cup U \cup \{c\}]$. Notice that the number of non-empty 2-packings $\widehat{X}$ in $G[W \cup U \cup \{c\}]$, such that $c \notin \widehat{X}$ is equal to:

1. $\underbrace{2^q - 1}_{X\cap U=\emptyset} + \underbrace{q \cdot 2^{q-1}}_{X\cap U\neq\emptyset} = (q+2)2^{q-1} - 1$ for $q \geq 2$ in the case (B0).

2. $\underbrace{2^q - 1}_{X\cap U=\emptyset} + \underbrace{2^q}_{x\in X} + \underbrace{q \cdot 2^{q-1}}_{X\cap(U\setminus\{x\})\neq\emptyset} = (q+4)2^{q-1} - 1$ for $q \geq 1$ in the case (B1).

Since the graphs $G - (W \cup U)$ and $G - (W \cup U \cup \{c\})$ are connected, we obtain that $u(n) = u(G) \leq u(G - (W \cup U)) + u(G - (W \cup U \cup \{c\}))$. Depending on the case, this gives us the following recursions:

$$\begin{aligned} u(n) \leq& u(n-2q) + \left((q+2)2^{q-1} - 1\right) u(n-2q-1) \\ u(n) \leq& u(n-2q-1) + \left((q+4)2^{q-1} - 1\right) u(n-2q-2). \end{aligned}$$

We shall prove by induction on n that for $n \geq 0$ the following holds:

$$u(n) \leq 2 \cdot \tau^n, \tag{3.2}$$

where $\tau = 1.5399..$ is the positive root of the equation $x^7 = x + 19$.

It is easy to observe that the inequality (3.2) holds for $n \leq 2$. Now assume that the inequality holds for all values smaller than n.

Case (A): $u(n) \leq u(n-1) + u(n-3) \leq 2\tau^{n-1} + 2\tau^{n-3} = 2(\tau^2+1)\tau^{n-3} < 2 \cdot \tau^3 \cdot \tau^{n-3} = 2 \cdot \tau^n$

Case (B0):

$$\begin{aligned} u(n) \leq& u(n-2q) + \left((q+2)2^{q-1} - 1\right) u(n-2q-1) \\ \leq& 2 \cdot \tau^{n-2q} + 2\left((q+2)2^{q-1} - 1\right) \tau^{n-2q-1} \\ \leq& 2 \cdot \tau^n \left(\tau^{-2q} + ((q+2)2^{q-1} - 1) \cdot \tau^{-2q-1}\right) \\ =& 2 \cdot \tau^n \cdot \left(\frac{1}{\tau^{2q}} + \frac{(q+2)2^{q-1} - 1}{\tau^{2q+1}}\right) \end{aligned}$$

We chose τ to be the minimum value such that $\left(\frac{1}{\tau^{2q}} + \frac{(q+2)2^{q-1}-1}{\tau^{2q+1}}\right) \leq 1$ for every $q \geq 2$. By a standard reasoning one can prove that this minimum is achieved by the root of the equation $\tau^7 = \tau + 19$. For this value of τ consider the function $h_0(q) = \frac{1}{\tau^{2q}} + \frac{(q+2)2^{q-1}-1}{\tau^{2q+1}}$. It is easy to verify that $h_0(q)$ is decreasing for all $q \geq 3$. Moreover, we have $h_0(2) \approx 0.9860 < 1$ and $h_0(3) = 1$. Hence $u(n) \leq 2 \cdot \tau^n \left(\frac{1}{\tau^{2q}} + \frac{(q+2)2^{q-1}-1}{\tau^{2q+1}}\right) \leq 2 \cdot \tau^n$.

The case (B0) for $q = 3$ determines the complexity of the whole algorithm.

Case (B1):

$$\begin{aligned}u(n) \leq & u(n-2q-1) + \left((q+4)2^{q-1}-1\right) u(n-2q-2) \\ \leq & 2\cdot\tau^{n-2q-1} + 2\left((q+4)2^{q-1}-1\right)\tau^{n-2q-2} \\ = & 2\cdot\tau^{n}\left(\tau^{-2q-1} + ((q+4)2^{q-1}-1)\cdot\tau^{-2q-2}\right) \\ = & 2\cdot\tau^{n}\cdot\left(\frac{1}{\tau^{2q+1}} + \frac{(q+4)2^{q-1}-1}{\tau^{2q+2}}\right)\end{aligned}$$

Since the function $h_1(q) = \frac{1}{\tau^{2q+1}} + \frac{(q+4)2^{q-1}-1}{\tau^{2q+2}}$ is decreasing for all $q \geq 1$ and $h_1(1) \approx 0.9852 < 1$, we obtain: $u(n) \leq 2\cdot\tau^{n}\left(\frac{1}{\tau^{2q+1}} + \frac{(q+4)2^{q-1}-1}{\tau^{2q+2}}\right) \leq 2\cdot\tau^{n}$.

We have shown that regardless of the structure of G, the function $2\cdot\tau^{n}$ is an upper bound on the number of 2-packings in G. Hence $u(n) = \mathcal{O}(\tau^{n}) = \mathcal{O}(1.5400^{n})$. Again, our proof is constructive and can be easily transformed into an algorithm enumerating all 2-packings. □

Now we shall show a lower bound for the value of $u(n)$. Let us consider the graphs A_k, B_k, C_k and D_k shown in Figure 3.5. Observe that D_k is isomorphic to A_k and the reason to introduce it as a separate graph is only technical.

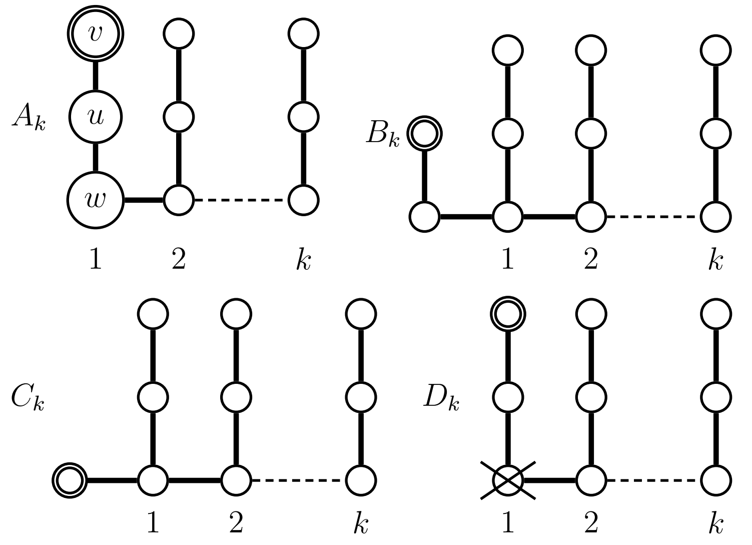


Figure 3.5: Graphs showing the lower bound on $u(n)$.

Let a_k, b_k and c_k denote the number of 2-packings in the graphs A_k, B_k and C_k, respectively. Let d_k denote the number of 2-packings in the graph D_k, which do not contain the crossed out vertex.

Consider the graph A_k and let v, u, w denote the marked vertex, its neighbor and the neighbor of u (other than v), respectively (see Figure 3.5). We can partition the set of all 2-packings in A_k into two subsets: those containing v and the remaining ones. If a 2-packing X contains v, then it contains no vertex from $\{u, w\}$. Moreover, $X' := X \setminus \{v\}$ is a 2-packing in $A_k - \{v, u, w\}$, which is isomorphic to A_{k-1}. Therefore X' can be chosen in a_{k-1} ways. On the other hand, if a 2-packing X does not contain v, then it is a 2-packing in $A_k - v$ (which is isomorphic to B_{k-1}), and thus can be chosen in b_{k-1} ways. So we obtain that $a_k = a_{k-1} + b_{k-1}$.

By similar analysis of the remaining three graphs, we obtain the following system of recursions:

$$\begin{cases} a_k = b_{k-1} + a_{k-1} \\ b_k = c_k + d_k \\ c_k = a_k + 2d_{k-1} \\ d_k = 2a_{k-1} + d_{k-1}. \end{cases}$$

By transforming it, we obtain $a_k = 3a_{k-1} + 4a_{k-3}$. This equation has a solution $a_k = \Theta(\tau^k)$, where $\tau \approx 3.3553$ is the root of $x^3 = 3x^2 + 4$. Since $k = n/3$, the graph A_k contains $a_k = \Theta(\tau^{n/3})$ 2-packings (where $\tau^{1/3} \approx 1.4970$). This gives us the following theorem.

Theorem 3.10 ([58]). *The maximum number of* 2*-packings in a connected graph with n vertices is $\Omega(1.4970^n)$.*

3.2.2 Selected graph classes

In this section we turn to special classes of graphs.

Regular graphs

Let us start by mentioning an unpublished result by Alon [3].

Alon [4] and then Kahn [60] and Zhao [89] considered the problem of bounding the maximum number of independent sets in a regular graph. After a series of papers, the following bound has been established.

Theorem 3.11 (Zhao [89])**.** *The maximum number of independent sets in a d-regular graph with n vertices is at most $(2^{d+1}-1)^{\frac{n}{2d}}$.*

To see that this bound is tight, one has to consider a graph consisting of disjoint copies of the complete bipartite graph $K_{d,d}$.

Motivated by these results, Łuczak [71] stated a problem of determining the maximum number of 2-packings in a d-regular graph. Intuitively, the extremal graph should consist of disjoint copies of K_{d+1}. The number of 2-packings in such a graph is clearly $(d+2)^{\frac{n}{d+1}}$. This intuition has been confirmed by Alon, using the following version of Shearer's lemma.

Lemma 3.12 (Chung *et al.* [17])**.** *Let N be a finite set and let $\mathcal{F}$ be a family of its subsets. Let $\mathcal{S} = \{S_1, S_2, \ldots, S_m\}$ be a family of subsets of N, such that every element of N belongs to at least k members of $\mathcal{S}$. For each $i = 1, 2, .., m$ define $\mathcal{F}_i = \{F \cap S_i \colon F \in \mathcal{F}\}$. Then*

$$|\mathcal{F}|^k \le \prod_{i=1}^{m} |\mathcal{F}_i|.$$

Theorem 3.13 (Alon [3])**.** *The maximum number of 2-packings in a d-regular graph with n vertices is $(d+2)^{\frac{n}{d+1}}$.*

Proof. Let $G = (V, E)$ be a d-regular graph with n vertices. By $\mathcal{F}$ we denote the family of all 2-packings in G. Define $\mathcal{S} = \{N[v] \colon v \in V\}$. Note that every vertex from G belongs to exactly $d+1$ sets from $\mathcal{S}$. Moreover, no 2-packing from $\mathcal{F}$ contains more than one vertex from each closed neighborhood $N[v]$ (otherwise it would contain two vertices in distance at most 2 from each other). Therefore $|\mathcal{F}_v| = |\{F \cap N(v) \colon F \in \mathcal{F}\}| = d+2$ (each intersection is either empty or has exactly one element). From Lemma 3.12 we have:

$$\begin{aligned} |\mathcal{F}|^{d+1} \le &\prod_{v \in V} |\mathcal{F}_i| = (d+2)^n \\ |\mathcal{F}| \le &(d+2)^{\frac{n}{d+1}}. \end{aligned}$$

□

It is interesting to note a different behavior of power bases in the bounds for the maximum number of independent sets and the maximum number of 2-packings in a d-regular graph for large d. Namely, $\lim_{d\to\infty} \sqrt[n]{(2^{d+1}-1)^{\frac{n}{2d}}}$ is $\sqrt{2}$, while $\lim_{d\to\infty} \sqrt[n]{(d+2)^{\frac{n}{d+1}}}$ is 1.

Claw-free graphs

The complete bipartite graph $K_{1,3}$ is also called a *claw*. A graph is called *claw-free* if it contains no copy of $K_{1,3}$ as an induced subgraph. Claw-free graphs form a well-studied class of graphs. We refer the reader to the survey by Branstädt *et al.* [11] for properties of this graph class, which are not listed here.

In this section we show an asymptotically tight bound for $u_{cf}(n)$, being the maximum number of 2-packings in a connected claw-free graph with n vertices. The idea of the proof is similar to the proof of Theorem 3.9, however it involves an additional analysis of the structure of claw-free graphs. Let us start with the following simple observation.

Observation 3.14. *If the diameter of G is at most 2, then at most one of the vertices in $V(G)$ can belong to a 2-packing. Hence $u(G) \leq n+1$.*

The following lemma, in which we investigate the structure of claw-free graphs, is the key ingredient of our proof for the bound on $u_{cf}(n)$.

Lemma 3.15. *Let G be a claw-free graph with $\operatorname{diam}(G) \geq 3$ and let T be its spanning tree with the largest diameter. Let P be a longest path in T, v_1 be an end-vertex of P, u_1 be its neighbor on P, w be the neighbor of u_1 on P other that v_1 (the third vertex on P) and x be a neighbor of w on P other that u_1 (the fourth vertex on P). One of the following cases occurs:*

Case (C1) $\deg_T u_1 = 2$ *and* $\deg_T w = 2$ *(see Figure 3.6).*

Case (C2) $\deg_T u_1 = 2$ *and* $\deg_T w = 3$*. The T-neighbor y_1 of w other than u_1 and x is a leaf in T. Moreover, xu_1 is an edge in G (see Figure 3.7).*

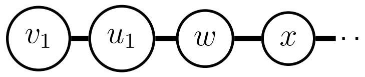


Figure 3.6: Case (C1)

Proof. We shall proceed is several steps by proving a few claims.
Claim 1. For every T-neighbor u of w other than x we have $\deg_T u \leq 2$.

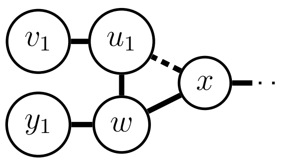


Figure 3.7: Case (C2)

Proof of Claim 1. Suppose that there exists a neighbor u of w with $\deg_T u > 2$. Let v and v' be T-neighbors of u other than w. Since P is a longest path in T, both v and v' are leaves in T (see Figure 3.8).

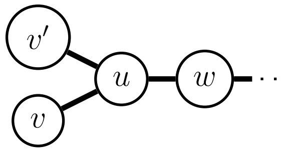


Figure 3.8: There exist $u \in N_T(w) \setminus \{x\}$ with $\deg u > 2$.

Notice that since G is claw-free, there exists in G at least one of the edges vw, $v'w$ or vv'.

Subcase 1.1. $yw \in E(G)$ for $y \in \{v, v'\}$ (without loss of generality let $y = v$).

Consider the spanning tree $T' := (V(G), E(T) \cup \{vw\} \setminus \{uw\})$ (see the Figure 3.9). Note that the diameter of T' is larger than the diameter of T, which contradicts to the choice of T.

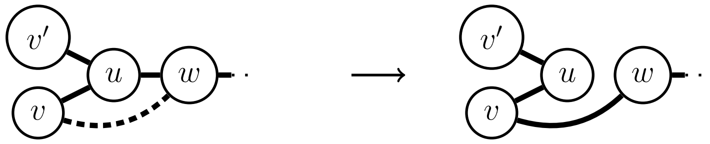


Figure 3.9: Subcase 1.1

Subcase 1.2 $vv' \in E(G)$.

Consider the spanning tree $T' := (V(G), E(T) \cup \{vv'\} \setminus \{uv'\})$. Again $\operatorname{diam}(T') > \operatorname{diam}(T)$ and we obtain a contradiction (see Figure 3.10).

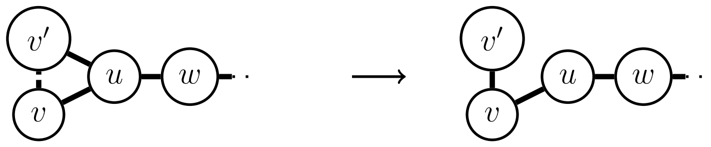


Figure 3.10: Subcase 1.2

■

Let $y_1, .., y_q$ be the T-neighbors of w other than x, which are leaves in T. Let $u_1, .., u_p$ be the neighbors of w other than x, which have degree 2 in T. Moreover, for any $i \in \{1, 2, .., p\}$, by v_i we denote the neighbor of u_i other than w (see Figure 3.11).

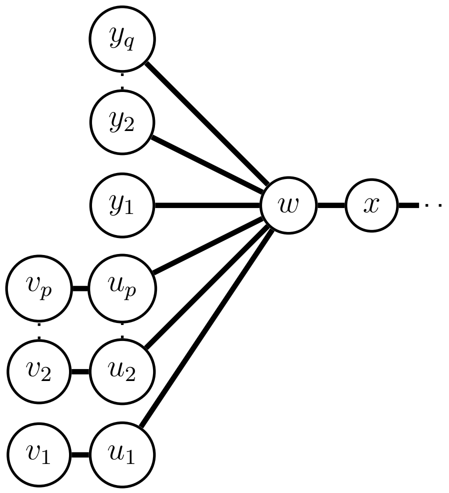


Figure 3.11: Neighborhood of w in T.

Claim 2. We have $p = 1$.

Proof of Claim 2. By the definition of u_1 we have $p \geq 1$. Suppose that $p \geq 2$ and consider the vertices u_1, u_2. Since G is claw-free, at least one of the edges u_1x, u_2x or u_1, u_2 exists in G.

Subcase 2.1. $u_ix \in E(G)$ for some $i \in \{1, 2\}$ (without loss of generality let $i = 1$).

Consider the spanning tree $T' := (V(G), E(T) \cup \{u_1x\} \setminus \{xw\})$. Since $\operatorname{diam}(T') > \operatorname{diam}(T)$, we obtain a contradiction (see Figure 3.12).

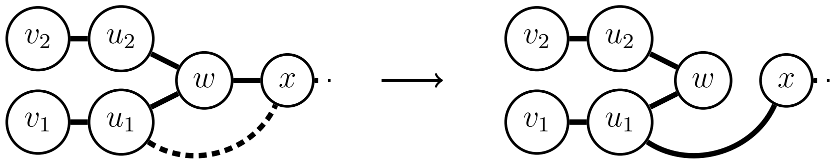


Figure 3.12: Subcase 2.1

Subcase 2.2. $u_1u_2 \in E(G)$.

Consider the spanning tree $T' := (V(G), E(T) \cup \{u_1u_2\} \setminus \{u_2w\})$. Since $\operatorname{diam}(T') > \operatorname{diam}(T)$, we obtain a contradiction (see Figure 3.13).

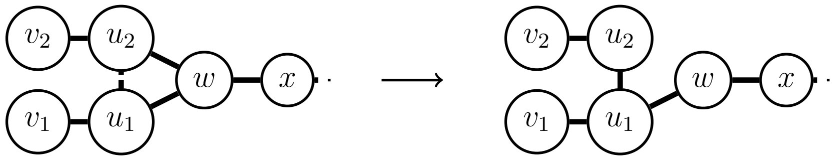


Figure 3.13: Subcase 2.2

■

Claim 3. We have $q \leq 1$.

Proof of Claim 3. Suppose that $q \geq 2$ and consider the vertices y_1, y_2. Since G is claw-free, at least one of the edges y_1x, y_2x or y_1, y_2 exists in G.

Subcase 3.1. $y_ix \in E(G)$ for some $i \in \{1, 2\}$.

We proceed in a way analogous to the Subcase 2.1, obtaining a spanning tree with the diameter larger than $\operatorname{diam}(T)$ – a contradiction.

Subcase 3.2. $y_1y_2 \in E(G)$.

Since G is claw-free, there also exists in G at least one of the edges y_1u_1,y_1x or u_1x.

If $y_1u_1 \in E(G)$, then consider the tree $T' := (V(G), E(T) \cup \{y_1y_2, y_1u_1\} \setminus \{wu_1, wy_1\}$. If $y_1x \in E(G)$, then consider the tree $T' := (V(G), E(T) \cup$

$\{y_1y_2, y_1x\}\setminus\{wy_1, wx\}$. If $u_1x \in E(G)$, then consider the tree $T' := (V(G), E(T)\cup \{y_1y_2, u_1x\} \setminus \{wy_1, wx\}$ (see Figure 3.14). In all cases $\text{diam}(T') > \text{diam}(T)$, which contradicts to the choice of T.

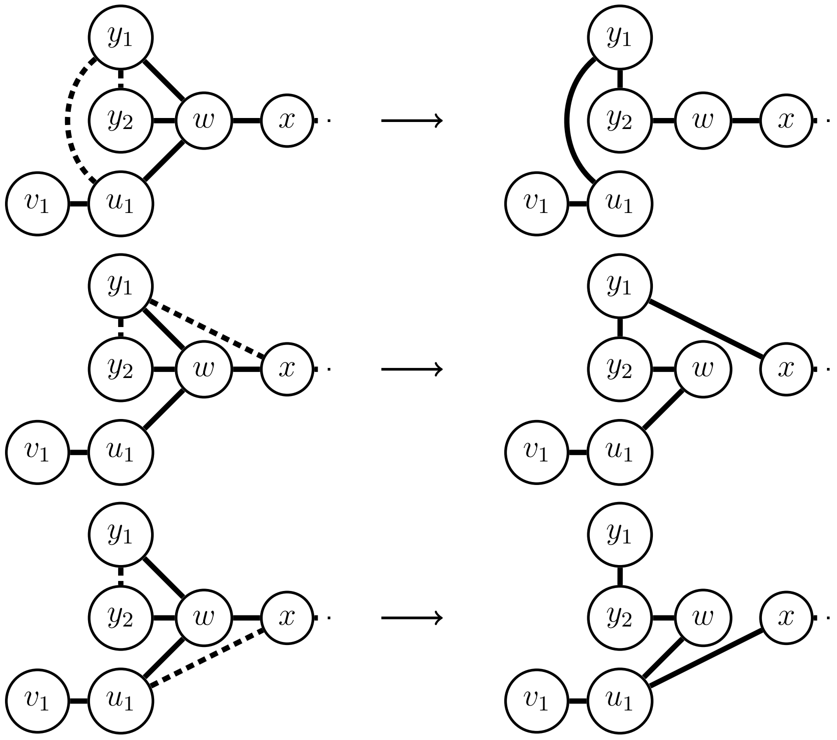


Figure 3.14: Subcase 3.2

■

Claim 4. If $q = 1$, then the edge xu_1 exists in G.

Proof of Claim 4. Suppose the contrary, so $xu_1 \notin E(G)$. Since G is claw-free, at least one of the edges xy_1 or y_1u_1 exists in G. If $xy_1 \in E(G)$, then we proceed in a way analogous to the one in Subcase 2.1. In the latter case, we proceed as in Subcase 2.2. In both cases we finally obtain a spanning tree T' with $\text{diam}(T') > \text{diam}(T)$, which is a contradiction. ■

This completes the proof of Lemma 3.15. □

Now we are ready to present the main theorem of this section.

Theorem 3.16. *The value $u_{cf}(n)$ is $\Theta(\tau^n)$, where $\tau \approx 1.4656$. Moreover, all 2-packings in a connected claw-free graph with n vertices can be enumerated in time $\mathcal{O}(1.4656^n)$, using polynomial space.*

Proof. We shall prove by induction on n that for $n \geq 0$ the following holds:

$$u_{cf}(n) \leq 2 \cdot \tau^n \tag{3.3}$$

where $\tau = 1.4655..$ is the positive root of the equation $x^3 = x^2 + 1$.

Clearly the theorem is true for all $n < 3$. Assume that $n \geq 3$ and let G be a connected claw-free graph on n vertices with $u_{cf}(n)$ 2-packings. If $\operatorname{diam}(G) \leq 2$, then by the Observation 3.14 we have $u_{cf}(n) \leq n + 1 \leq 2 \cdot \tau^n$ for $n \geq 3$. Thus we can assume that $\operatorname{diam}(G) \geq 3$. Let T be a spanning tree of G with the largest diameter. Define P, v_1, u_1, y_1, w and x as in Lemma 3.15. By Lemma 3.15 one of the cases (C1) or (C2) occurs (see Figures 3.6 and 3.7).

Case (C1). In this case we can partition the set of all 2-packings into two disjoint subsets: one subset containing 2-packings X in which $v_1 \notin X$ and the other subset with 2-packings X in which $v_1 \in X$. If $v_1 \in X$, then none of the vertices $\{u_1, w\}$ can belong to X. Since the graphs $G - v_1$ and $G - \{v_1, u_1, w\}$ are connected and claw-free, we obtain the following recursion:

$$\begin{aligned} u_{cf}(n) =& u(G) \leq u(G - v_1) + u(G - \{v_1, u_1, w\}) \\ \leq& u_{cf}(n-1) + u_{cf}(n-3). \end{aligned}$$

Using induction hypothesis we obtain that:

$$u_{cf}(n) \leq u_{cf}(n-1) + u_{cf}(n-3) \leq 2 \cdot \tau^{n-3}(\tau^2 + 1) = 2 \cdot \tau^n.$$

The Case (C1) is the worst one and determines the complexity of the algorithm.

Case (C2). In this case we can partition the set of all 2-packings into two disjoint subsets: one with 2-packings containing no vertex from $\{v_1, y_1\}$ and the other one with the remaining 2-packings. Let X be a 2-packing in G.

If $\{v_1, y_1\} \cap X \neq \emptyset$, then none of the vertices in $\{u_1, w, x\}$ can belong to X (recall that $xu_1 \in E(G)$). On the other hand $\{v_1, y_1\} \cap X$ can be equal to $\{v_1\}$

or $\{y_1\}$ or $\{v_1, y_1\}$. Since the graphs $G - \{v_1, y_1\}$ and $G - \{v_1, y_1, u_1, w, x\}$ are connected and claw-free, we obtain the following recursion:

$$\begin{aligned} u_{cf}(n) =& u(G) \leq u(G - \{v_1, y_1\}) + 3u(G - \{v_1, y_1, u_1, w, x\}) \\ \leq& u_{cf}(n-2) + 3u_{cf}(n-5). \end{aligned}$$

By the induction hypothesis we obtain that:

$$u_{cf}(n) \leq u_{cf}(n-2) + 3u_{cf}(n-5) \leq$$
$$2\tau^{n-2} + 6\tau^{n-5} \leq 2 \cdot \tau^n \left(\frac{1}{\tau^2} + 3\frac{1}{\tau^5}\right)$$

It is easy to verify that $\left(\frac{1}{\tau^2} + 3\frac{1}{\tau^5}\right) < 1$ and thus $u_{cf}(n) \leq 2 \cdot \tau^n$. Thus we have shown that $u_{cf}(n) = \mathcal{O}(\tau^n)$.

On the other hand, a path is claw-free and there are exactly $\Theta(\tau^n)$ 2-packings in P_n. Again, the proof is constructive and can be easily transformed into an algorithm for enumerating 2-packings. □

Graphs with small dominating set

In this section we investigate a class of graphs with minimum dominating set of size at most r (let us call them *r-dominated*). Let us start by recalling the definition of a dominating set. Given a graph $G = (V, E)$, a subset $D \subseteq V$ is called *a dominating set* if each vertex of $V \setminus D$ has at least one neighbor in D. In this section, graphs do not have to be connected. Assume that D is a dominating set in a given graph G and $|D| \leq r$. Then the set of vertices of G can be easily partitioned into r disjoint stars with centers in vertices of D. If some vertex is dominated by more than one vertex from D, we include it to only one, arbitrarily chosen, star. Let $S(v)$ for $v \in D$ denote the set of vertices of a star with the center in v.

Proposition 3.17 ([53])**.** *There are at most $\binom{r}{k}\left(\frac{n}{r}\right)^k$ 2-packings of size k in an r-dominated graph G with n vertices. Moreover, if a dominating set of G with at most r vertices is given, then we can enumerate all k-element 2-packings in G in time $\mathcal{O}^*\left(\binom{r}{k}\left(\frac{n}{r}\right)^k\right)$.*

Proof. Notice that at most one vertex from every star belongs to a 2-packing. First we can choose k vertices $w_1, w_2, .., w_k$ from D. Then we can choose the vertices for the 2-packing in $\prod_{i=1}^{k} |S(w_i)|$ ways. It is easy to verify that this product is maximized if all factors are equal.

Hence the number of sets can be bounded by

$$\sum_{\substack{\text{distinct} \\ w_1, w_2, .., w_k \in D}} \prod_{i=1}^{k} |S(w_i)| \leq \binom{r}{k} \left(\frac{n}{r}\right)^k .$$

□

If we consider a graph consisting of r disjoint stars, each with the same number of vertices, we notice that the bound presented above is tight.

Let us mention that, as it was shown by Fomin *et al.* [36], finding a minimum dominating set in a graph on n vertices can be done in time $\mathcal{O}(1.5263^n)$ and polynomial space. Using some additional memorization, the authors were also able to speed the algorithm up to $\mathcal{O}(1.5137^n)$, but the exponential space is needed in this case.

If we add up the values of upper bounds from Proposition 3.17 over all possible k's, we obtain the following corollary.

Corollary 3.18. *There are at most* $\sum_{k=0}^{r} \binom{r}{k} (\frac{n}{r})^k = (1 + \frac{n}{r})^r$ *2-packings in an* r*-dominated graph with* n *vertices.*

A simple calculation shows that for $\frac{n}{r} > 3.4657$ the bound from Corollary 3.18 is better than the bound from Corollary 3.9. Observe that power base in the bound in Corollary 3.18 converges to 1 as $\frac{n}{r}$ grows (more precisely, $\lim_{n/r \to \infty} \sqrt[n]{\left(1 + \frac{n}{r}\right)^r} = 1$).

3.3 Proper pairs

A pair of disjoint subsets (X, Y) of a vertex set of a graph G is called a *proper pair* if and only if X is a 2-packing. The number of proper pairs in G will be denoted by $pp(G)$ and by the definition, we have

$$pp(G) = \sum_{k=0}^{n} u_k(G) \cdot 2^{n-k}. \tag{3.4}$$

Finally, we define

$$pp(n) = \max\{pp(G) \colon G \text{ is a connected graph with } n \text{ vertices}\}.$$

The value of $pp(n)$ will be needed to estimate the complexity of an algorithm for $L(2,1)$-labeling of graphs, described in Section 4.3.

Clearly we can use the bound for $u_k(n)$ from Theorem 3.6, obtaining that $pp(n) \leq \sum_{k=0}^{n} u_k(n) \cdot 2^{n-k} \leq \binom{n-k+1}{k} \cdot 2^{n-k} = \mathcal{O}^*\left((1+\sqrt{3})^n\right) = \mathcal{O}(2.7321^n)$. However, we can obtain a significantly better bound by enumerating all proper pairs similarly as we enumerated all 2-packings.

In general, when enumerating all 2-packings X in a graph G, we choose some subset of vertices A to be included in X, and some subset of vertices B (disjoint with A) not belonging to X. Then we proceeded recursively with the graph $G - (A \cup B)$. Now, every vertex from B can be contained in Y or not. Therefore, in most cases, we just include an additional multiplicative factor of $2^{|B|}$ in all recursions corresponding to the branching rules.

Since all proofs in this section are similar to the proofs of corresponding theorems in Section 3.2, we just provide the sketches of proofs, pointing out the differences.

3.3.1 General graphs

In this section we show an upper and a lower bound for $pp(n)$.

Theorem 3.19 ([53]). *The value of $pp(n)$ is bounded above by $\mathcal{O}(2.64877^n)$. All proper pairs in a connected graph with n vertices can also be enumerated within this time bound, using polynomial space.*

Proof. The proof is analogical to the proof of Theorem 3.9. Again we obtain cases (A) – see Figure 3.3, and (B) with subcases (B0) and (B1) – see Figure 3.4.

Case (A): Here we obtain the following recursion:

$$pp(G) \leq \underbrace{2\,pp(G-v)}_{v \notin X} + \underbrace{4\,pp(G-\{v,u,c\})}_{v \in X}$$

and hence

$$pp(n) \leq 2\,pp(n-1) + 4\,pp(n-3). \tag{3.5}$$

Case (B): In this case we obtain the following:

$$pp(G) \leq \underbrace{2^{|W \cup U|} pp(G-(W \cup U))}_{(W \cup U) \cap X = \emptyset} + \underbrace{\alpha \cdot pp(G-(W \cup U \cup \{c\}))}_{(W \cup U) \cap X \neq \emptyset},$$

where α denotes the number of proper pairs $(\widehat{X}, \widehat{Y})$ in $G[W \cup U \cup \{c\}]$, such that $\widehat{X} \neq \emptyset$ and $c \notin \widehat{X}$. Note that each of the vertices in $(W \cup U \cup \{c\}) \setminus \widehat{X}$ can be in $\widehat{Y}$ or outside $\widehat{X} \cup \widehat{Y}$. Therefore α is equal to:

1. $\underbrace{(3^q - 2^q)2^{q+1}}_{\widehat{X} \cap U = \emptyset} + \underbrace{q \cdot 3^{q-1}2^{q+1}}_{\widehat{X} \cap U \neq \emptyset} = 3^{q-1}2^{q+1}(3+q) - 2^{2q+1}$ for $q \geq 2$ in the case (B0).

2. $\underbrace{(3^q - 2^q)2^{q+2}}_{\widehat{X} \cap U = \emptyset} + \underbrace{q \cdot 3^{q-1}2^{q+2}}_{\widehat{X} \cap (U \setminus \{x\}) \neq \emptyset} + \underbrace{3^q 2^{q+1}}_{\widehat{X} \cap U = \{x\}} = 3^{q-1}2^{q+1}(9+2q) - 2^{2q+2}$ for $q \geq 1$ in the case (B1).

This gives us the following recursive formulae:

$$pp(n) \leq 2^{2q}\,pp(n-2q) + (3^{q-1}2^{q+1}(3+q) - 2^{2q+1})\,pp(n-2q-1) \tag{3.6}$$

(for $q \geq 2$) in case (B0) and

$$pp(n) \leq 2^{2q+1}\,pp(n-2q-1) + (3^{q-1}2^{q+1}(9+2q) - 2^{2q+2})\,pp(n-2q-2) \tag{3.7}$$

(for $q \geq 1$) in case (B1).

Using methods similar to the ones in the proof of Theorem 3.9, we observe that the case (B0) for $q = 2$ is the worst one. It is described by the recursion $pp(n) \leq 16\,pp(n-4) + 88\,pp(n-5)$. Thus we obtain that the inequalities (3.5), (3.6) and (3.7) are satisfied by $pp(n) \leq 2 \cdot \tau^n$, where $\tau = 2.64876..$ is the positive root of the equation $x^5 = 16x + 88$. $\square$

For the lower bound, again consider the graphs A_k, B_k, C_k, D_k in Figure 3.5. Let a'_k, b'_k and c'_k denote the number of proper pairs in the graphs A_k, B_k and C_k, respectively. Let d'_k denote the number of such proper pairs (X, Y) in the graph D_k, in which the 2-packing X does not contain the crossed out vertex.

Consider the graph A_k and let v, u, w denote the marked vertex, its neighbor and the neighbor of u (other than v), respectively (again see Figure 3.5). We can partition the set of all proper pairs in A_k into two subsets: one containing proper pairs (X, Y) in which $v \in X$ and the other one with the remaining proper pairs. Consider a proper pair (X, Y). If a 2-packing X contains v, then it contains no vertex from $\{u, w\}$. However, each of those vertices may or may not belong to Y. Moreover, $(X', Y') := (X \setminus \{v\}, Y \setminus \{u, w\})$ is a proper pair in $A_k - \{v, u, w\}$, which is isomorphic to A_{k-1}. Therefore (X', Y') can be chosen in a'_{k-1} ways. On the other hand, if a 2-packing X does not contain v, then it is a 2-packing in $A_k - v$ (which is isomorphic to B_{k-1}). Also v may or may not belong to Y. So we obtain that $a'_k = 4a'_{k-1} + 2b'_{k-1}$.

By similar analysis of the remaining three graphs, we obtain the following system of recursions:

$$\begin{cases} a'_k = 2b'_{k-1} + 4a'_{k-1} \\ b'_k = 2c'_k + 2d'_k \\ c'_k = 2a'_k + 12d'_{k-1} \\ d'_k = 4d'_{k-1} + 12a'_{k-1}. \end{cases}$$

Solving this system we obtain the recursive formula $a'_k = 16a'_{k-1} + 576a'_{k-3}$ and therefore $a'_k = \Theta(\tau^k)$, where $\tau \approx 17.8149$ is the positive solution of the equation $x^3 = 16x^2 + 576$. Since $k = n/3$, the graph A_k contains $a'_k = \Theta(\tau^{n/3})$ proper pairs and $\tau^{1/3} \approx 2.6117$. Hence we obtain the following corollary.

Corollary 3.20 ([53]). *The maximum number of proper pairs in a connected graph is $\Omega(2.6117^n)$.*

3.3.2 Selected graph classes

Claw-free graphs

In this section we investigate the value $pp_{cf}(n)$, being the maximum number of proper pairs in a claw-free graph on n vertices. Again, the idea of the proof of the main theorem is very similar to the one of Theorem 3.16.

Theorem 3.21 ([53]). *The value of $pp_{cf}(n)$ is $\Theta(\tau^n)$, where $\tau \approx 2.59431$. All proper pairs in a connected claw-free graph with n vertices can also be enumerated in time $\mathcal{O}(2.5944^n)$, using polynomial space.*

Proof. It is easy to verify that the theorem is true for all graphs with diameter at most 2. Let G be a connected claw-free graph with $n \geq 3$ vertices and exactly $pp_{cf}(n)$ proper pairs and let T be its spanning tree with the largest diameter. By Lemma 3.15 one of the cases (C1) or (C2) occurs (we define P, v_1, u_1, y_1, w and x as in Lemma 3.15) – see Figures 3.6 and 3.7.

Case (C1). We obtain the following recursion:

$$\begin{aligned} pp_{cf}(n) =& pp(G) \leq \underbrace{2\,pp(G - v_1)}_{v_1 \notin X} + \underbrace{4\,pp(G - \{v_1, u_1, w\})}_{v_1 \in X} \\ \leq& 2\,pp_{cf}(n-1) + 4\,pp_{cf}(n-3). \end{aligned} \tag{3.8}$$

Case (C2). We separately consider proper pairs (X, Y) in which $X \cap \{v_1, y_1\} = \emptyset$ and the remaining ones. If $X \cap \{v_1 y_1\} = \emptyset$, then any vertex from $\{v_1, y_1\}$ may or may not be included in Y (this gives us 4 possibilities) and we continue with the graph $G - \{v_1, y_1\}$. If $X \cap \{v_1, y_1\} \neq \emptyset$, then none of the vertices from $\{u_1, w, x\}$ can be in X. However, each of them can be in Y. Moreover, every non-empty subset of $\{v_1, y_1\}$ can be in X and any every vertex which is not in X, may or may not be included into Y. This gives us $2^3 \cdot (3^2 - 2^2) = 40$ possibilities and we can continue with the graph $G - \{v_1, y_1, u_1, w, x\}$.

$$\begin{aligned} pp_{cf}(n) =& pp(G) \leq 4pp(G - \{v_1, y_1\}) + 40pp(G - \{v_1, y_1, u_1, w, x\}) \\ \leq& 4pp_{cf}(n-2) + 40pp_{cf}(n-5). \end{aligned} \tag{3.9}$$

One can verify that inequalities (3.8) and (3.9) imply that $pp_{cf}(n) \leq 2 \cdot \tau^n$, where $\tau \approx 2.59431$ is the positive root of the equation $x^3 = 2x^2 + 4$. As the path P_n has exactly $\Theta(\tau^n)$ proper pairs, the obtained bound is tight (up to a polynomial factor). □

Graphs with a small dominating set

Let G be an r-dominated graph with n vertices. Using the formula (3.4) and the bound for $u_k(G)$ from Proposition 3.17, we obtain the following corollary.

Corollary 3.22 ([53])**.** *There are at most $2^{n-r}\left(2+\frac{n}{r}\right)^r$ proper pairs in any r-dominated graph on n vertices. If a dominating set of size at most r is given, then we can enumerate them in time $\mathcal{O}^*(2^{n-r}\left(2+\frac{n}{r}\right)^r)$, using polynomial space.*

A simple calculation shows that for $\frac{n}{r} > 3.7729$ the bound from Corollary 3.22 is better than the bound from Theorem 3.19. Observe that the power base in the bound from Corollary 3.22 converges to 2 as $\frac{n}{r}$ grows (more formally, we have $\lim_{n/r\to\infty} \sqrt[n]{2^{n-r}\left(2+\frac{n}{r}\right)^r} = 2$).

3.4 Red-black graphs

In this section we consider graphs with two kinds of edges. A triple $G = (V, R, B)$ is called a *red-black graph* if V is a finite set of vertices and R, B are disjoint families of 2-element subsets of V fulfilling the condition: if $vw \in B$ and $vu \in B$, then $uw \in R \cup B$ for any pairwise distinct vertices $v, u, w \in V$. For a red-black graph $G = (V, R, B)$ we refer to V, R, B, $R \cup B$ by $V(G)$, $R(G)$, $B(G)$, $E(G)$, respectively. We call $R(G)$ the set of red edges and $B(G)$ the set of black edges, while $E(G)$ is the set of edges.

If not stated otherwise, we define all basic graph-theoretic terms, such as connectivity, independent set, induced subgraphs etc. for a red-black graph (V, R, B) to be equivalent to the corresponding terms for the graph $(V, R \cup B)$.

An *R-closure* of a graph H is a red-black graph G, such that $V(G) = V(H)$, $B(G) = E(H)$ and $R(G) = E(H^2) \setminus E(H) = \{uw \in V(H) \colon \exists v \in V(H)\ uv \in E(H) \text{ and } vw \in \binom{V(H)}{2} \text{ and } uw \notin E(H)\}$.

A *red neighborhood* (*black neighborhood*, respectively) of a vertex v, denoted by $N_R(v)$ ($N_B(v)$, respectively), is the set of vertices w such that $vw \in R(G)$ ($vw \in B(G)$, respectively). The red neighborhood and the black neighborhood of a set Y of vertices in G are denoted by $N_R(Y) = \bigcup_{v \in Y} N_R(v)$ and $N_B(Y) = \bigcup_{v \in Y} N_B(v)$, respectively.

Observation 3.23. *If G is the R-closure of some graph H, then independent sets in G are exactly 2-packings in H.*

Now let us extend the definition of proper pairs (see Section 3.3) to red-black graphs. For a red-black graph G, a pair (X, Y) of disjoint subsets of $V(G)$ is *proper* if X is independent.

An independent set X is *R-maximal* if every vertex v such that $N_R(v) = N(v)$, is either in X or has a neighbor in X. We say that a proper pair (X, Y) is *R-maximal* if X is R-maximal.

For a red-black graph G let G_B denote the red-black graph induced by the set of vertices belonging to black edges, i.e. $G[\bigcup_{e \in B(G)} e]$ (note that it may have some red edges). By G_R we denote the red-black subgraph induced by the set of vertices $V(G) \setminus V(G_B)$. We say that a red-black graph G is *black* (*red*, respectively) if $G = G_B$ ($G = G_R$, respectively).

3.4.1 The number of R-maximal independent sets

Let $r_k(G)$ denote the maximum number of k-element R-maximal independent sets in a red-black graph G and let $r_k(n)$ be a maximum of $r_k(G)$ over all red-black graphs G on n vertices (note that we do not require connectivity).

We shall use the bound for the number of k-element 2-packings by Havet *et al.* [45] we already mentioned in Section 3.1 in Theorem 3.4. For our purpose it is more convenient to restate it in the language of R-closures, according to Observation 3.23.

Corollary 3.24 (Havet *et al.* [45]). *The maximum number $u'_k(n)$ of k-element independent sets in a graph with no isolated vertices, which is an R-closure of some graph, is at most $\binom{n/2}{k} \cdot 2^k$, where n denotes the number of vertices in a graph.*

We shall also use the following bound for the maximum number of k-element maximal independent sets.

Theorem 3.25 (Eppstein [24]). *The maximum number $mis_k(n)$ of k-element maximal independent sets in a graph on n vertices is at most $3^{4k-n}4^{n-3k} = (81/64)^k(4/3)^n$.*

Now we can bound the value of $r_k(n)$.

Lemma 3.26 ([54]). *Let G be a red-black graph on n vertices with $r_k(n)$ R-maximal independent sets and let $n' = |V(G_B)|$. Then*

$$r_k(n) \leq (4/3)^{n-n'} \left(\frac{81}{64}\right)^k \sum_{k'=0}^{k} \binom{n'/2}{k'} \left(\frac{128}{81}\right)^{k'}.$$

Moreover, all R-maximal independent sets in G can be enumerated in time $n^{\mathcal{O}(1)} \cdot (4/3)^{n-n'} \left(\frac{81}{64}\right)^k \sum_{k'=0}^{k} \binom{n'/2}{k'} \left(\frac{128}{81}\right)^{k'}$, using polynomial space.

Proof. We shall construct all R-maximal independent sets X in two steps:

1. From G_B select an independent set X_B.
2. From G_R select a maximal independent set X_R such that $X_R \cap N(X_B) = \emptyset$.
3. Return $X = X_B \cup X_R$.

To see that this algorithm is correct, consider sets X_B and X_R selected in steps 1 and 2. Since each of the sets X_B and X_R is independent and $X_R \cap N(X_B) = \emptyset$, we obtain that $X = X_B \cup X_R$ is independent in G. Since G_B is induced by the set of black edges (and therefore has no vertices v with $N(v) = N_R(v)$) and X_R is maximal in G_R, we can observe that X is an R-maximal independent set in G.

On the other hand consider an R-maximal independent set X and let $X_B = X \cap V(G_B)$ and $X_R = X \cap V(G_R)$. Clearly each of the subsets X_R and X_B is independent and there are no edges between X_B and X_R. Moreover, since X is R-maximal, also X_R has to be R-maximal.

Notice that the R-closure of the graph induced by the black edges of G_B is a spanning subgraph of G_B with no isolated vertices. Since every independent set in a graph is independent in its spanning subraph, by Corollay 3.24 we get the following bound:

$$r_k(n) \le \sum_{k'=0}^{k} u'_{k'}(n') \cdot mis_{k-k'}(n-n') \le \sum_{k'=0}^{k} \binom{n'/2}{k'} 2^{k'} \left(\frac{81}{64}\right)^{k-k'} \left(\frac{4}{3}\right)^{n-n'} =$$

$$= \left(\frac{4}{3}\right)^{n-n'} \left(\frac{81}{64}\right)^{k} \sum_{k'=0}^{k} \binom{n'/2}{k'} \left(\frac{128}{81}\right)^{k'}.$$

As all the local operations are polynomial, the time complexity is determined by the upper bound for the number of generated sets (up to a polynomial factor). $\square$

3.4.2 The number of R-maximal proper pairs

Let $rpp(G)$ denote the number of R-maximal proper pairs in a red-black graph G. By $rpp(n)$ we denote the maximum value of $rpp(G)$ over all connected red-black graphs on n vertices.

In this subsection we prove the following Theorem.

Theorem 3.27 ([54]). *The maximum number of R-maximal proper pairs in a connected red-black graph on n vertices is $\Theta(\sqrt{8}^n) = \mathcal{O}(2.8285^n)$. They can be enumerated in time $\mathcal{O}^*(\sqrt{8}^n)$, using polynomial space.*

The proof requires a few steps. First let us prove an appropriate bound for red graphs. Let $rpp_R(n)$ denote the maximum value of $rpp(G)$ over all red graphs on n vertices. Note that R-maximal independent sets in a red graph $(V, R, \emptyset)$ are just maximal independent sets in (V, R). Therefore R-maximal proper pairs in a red graph are the pairs (X, Y) of disjoint sets, where X is a maximal independent set. The proof of Lemma 3.28 is inspired by an elegant proof by Wood [87].

Lemma 3.28. *We have $rpp_R(n) = \mathcal{O}\left(\sqrt[5]{80}^n\right) = \mathcal{O}(2.4023^n)$.*

Proof. We shall prove the statement by induction on the number of vertices n. If $n \leq 2$, the statement is obviously true. Assume that $n \geq 3$ and the statement is true for all red graphs with less than n vertices.

Let G be a red graph on n vertices, such that $rpp(G) = rpp_R(n)$. Let v be a vertex of G of minimum degree δ. Notice that for every R-maximal proper pair (X, Y), at least one of the vertices in $N[v]$ must be in X (since X is a maximal independent set in G). Let $w \in N[v] \cap X$. Since the set X is independent, none of the vertices from $N(w)$ belongs to it. However, each of them may or may not be in Y. Hence we obtain the following recursion: $rpp_R(n) \leq \sum_{w \in N[v]} 2^{\deg w} rpp_R(n - \deg w - 1)$. Let d be the element of $\{\delta, \ldots, n-1\}$ maximizing the expression $2^d rpp_R(n-d-1)$. Then $rpp_R(n) \leq \sum_{w \in N[v]} 2^d rpp_R(n-d-1) = (\delta+1)2^d rpp_R(n-d-1) \leq (d+1)2^d rpp_R(n-d-1)$.

Using the induction hypothesis, we get the bound (c is a constant):

$$rpp_R(n) \leq c \cdot (d+1)2^d \sqrt[5]{80}^{n-d-1} = c \cdot \sqrt[5]{80}^n \cdot \left(\frac{(d+1) \cdot 2^d}{\sqrt[5]{80}^{d+1}} \right).$$

One can easily verify that the value of $\frac{(d+1)\cdot 2^d}{\sqrt[5]{80}^{d+1}}$ does not exceed 1 (the maximum value is 1, achieved for $d = 4$). Hence $rpp_R(n) \leq c\left(\sqrt[5]{80}^n\right) = \mathcal{O}(\sqrt[5]{80}^n) = \mathcal{O}(2.4023^n)$. □

The proof of the lower bound for $rpp_R(n)$ is also analogical to the case of maximal independent sets (see Miller and Muller [75] and Moon an Moser [76]). Consider the red graph H_k consisting of k disjoint copies of the complete graph K_5. A direct computation shows that $rpp(H_k) = \Theta\left((5 \cdot 2^4)^k\right) = \Theta\left(\sqrt[5]{80}^n\right)$, which proves that $rpp_R(n) = \Theta\left(\sqrt[5]{80}^n\right)$.

It is interesting to mention that the same asymptotical bound applies if we restrict ourselves to connected red graphs. Let H'_m be the graph obtained from H_m by adding one new vertex adjacent to exactly one vertex from each copy of K_5. It is easy to check that $rpp(H'_m) = \Theta\left(\sqrt[5]{80}^n\right)$.

Let us proceed to bounding the number of R-maximal proper pairs in black graphs. First, notice that in a black graph, R-maximal proper pairs are just proper pairs. Therefore we can use the bounds from Section 3.3. By Theorem 3.19 we get the following.

Observation 3.29. *If G is a connected black graph with n vertices, then $rpp(G) = \mathcal{O}(2.6488^n)$. Moreover, all R-maximal proper pairs in G can be enumerated in time $\mathcal{O}(2.6488^n)$, using polynomial space only.*

If G is not connected, but has no isolated vertices, then we obtain the following bound.

Lemma 3.30. *The maximum number of R-maximal proper pairs in a black graph G without isolated vertices is upper-bounded by $\mathcal{O}(\sqrt{8}^n) = \mathcal{O}(2.8285^n)$.*

Proof. Let us consider proper pairs (X, Y) and let $k = |X|$. Since the graph G has a spanning subgraph, which is an R-closure of some graph and has no isolated vertices, we can use Corollary 3.24. The set X can be chosen in at most $u'_k(n)$ ways. Notice that every vertex that is not in X may or may not be included in Y. Finally, by Corollary 3.24, we obtain the following formula: $rpp(G) = \sum_{k=0}^{n} u'_k(n) \cdot 2^{n-k} = \sum_{k=0}^{n} \binom{n/2}{k} \cdot 2^k \cdot 2^{n-k} = \mathcal{O}(\sqrt{8}^n) = \mathcal{O}(2.8285^n)$. □

Now we are ready to bound the number of R-maximal proper pairs in all connected red-black graphs.

Proof of Theorem 3.27. Let G be a connected red-black graph on n vertices with the largest number of R-maximal proper pairs. Again, we shall construct pairs of sets (X, Y) in two steps:

1. From G_B select an independent set X_B and a set Y_B disjoint with X_B.
2. From G_R select a maximal independent set X_R such that $X_R \cap N(X_B) = \emptyset$, and a set Y_R disjoint with X_R.
3. Return $X = X_B \cup X_R$ and $Y = Y_B \cup Y_R$.

Notice that such constructed pairs (X, Y) are exactly R-maximal proper pairs in G (the correctness of this procedure can be proven similarly as in the proof of Lemma 3.26). Since the graph G_B has no isolated vertices, for $|V(G_B)| = n'$ we obtain the following formula by Lemmas 3.28 and 3.30:

$$\begin{aligned} rpp(n) = & rpp(G) = \mathcal{O}(\sqrt{8}^{n'} \cdot rpp_R(n - n')) = \\ & \mathcal{O}(\sqrt{8}^{n'} \cdot \sqrt[5]{80}^{n-n'}) = \mathcal{O}(\sqrt{8}^{n}) = \mathcal{O}(2.8285^n). \end{aligned}$$

To show that this bound is best possible, let us consider the graph M_k consisting of k disjoint black edges and a vertex v. We join with red edges one of the vertices of each black edge with v (see Figure 3.15).

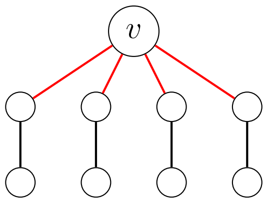


Figure 3.15: Graph M_4.

A direct calculation shows that $rpp(M_k) = \Theta(8^k) = \Theta(8^{n/2})$. Thus $rpp(n) = \Theta(\sqrt{8}^n)$. The proof is constructive and can be easily transformed to an algorithm enumerating all R-maximal proper pairs in a connected red-black graph with n vertices in time $\mathcal{O}^*(\sqrt{8}^n)$, using polynomial space. This completes the proof of Theorem 3.27. □

3.5 Partitions of graphs to subgraphs of bounded order

In this section we consider a problem closely related to the *factorization* of a graph (see the book by Akiyama and Kano [2] for more details). For a graph class Γ, a Γ-*factor* in a graph is its spanning subgraph, whose every connected component belongs to Γ.

We shall be mostly interested in classes Γ in which all graphs are connected and have at least ℓ and at most ℓ' vertices (for some constants $1 < \ell \leq \ell'$). It is easy to note that not every graph G admits such a Γ-factor (a large star is a counterexample). We deal with this problem in two ways: by relaxing our constraints (in Section 3.5.1) or by imposing some additional constraints on the graph G (Section 3.5.2).

3.5.1 Covering graphs with connected subgraphs

If we relax the constraints for a factorization and allow a small overlap between subgraphs, we can always find a *covering* of any graph G with connected subgraphs with at least ℓ and at most 2ℓ vertices each.

Theorem 3.31 ([53]). *Let G be a connected graph of order n and let $\ell < n$ be a positive integer. Then there exist connected subgraphs $G_1, G_2, \ldots, G_q$ of G such that*

1. *every vertex of G belongs to the vertex set of at least one of them,*
2. *the order of each of the graphs $G_1, G_2, \ldots, G_{q-1}$ is at least ℓ and at most 2ℓ and $|V(G_q)| \leq 2\ell$,*
3. $\sum_{i=1}^{q} |G_i| \leq n\left(1 + \frac{1}{\ell}\right)$.

Proof. Choose any vertex $r \in V(G)$ and consider a DFS-tree T of G rooted in r. For every vertex v let $T(v)$ be the subtree of T rooted in v. If $|T(r)| \leq 2\ell$ then add G to the set of desired subgraphs and stop. If there is a vertex v such that $\ell \leq |T(v)| \leq 2\ell$ then add $G[V(T(v))]$ to the set of desired subgraphs and proceed recursively with $G \setminus V(T(v))$. Otherwise there must be a vertex v such that $|T(v)| > 2\ell$ and for its every child u, $|T(u)| < \ell$. In such a case find a subset $\{u_1, \ldots, u_j\}$ of children of v such that $\ell \leq |T(u_1)| + \cdots + |T(u_j)| \leq 2\ell - 1$. Add $G[\{v\} \cup V(T(u_1)) \cup \cdots \cup V(T(u_j))]$ to the set of desired subgraphs and proceed with the graph $G\setminus(V(T(u_1))\cup\cdots\cup V(T(u_j)))$. This procedure terminates after at most $\frac{n}{\ell}$ steps and in each of them we have left at most one vertex of the identified connected subgraph in the further processed graph. Notice that from this construction it follows that for every $i \in \{1,\dots,q\}$ the graph $G-(V(G_1)\cup V(G_2)\cup\cdots\cup V(G_{q-1}))$ is connected. $\square$

3.5.2 Partitioning graphs to stars

Let $\mathcal{S}=\{S_1,S_2,..,S_r\}$ be a partition of the vertex set of G, such that for any $i=1,2,..,r$ we have $|S_i|\geq 2$ and $G[S_i]$ has a spanning subgraph, which is a star (sets S_i are vertex sets of connected components of a star factor of G). Such a partition $\mathcal{S}$ will be called a *star partition* of G.

Notice that a star partition can be constructed from a spanning tree of a connected graph. Let us consider T being a spanning tree of G. Let v and u be, respectively, the end-vertex and its neighbor on a longest path in T. If T is not a star, then all neighbors of u in T except exactly one are leaves in T. We include the set S_i consisting of u and all its neighbors which are leaves in T to our partition $\mathcal{S}$. Then we proceed recursively with the tree $T\setminus S_i$. If T is a star, we set $S_r=V(T)$ and stop.

Moreover, notice that if we construct our star partition $\mathcal{S}=\{S_1,S_2,..,S_r\}$ using a spanning tree T, in the way described above, each set S_i for $i=1,2,..,r-1$ has at most $\Delta(T)$ elements, while S_r has at most $\Delta(T)+1$ elements. Since $\Delta(T)\leq\Delta(G)$, we obtain the following lemma.

Lemma 3.32 ([59]). *Every connected graph on n vertices and maximum degree Δ has a star partition $\mathcal{S}=\{S_1,S_2,..,S_r\}$ with $2\leq|S_i|\leq\Delta$ for $i=1,2,..,r-1$ and $2\leq|S_r|\leq\Delta+1$.*

Notice that if we want all sets in a star partition to be small, we should start with a spanning tree whose maximum degree is lowest possible. However, deciding if the input graph has a spanning tree with maximum degree at most k is NP-complete for every $k\geq 2$ (see Garey and Johnson [39]).

The following theorem shows that if the minimum degree of G is big, we may obtain a star partition consisting of smaller sets.

Theorem 3.33 (Amashi, Kano [6], Payan [2]). *Let $D\geq 2$ be an integer such that $\Delta(G)/\delta(G)\leq D$. Then G has a star partition $\mathcal{S}=\{S_1,S_2,..,S_r\}$ such that $2\leq|S_i|\leq D+1$ for any $i=1,2,..,r$.*

We can improve those bounds for graphs with no big induced stars. The simplest and probably best-studied class with such a property are already mentioned claw-free graphs. Sumner [83] showed that every claw-free graph with even number of vertices has a perfect matching. This implies the existence of a star factor with almost all sets of cardinality 2.

Corollary 3.34. *Every connected claw-free graphs has a star partition $\mathcal{S} = \{S_1, S_2, .., S_r\}$ in which $|S_i| = 2$ for $i = 1, 2, .., r-1$ and $|S_r| \leq 3$.*

A similar result can be obtained for $K_{1,d}$-free graph for any $d \geq 3$.

Lemma 3.35 ([59]). *Every connected $K_{1,d}$-free graph G has a star partition $\mathcal{S} = \{S_1, S_2, .., S_r\}$ such that $|S_i| \leq d-1$ for any $i = 1, 2, .., r-1$ and $|S_r| \leq d$.*

Proof. It is easy to verify that the theorem is true for $d = 2$. Assume that $d \geq 3$. Again we shall construct $\mathcal{S}$ using a spanning tree. If G has at most d vertices, we partition G into stars in any way and finish. Otherwise, let T be a spanning tree of G with the largest diameter and let v_1, u and x be, respectively, the first, the second and the third vertex of a longest path in T. Notice that v_1 is a leaf in T and every neighbor of u in T but at most one vertex (i.e. x) is a leaf in T as well. Let $\{v_1, v_2, \ldots, v_k\}$ be the set of neighbors of u, which are leaves in T. We shall consider two cases.

Case 1. If $k \leq d-2$, then we include the set $\{u, v_1, v_2, \ldots, v_k\}$ to our partition and proceed with the graph $G \setminus \{u, v_1, v_2, \ldots, v_k\}$. The included set has at most $d-1$ vertices.

Case 2. Suppose now that $k \geq d-1$. Observe that if $k = d-1$, then x is not a leaf, since we assumed that G has at least $d+1$ vertices. Therefore the set $\{u\} \cup \{x, v_1, v_2, \ldots, v_k\}$ does not induce a star in G (as G is $K_{1,d}$-free). Thus $v_iv_j \in E(G)$ for some $1 \leq i < j \leq k$ or $v_ix \in E(G)$ for some $1 \leq i \leq k$.

Subcase 2.1 If $v_iv_j \in E(G)$ for some $1 \leq i < j \leq k$ (without loss of generality let $i = 1$ and $j = 2$), consider the spanning tree tree $T' := (V(G), E(T) \cup \{v_1v_2\} \setminus \{uv_1\})$ (see Figure 3.16). Its diameter is strictly larger than the diameter of T, which contradicts with the choice of T.

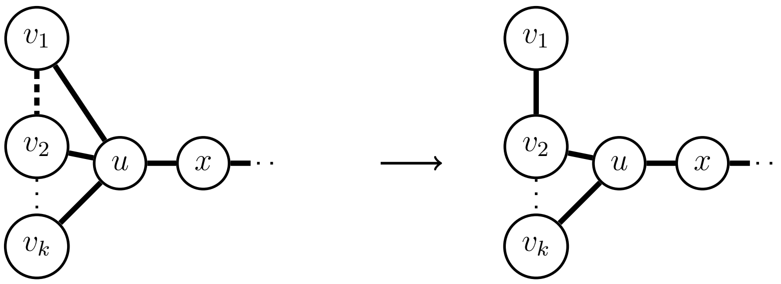


Figure 3.16: Subcase 2.1)

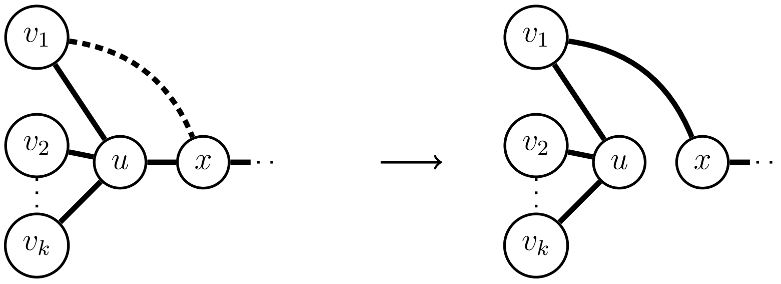


Figure 3.17: Subcase 2.2

Subcase 2.2 If $v_ix \in E(G)$ for some $1 \leq k$ (without loss of generality let $i = 1$), consider the tree $T' := (V(G), E(T) \cup \{v_1x\} \setminus \{ux\})$ (see Figure 3.17). Again, $\text{diam}(T') > \text{diam}(T)$, which is a contradiction.

This completes the proof. □

Chapter 4

Algorithms using exponential space

In this chapter we present exact algorithms for some graph labeling problems. Key ingredients in our approach are algebraic manipulations loosely inspired by the fast matrix multiplication: if we have $2^k \times 2^k$-matrices A and B, we can divide each of them into four $2^{k-1} \times 2^{k-1}$ block matrices. We can then compute $A \cdot B$ very easily by eight matrix multiplications of $2^{k-1} \times 2^{k-1}$-matrices. Doing so recursively leads again to a running time of $\mathcal{O}(n^3)$, just as the in naive algorithm itself. It is, however, possible to improve the running time by using only *seven* matrix multiplications to achieve the same result (see the classical result of Strassen [82]). We also use one other trick: we jump between two representations of partial labelings in the course of our dynamic programming algorithm. The idea to use different representations of the same data in dynamic programming is not new and was used in a similar way before (see for example van Rooij *et al.* [84]).

We start this chapter with definitions of the problem we are going to consider. The remaining sections contain mostly joint results from the following papers:

Section 4.2:	a paper co-authored with K. Junosza-Szaniawski [59],
Section 4.3:	a paper co-authored with K. Junosza-Szaniawski, J. Kratochvíl, M. Liedloff and P. Rossmanith [53],
Section 4.4:	a recent paper by the author [78].

4.1 List of problems

In this section we formally define the problems we are going to discuss later. Most of them have already been mentioned in the introduction. Observe that some of the problems are *decision problems* (i.e. YES/NO questions) and some are *optimization problems* or, to be more specific, *minimization problems* (i.e. we are interested in finding a minimum value of some parameter).

Graph coloring. An instance of the *graph coloring* problem is a graph $G = (V, E)$. A *proper coloring* of G is a mapping $\varphi\colon V \to \mathbb{N}$, such that $\varphi(u) \neq \varphi(v)$ for every edge $uv \in E$. In the graph coloring problem we want to compute the *chromatic number* of G (denoted by $\chi(G)$), being the minimum possible number of colors used by a proper coloring of G. In the decision version of this problem, called the *k-coloring* problem, we ask if the chromatic number of the input graph is at most k.

The *list graph coloring* problem is the generalization of the graph coloring problem. The instance of the generalized problem is a graph G and a function $\Lambda\colon V \to 2^{\mathbb{N}}$ (for a vertex $v \in V$, the set $\Lambda(v)$ is called the *list* for v). We are interested in the existence of a proper coloring φ of G, in which $\varphi(v) \in \Lambda(v)$ for every $v \in V$. The list versions of the problems described below will be defined in an analogous way.

$L(p, q)$**-labeling.** For a graph $G = (V, E)$ and integers $p \geq q \geq 1$, an *$L(p, q)$-labeling* is a function $\varphi\colon V \to \mathbb{N}_0$, such that:

1. $|\varphi(v) - \varphi(u)| \geq p$ if $\text{dist}(u, v) = 1$,

2. $|\varphi(v) - \varphi(u)| \geq q$ if $\text{dist}(u, v) = 2$.

The *span* of an $L(p, q)$-labeling is the difference between the maximum and the minimum label used. The *$L(p, q)$-labeling problem* asks to find the minimum possible span of an $L(p, q)$-labeling of G. The *k-$L(p, q)$-labeling problem* asks for existence of a *k-$L(p, q)$-labeling* of G, i.e. an $L(p, q)$-labeling of G using only labels $\{0, 1, .., k\}$ (note that we have $k + 1$ available labels).

We shall focus mostly on the case of $p = 2$ and $q = 1$. The minimum possible span of an $L(2, 1)$-labeling of G is denoted by $\lambda(G)$.

Channel assignment. Let $G = (V, E)$ be a graph and let $\omega\colon E \to \mathbb{N}$ be a weight function. A *channel assignment* of (G, ω) is a mapping $\varphi\colon V \to \mathbb{N}$, such that $|\varphi(u) - \varphi(v)| \geq \omega(uv)$ for any $uv \in E$. The *channel assignment problem* asks for the minimum span (i.e. the difference between the smallest and the largest label used) of a channel assignment of (G, ω). We say that the instance (G, ω) of the channel assignment problem is *ℓ-bounded* for some integer ℓ, if $\omega(e) \leq \ell$ for every $e \in E$.

We can consider the channel assignment problem as a generalization of the coloring and the $L(p, q)$-labeling problems. Indeed, the channel assignment problem for (G, ω), where $\omega(e) = 1$ for all $e \in E$ is equivalent to the graph coloring problem of G.

Now for a graph $G = (V, E)$ let $G' = (V, E')$ be its square graph. Define $\omega\colon E' \to \mathbb{N}$ as follows:

$$\omega(e) = \begin{cases} p & \text{if } e \in E \\ q & \text{if } e \in E' \setminus E. \end{cases}$$

It is easy to see that the channel assignment problem for (G', ω) is equivalent to the $L(p, q)$-labeling problem for G.

T-coloring. Let $G = (V, E)$ be a graph and let $T \subseteq \mathbb{N}_0$ be a set of integers. A *T-coloring* of G is a mapping $\varphi\colon V \to \mathbb{N}$, such that $|\varphi(u) - \varphi(v)| \notin T$ for every $uv \in E$. The *T-coloring problem* asks for the minimum span (i.e. the difference between the smallest and the largest label used) of a T-coloring of G. Observe that if $T = \{0\}$, then we obtain the graph coloring problem.

Generalized list T-coloring. An instance of a the *generalized list T-coloring problem* is a triple (G, Λ, t), where $G = (V, E)$ is a graph, $\Lambda\colon V \to 2^{\mathbb{N}}$ is a function that assigns to each vertex a set (list) of permitted labels and $t\colon E \to 2^{\mathbb{N}_0}$ is a function that assigns to each edge a set of forbidden differences over that edge. We assume that $0 \in t(e)$ for any $e \in E$. We are interested in determining the existence of a mapping $\varphi\colon V \to \mathbb{N}$, satisfying the following conditions:

1. $\varphi(v) \in \Lambda(v)$ for every $v \in V$,

2. $|\varphi(v) - \varphi(w)| \notin t(vw)$ for every edge $vw \in E$.

Such a function is called a *proper labeling.* The generalized list T-coloring has been introduced as a common generalization of the (list) channel assignment and the (list) T-coloring problems (and thus also the graph coloring and the $L(p, q)$-labeling problems).

For an instance (G, ω) of the list channel assignment problem with lists Λ, we have an equivalent instance (G, Λ, t) of the generalized list T-coloring, where $\Lambda(v)$ is the list of labels available for v (or the set of all labels in a non-list case) and $t(e) = \{0, 1, .., \omega(e) - 1\}$ for $e \in E(G)$.

On the other hand consider a graph G and the set $T \in \mathbb{N}_0$, being an instance of the T-coloring problem. Define $t \colon E \to 2^{\mathbb{N}_0}$, such that $t(e) = T$ for every $e \in E$. The instance (G, Λ, t) of the generalized list T-coloring problem is equivalent to the T-coloring problem for G with lists Λ.

(m, k)-coloring. For a graph $G = (V, E)$ and integers $m \geq 2k$, the *(m, k)-coloring* of G (see Zhu [90]) is an assignment $\varphi \colon V \to \{0, 1, .., m - 1\}$, such that $k \leq |\varphi(v) - \varphi(u)| \leq m - k$ whenever $vu \in E$. The *(m, k)-coloring problem* asks for the existence of such an assignment for G.

Graph homomorphism (or H-coloring). For graphs G and H we say that $\varphi \colon V(G) \to V(H)$ is a *homomorphism* from G to H if $\varphi(v)\varphi(u) \in E(H)$ for any $uv \in V(G)$. The *graph homomorphism problem* for graphs G and H asks if there exists a homomorphism from G to H.

The graph homomorphism problem is a natural generalization of the graph coloring problem (in this case H is a complete graph). Also observe that the (m, k)-coloring problem of G is equivalent to the homomorphism problem from G to $\overline{C}_m^{k-1}$, the complement of the $(k - 1)$'th power of the m-cycle[†].

$H(2, 1)$-labeling. For graphs G and H, the *$H(2, 1)$-labeling problem* asks for the existence of the *$H(2, 1)$-labeling* of G, i.e. a mapping $\varphi \colon V(G) \to V(H)$, such that:

1. $\text{dist}_H(\psi(v), \psi(w)) \geq 2$ if $\text{dist}_G(v, w) = 1$,

2. $\text{dist}_H(\psi(v), \psi(w)) \geq 1$ if $\text{dist}_G(v, w) = 2$.

[†]Here we consider C_m^0 to be a graph with m vertices and no edges.

It is easy to see that the $L(2,1)$-labeling of G is an $H(2,1)$-labeling of G for H being a path. Another interesting case is the $L_c(2,1)$-labeling (see Liu, Zhu [69]). It is equivalent to finding the minimum k, for which the input graph G has an $H(2,1)$-labeling for H being a cycle with $k+1$ vertices.

Locally injective graph homomorphism. We say that a homomorhism φ from G to H is *locally injective*, if the neighborhood of $v \in V(G)$ is mapped injectively to the neighborhood of $\varphi(v)$. Formally, for any two vertices $u, v \in V(G)$ with $\operatorname{dist}(u,v) \leq 2$ we have $\varphi(u) \neq \varphi(v)$. The *locally injective homomorphism problem* for graphs G and H is to determine the existence of a locally injective homomorphism from G to H.

Fiala and Kratochvíl [29] showed a close relationship between locally injective homomorphisms and $H(2,1)$-labelings.

Theorem 4.1 (Fiala, Kratochvíl [29]). *For graphs G and H, an $H(2,1)$-labeling of G is exactly a locally injective homomorphism from G to $\overline{H}$.*

Remark 4.2. It is clear that if we have an algorithm for the minimization version of some problem, then we can easily solve the corresponding decision problem as well. For example, having computed $\chi(G)$ for a graph G (and thus solving the coloring problem), it suffices to check if $\chi(G) \leq k$ to solve the k-coloring problem for G. On the other hand, if we have an algorithm for the decision problem (e.g. the k-coloring problem) working in time $F(n)$ (where n is the size of the instance), then we can solve the corresponding minimization problem by repeatedly running the algorithm for the k-coloring problem for $k = 1, 2, 3, ...$ and returning the smallest k for which we obtain Yes as the answer. If the maximum value of the parameter we want to minimize is bounded by a polynomial function of n, the time complexity of the whole procedure is bounded by $n^{\mathcal{O}(1)} \cdot F(n)$.

4.2 Exact algorithm for the generalized list T-coloring problem

Consider (G, Λ, t) being an instance of the generalized list T-coloring problem. We say that an instance of the generalized list T-coloring problem is τ-bounded if the largest value in the set $\bigcup_{e \in E} t(e)$ does not exceed τ.

Let $[\tau + 1]$ denote the set $\{0, 1, .., \tau + 1\}$ and $[\![\tau + 1]\!]$ denote the set $[\tau + 1] \cup \{\bar{0}\}$, where $\bar{0}$ is a special symbol, whose meaning will be made clear later. Note that $|[\tau + 1]| = \tau + 2$.

Definition 4.3. For a vector $\mathbf{w} \in \Sigma^k$ (for some alphabet Σ) and a set of vectors $A \subseteq \Sigma^n$ let $A_{\mathbf{w}}$ denote the set $\{\mathbf{v} \in \Sigma^{n-k} \colon \mathbf{wv} \in A\}$ (by $\mathbf{wv}$ we denote the vector from Σ^n obtained from $\mathbf{w}$ by appending $\mathbf{v}$ to its end). The vector $\mathbf{w}$ is also called a *prefix* of the vector $\mathbf{wv}$.

Moreover, we say that $\mathbf{a}' \in \Sigma^{k'}$ is a *segment* of $\mathbf{a} = \mathbf{a}_1\mathbf{a}_2 \dots \mathbf{a}_k \in \Sigma^k$ (for any alphabet Σ and any $k' \leq k$) if $\mathbf{a}' = \mathbf{a}_i\mathbf{a}_{i+1} \dots \mathbf{a}_{i+k'-1}$ for some $i \leq k - k' + 1$.

4.2.1 The algorithm

Let (G, Λ, t) be a τ-bounded instance of the generalized list T-coloring problem. If $\tau = 0$, then we obtain a well-studied list coloring problem, which can be solved in time $\mathcal{O}^*(2^n)$ by adapting the algorithm by Björklund *et al.* [9]. Thus we focus on the case when $\tau \geq 1$. Moreover, we assume that the graph $G = (V, E)$ is connected – in the other case we may label each of its connected components separately.

By a *partial labeling* (*partial k-labeling*) of G we mean a mapping $\varphi \colon V' \to \mathbb{N}$ (resp. $\varphi \colon V' \to \{1, 2, .., k\}$), such that $V' \subseteq V$ and $\varphi(v) \in \Lambda(v)$ for every $v \in V'$ and $|\varphi(u) - \varphi(v)| \notin t(uv)$ for any $uv \in E \cap \binom{V'}{2}$.

Let Λ_{max} denote the maximum value in the set $\bigcup_{v \in V} \Lambda(v)$. Observe that in any partial labeling of G, a vertex $v \in V$ may have at most $(2\tau+1) \cdot \deg v \leq (2\tau + 1)n$ forbidden labels. Thus, if $|\Lambda(v)| > (2\tau + 1)n$, then we may label the graph $G - v$ and then restore v and give it some non-forbidden label. So we may assume that $|\Lambda(v)| \leq (2\tau + 1)n$ for any vertex v and thus:

$$|\bigcup_{v \in V} \Lambda(v)| \leq \sum_{v \in V} |\Lambda(v)| \leq (2\tau + 1)n^2. \tag{4.1}$$

Assume that there exist two labels $x, y \in \bigcup_{v \in V} \Lambda(v)$ such that:

1. $d := y - x - \tau - 1 > 0$,

2. there is no $z \in \bigcup_{v \in V} \Lambda(v)$ such that $x < z < y$.

It is easy to verify that in this case the instance (G, Λ, t) of the generalized list T-coloring is equivalent to the instance (G, Λ', t), where $\Lambda'(v) = \{c \in$

$\Lambda(v)\colon c \leq x\} \cup \{c - d\colon c \in \Lambda(v) \text{ and } c > x\}$ for every $v \in V$. Informally speaking, every label greater than x is just shifted down by d. Since they are still greater than $x + \tau$, no new conflict between labels appears (and clearly no conflicts disappear). We can repeat this transformation until the difference of every pair of subsequent labels in $\bigcup_{v \in V} \Lambda(v)$ is at most $\tau + 1$. For a similar reason we may assume that $1 \in \bigcup_{v \in V} \Lambda(v)$, because otherwise we can shift all the labels down (recall that $0 \notin \Lambda(v)$ for all v). Combining this with the formula (4.1), we may assume that:

$$\Lambda_{max} \leq |\bigcup_{v \in V} \Lambda(v)| \cdot (\tau + 2) \leq (2\tau + 1)(\tau + 2)n^2. \tag{4.2}$$

For a graph G we consider some ordering $v_1, v_2, .., v_n$ of the vertices in V (this ordering will be specified later).

Let φ be a partial k-labeling of G. Consider a vertex v, which is unlabeled with φ, and $k + 1 \in \Lambda(v)$. Then we can extend φ by setting $\varphi(v) = k + 1$ if and only if there is no neighbor u of v such that $\varphi(u)$ is defined and $(k + 1) - \varphi(u) \in t(uv)$. Notice that if $\varphi(u) \leq k - \tau$, then $(k + 1) - \varphi(n) \geq \tau + 1 \geq t(uv) + 1$. So we only need to keep track of vertices labeled with labels greater than $k - \tau$. This is the reason why in our encoding of partial k-labelings we do not have to distinguish vertices labeled with labels not exceeding $k - \tau$.

For every $k \in \{1, .., \Lambda_{max}\}$ we introduce a set of vectors $T[k] \subseteq [\tau + 1]^n$, defined as follows.

Definition 4.4. The set $T[k]$ consists of vectors $\mathbf{a} \in [\tau + 1]^n$ such that there exists a partial labeling $\varphi\colon V \to \{1, 2, .., k\}$, satisfying the following condition:

$$\mathbf{a}_i = \begin{cases} 0 & \text{if } v_i \text{ is not labeled by } \varphi, \\ 1 & \text{if } \varphi(v_i) \leq k - \tau, \\ \varphi(v_i) - k + \tau + 1 & \text{if } \varphi(v_i) > k - \tau. \end{cases}$$

Moreover, define $T[0] := \{0^n\}$.

Observe that if $\varphi(v_i) > k - \tau$, then $a_i \in \{2, 3, .., \tau + 1\}$. Note that vectors with no 0's correspond to generalized list T-colorings of G. Hence, once we have computed $T[\Lambda_{max}]$, we can easily verify if there exists a proper labeling of the input graph. Moreover, we can do it in time linear in size of

$T[\Lambda_{max}]$. Formally, the instance of the generalized list T-coloring problem is a YES-instance if and only if $T[\Lambda_{max}] \cap \{1, 2, \ldots, \tau + 1\}^n \neq \emptyset$.

To compute the sets $T[k]$ quickly, we shall define two additional operations. Let $k \in \{1, 2, .., \Lambda_{max} - 1\}$ be fixed and assume we have already computed the set $T[k]$. Consider any $\mathbf{a} \in T[k]$ (for $k < \Lambda_{max}$) and a partial k-labeling φ corresponding to $\mathbf{a}$.

We will obtain a vector $\overline{\mathbf{a}} \in [\![\tau + 1]\!]^n$ from $\mathbf{a}$, by changing some 0s to $\bar{0}$s. Let v_i be a vertex, which is not labeled with φ (and thus $\mathbf{a}_i = 0$). If the labeling φ cannot be extended by labeling v_i with $k+1$, then we set $\overline{\mathbf{a}}_i = \bar{0}$. This takes place in one of the following cases: $k+1 \notin \Lambda(v_i)$ or v_i has a neighbor v_j such that $(k+1) - \varphi(v_j) = \tau - a_j + 2 \in t(v_iv_j)$. Otherwise $\overline{\mathbf{a}}_i$ remains equal to 0. Observe that we do not have to care about vertices v for which $\varphi(v) \leq k - \tau$, because they do not create conflicts with vertices labeled with $k+1$. This is why we do not distinguish vertices labeled with labels up to $k - \tau$ in our representation of partial labelings.

Formally, let $\mathbf{a}$ be a vector in $T[k]$. Define a vector $\overline{\mathbf{a}} \in [\![\tau + 1]\!]^n$ in the following way:

$$\overline{\mathbf{a}}_i = \begin{cases} 0 & \text{if } \mathbf{a}_i = 0 \text{ and } k+1 \in \Lambda(v_i) \text{ and there is no} \\ & \quad v_j \in N(v_i) \text{ such that } \tau - \mathbf{a}_j + 2 \in t(v_iv_j) \\ \bar{0} & \text{if } \mathbf{a}_i = 0 \text{ and one of the following cases occurs:} \\ & \quad \bullet\ k+1 \notin \Lambda(v_i) \text{ or} \\ & \quad \bullet\ \text{there exists } v_j \in N(v_i) \text{ with } \tau - \mathbf{a}_j + 2 \in t(v_iv_j) \\ \mathbf{a}_i & \text{if } \mathbf{a}_i > 0 \end{cases}$$

Let $\overline{T}[k]$ denote the set $\{\overline{\mathbf{a}} \colon \mathbf{a} \in T[k]\}$. Note that computing $\overline{T}[k]$ from $T[k]$ takes time $\mathcal{O}(|T[k]| \cdot n^2)$, since we just have to check if $k+1 \in \Lambda(v_i)$ and inspect all edges v_iv_j for all vertices v_i such that $\mathbf{a}_i = 0$. Moreover, observe that $|T[k]| = |\overline{T}[k]|$ for all k.

Now let us define a partial function: $\oplus \colon [\![\tau + 1]\!] \times \{0, 1\} \to [\tau + 1]$ in the following way:

$$x \oplus y = \begin{cases} 0 & \text{if } x \in \{0, \bar{0}\} \text{ and } y = 0 \\ 1 & \text{if } x \in \{1, 2\} \text{ and } y = 0 \\ x - 1 & \text{if } x \in \{3, 4, .., \tau + 1\} \text{ and } y = 0 \\ \tau + 1 & \text{if } x = 0 \text{ and } y = 1 \\ \text{undefined} & \text{otherwise.} \end{cases}$$

The table below shows the values of $\oplus$ for different inputs (entry „$-$" means that the value is undefined) .

$\oplus$	0	$\bar{0}$	1	2	3	4	$\dots$	τ	$\tau+1$
0	0	0	1	1	2	3	$\dots$	$\tau-1$	τ
1	$\tau+1$	$-$	$-$	$-$	$-$	$-$	$\dots$	$-$	$-$

We generalize $\oplus$ to vectors coordinate-wise, i.e.

$$x_1x_2..x_m \oplus y_1y_2..y_m = \begin{cases} (x_1 \oplus y_1)..(x_m \oplus y_m) & \text{if } x_i \oplus y_i \text{ is defined for} \\ & \text{all } i \in \{1, .., m\}, \\ \text{undefined} & \text{otherwise.} \end{cases}$$

For two sets of vectors $A \subseteq [\![\tau + 1]\!]^m$ and $B \subseteq \{0, 1\}^m$ we define

$$A \oplus B = \{\, \mathbf{a} \oplus \mathbf{b} \colon \mathbf{a} \in A,\ \mathbf{b} \in B,\ \mathbf{a} \oplus \mathbf{b} \text{ is defined} \,\}.$$

Let $P \subseteq \{0, 1\}^n$ be the set of the characteristic vectors of all independent sets of G. Formally, $\mathbf{p} \in P$ if and only if there is an independent set $X \subseteq V$ such that for all $i \in \{1, 2, .., n\}$ it holds $\mathbf{p}_i = 1$ if and only if $v_i \in X$.

In the next lemma we observe that for $\mathbf{a} \in T[k]$ and $\mathbf{p} \in P$, computing $\overline{\mathbf{a}} \oplus \mathbf{p}$ corresponds to extending a partial k-labeling φ corresponding to $\mathbf{a}$ by setting $\varphi(v) = k + 1$ for all v from the independent set encoded by $\mathbf{p}$. This is how we shall use $T[k] \oplus P$ to find $T[k + 1]$.

Lemma 4.5. *For all $k \in \{0, .., \Lambda_{max} - 1\}$ it holds that $T[k + 1] = \overline{T}[k] \oplus P$.*

Proof. First we shall prove that $T[k+1] \subseteq \overline{T}[k] \oplus P$. Let $\mathbf{a} \in T[k+1]$ and φ be a partial $(k + 1)$-labeling corresponding to $\mathbf{a}$. Let $\mathbf{p}$ be the characteristic vector of the set $X := \varphi^{-1}(k+1)$. This set is clearly independent, so $\mathbf{p} \in P$. Let φ' be a partial k-labeling obtained from φ by removing the label $k + 1$ (i.e. the vertices from X are not labeled by φ'), and let $\mathbf{a}'$ be a vector in $T[k]$ corresponding to φ'. Consider a vertex $v_i \in X$ (so $\mathbf{p}_i = 1$). Since φ is a partial $(k+1)$-labeling, we have $k+1 \in \Lambda(v_i)$. Moreover, there is no neighbor v_j of v_i, such that $(k+1) - \varphi'(v_j) \in t(v_i v_j)$. By the definition of $T[k]$, it is equivalent to saying that there is no neighbor v_j of v_i with $\tau - \mathbf{a}_j + 2 \in t(v_i v_j)$. Therefore $\overline{\mathbf{a}'}_i = 0$. It is not hard to verify that $\mathbf{a} = \overline{\mathbf{a}'} \oplus \mathbf{p}$.

Now let us prove that $\overline{T}[k] \oplus P \subseteq T[k+1]$. Let $\mathbf{a}$ be a vector from $\overline{T}[k] \oplus P$. Thus there exist $\mathbf{a}' \in T[k]$ and $\mathbf{p} \in P$ such that $\overline{\mathbf{a}'} \oplus \mathbf{p} = \mathbf{a}$. Let φ' be a partial k-labeling corresponding to $\mathbf{a}'$ and X be the independent set encoded by $\mathbf{p}$.

Note that since $\overline{\mathbf{a}'} \oplus \mathbf{p}$ is defined, then X consists of vertices v_i, which are not labeled with φ' and have $\mathbf{a}_i = 0$. Thus the labeling φ' can be extended by labeling every vertex from X with label $k+1$, obtaining a proper partial $(k+1)$-labeling φ. It is not hard to verify that $\mathbf{a}$ is a vector from $[\tau+1]^n$ corresponding to φ and thus $\mathbf{a} \in T[k+1]$. □

Now we are ready to present the algorithm.

Algorithm 1: Solve-GLTC(G, Λ, t)

```
1 $P \leftarrow$ the set of characteristic vectors of independent sets of $G$
2 $T[0] \leftarrow \{0^n\}$
3 for $k \leftarrow 1$ to $\Lambda_{max}$ do
4     compute $\overline{T}[k-1]$ from $T[k-1]$
5     $T[k] \leftarrow \overline{T}[k-1] \oplus P$
6 if $T[\Lambda_{max}] \cap \{1, 2, \ldots, \tau+1\}^n \neq \emptyset$ then return YES
7 else return NO
```

Now let us discuss how to compute the sets $\overline{T}[k] \oplus P$ efficiently. We will partition the vertex set v of G into subsets of a bounded size (they shall be defined later). Let $\mathcal{S} = (S_1, S_2, .., S_r)$ be an ordered partition of V. Let $s_i := |S_i|$ for all $i \in \{1, .., r\}$ (we require that all s_i's are upper-bounded by some constant D). Moreover, the ordering of sets in $\mathcal{S}$ corresponds to an ordering of vertices of the graph (the vertices within each S_i appear in any order):

$$\underbrace{v_1, v_2, \ldots, v_{s_1}}_{S_1}, \underbrace{v_{s_1+1}, v_{s_1+2}, \ldots, v_{s_1+s_2}}_{S_2}, \ldots, \underbrace{v_{n-s_r+1}, v_{n-s_r+2}, \ldots, v_n}_{S_r}. \qquad (4.3)$$

For a vector $\mathbf{a}' = \mathbf{a}_1 \mathbf{a}_2 \ldots \mathbf{a}_n \in T[k]$, its *i-th segment* (for $i \in \{1, 2, .., r\}$) is the segment $\mathbf{a} = \mathbf{a}_{1+\sum_{j=1}^{i-1} s_j} \ldots \mathbf{a}_{\sum_{j=1}^{i} s_j}$ (so $\mathbf{a}$ corresponds to vertices from

S_i). A vector $\mathbf{a} \in [\tau]^{s_i}$ is an i-th segment of $T[k]$ if it is the i-th segment of some $\mathbf{a}' \in T[k]$.

Moreover, we say that a vector $\mathbf{a} \in [\tau+1]^{s_i}$ is *k-feasible for S_i*, if it encodes some proper partial labeling of $G[S_i]$ from lists $\Lambda|_{S_i}$, using only labels $\{1, 2, .., k\}$ (consult Definition 4.4 to see how vectors encode partial labelings). A vector in $[\tau+1]^{s_i}$ is *feasible for S_i* if it is k-feasible for S_i for at least one k. One can observe that every i'th segment of $T[k]$ is k-feasible for S_i.

We shall process the vertices of the whole subsets S_i at once, in every step considering the first set from $\mathcal{S}$. After that we remove S_1 from $\mathcal{S}$ and reduce the index of every remaining set in $\mathcal{S}$ by one, so that the first one is still called S_1. The formula (4.4) describes a single step of this procedure (recall Definition 4.3).

$$\overline{T}[k] \oplus P = \bigcup_{\substack{\mathbf{b} \in [\![\tau+1]\!]^{s_1} \\ \mathbf{p} \in \{0,1\}^{s_1} \\ \text{s.t. } \mathbf{b} \oplus \mathbf{p} \text{ is defined}}} (\mathbf{b} \oplus \mathbf{p})(\overline{T}[k]_{\mathbf{b}} \oplus P_{\mathbf{p}}) \tag{4.4}$$

To demonstrate the advantage of processing whole sets S_i at once, consider the following example. If we process each vertex separately (so $\mathcal{S}$ consists of singletons only, $s_i = 1$ for all $i = 1, 2, .., r$), at each step we have to consider $\tau + 2$ one-term segments for $\mathbf{b} \oplus \mathbf{p}$ $(0, 1, \ldots, \tau + 1)$. Suppose now that our input graph has a perfect matching $\mathcal{S}$. Then, by processing two adjacent vertices at once, we can omit all segments of the form aa for $a \in \{2, 3, \ldots, \tau + 1\}$, since they are not feasible for any S_i. This is because each of values in $\{2, 3, \ldots, \tau+1\}$ describes a single label and no two adjacent vertices can get the same label (since $0 \in t(e)$ for every edge e). Therefore instead of considering all $(\tau+2)^2$ two-element segments in $[\tau+1] \times [\tau+1]$ (as we would do when processing each vertex separately), we have to deal with $(\tau+2)^2 - \tau$ two-element segments. One can observe that the choice of the partition $\mathcal{S}$ is crucial for this approach. This issue will be discussed in Section 4.2.2. From now on we will assume that each s_i (for $i = 1, 2, .., r$) is bounded by some constant value.

We can rewrite the formula (4.4) in the following way.

$$\overline{T}[k] \oplus P = \bigcup_{\substack{\mathbf{a} \in [\tau+1]^{s_1} \\ \mathbf{p} \in \{0,1\}^{s_1}}} \mathbf{a} \left[\left(\bigcup_{\substack{\mathbf{b} \in [\![\tau+1]\!]^{s_1} \\ \text{s.t. } \mathbf{b} \oplus \mathbf{p} = \mathbf{a}}} \overline{T}[k]_{\mathbf{b}} \right) \oplus P_{\mathbf{p}} \right] \tag{4.5}$$

The computation can be omitted whenever the vector $\mathbf{a}$ is not k-feasible for S_1, because then it cannot appear in $\overline{T}[k] \oplus P = T[k+1]$ as the 1st segment. Observe that $\mathbf{p}$ is uniquely determined by the choice of $\mathbf{a}$: we have $\mathbf{p}_i = 1$ whenever $\mathbf{a}_i = \tau + 1$ (and $\mathbf{p}_i = 0$ otherwise). See the pseudo-code of the Algorithm 2 for a more formal description of just presented computation of $T[k+1]$. The input arguments are: a graph G, a partition $(S_1, S_2, .., S_r)$ of its vertex set, the previously computed set $\overline{T}[k]$ and the set P of encodings of independent sets in G.

Algorithm 2: Compute$(G, (S_1, S_2, .., S_r), \overline{T}[k], P)$

```
1  **if** $r = 0$ **then** **return** $\emptyset$
2  $T[k+1] \leftarrow \emptyset$
3  **foreach** $\mathbf{a}$ *being* $(k+1)$*-feasible for* $S_1$ **do**
4      $\mathbf{p} \leftarrow$ the characteristic vector of the set $\{v_i \colon \mathbf{a}_i = \tau + 1; 1 \le i \le s_1\}$
5      $A \leftarrow \emptyset$
6      **foreach** $\mathbf{b} \in [\![\tau + 1]\!]^{s_1}$ *such that* $\mathbf{b} \oplus \mathbf{p} = \mathbf{a}$ **do**
7          $A \leftarrow A \cup \overline{T}[k]_{\mathbf{b}}$
8      **if** $A \neq \emptyset$ **then**
9          $Q \leftarrow$ **Compute**$(G - S_1, (S_2, .., S_r), A, P_{\mathbf{p}})$
10         $T[k+1] \leftarrow \{\mathbf{av} \colon \mathbf{v} \in Q\}$
11 **return** $T[k+1]$
```

Let us estimate the computational complexity of this algorithm. Recall that both τ and s_i (for all i) are constant values (while r depends on n). Thus line 3 is executed in a constant time. The number of iterations of the main loop in the lines 3–10 is equal to the number of vectors, which are $(k+1)$-feasible for S_1.

Let $f_i(k, \Lambda)$ denote the number of vectors from $[\tau + 1]^{s_i}$, which are k-feasible for S_i (for $i = 1, 2, .., r$). It is clearly an upper bound for the number of i-th segments in $T[k]$. Note that the value of $f_i(k, \Lambda)$ depends on G and $\mathcal{S}$, but we shall not write this to simplify the notation.

It is not hard to observe that for any choice of Λ we have $f_i(k, \Lambda) \le f_i(k, \Lambda')$, where $\Lambda'(v) = \{1, 2, .., \Lambda_{max}\}$ for every $v \in V$. Thus we define:

$$f_i := \max_{k \le \Lambda_{max}} f_i(k, \Lambda') \le (\tau + 2)^{s_i}. \tag{4.6}$$

One can easily verify that $|T[k]| \le \prod_{i=1}^{r} f_i$ for all $k \le \Lambda_{max}$.

Notice that if we fix $\mathbf{a}$, then $\mathbf{p}_i = 1$ whenever $\mathbf{a}_i = \tau + 1$ and $\mathbf{p}_i = 0$

otherwise. Since s_1 is a constant, the execution time of line 4 is also constant.

In each recursive computation in our algorithm we have to prepare up to f_i pairs of sets of vectors of length $n - s_i$ and compute $\oplus$ on these pairs. The time needed to prepare the set $A = \bigcup_{\substack{\mathbf{b} \in [\![\tau+1]\!]^{s_1} \\ \text{s.t. } \mathbf{b} \oplus \mathbf{p} = \mathbf{a}}} \overline{T}[k]_{\mathbf{b}}$ (in lines 6–7) is at most $\mathcal{O}(n \cdot |\overline{T}[k]|)$. Indeed, for each choice of $\mathbf{b}$ (we have a constant number of such choices) and each $\mathbf{b}' \in \overline{T}[k]$ we have to check if $\mathbf{b}$ is a prefix of $\mathbf{b}'$ and (if so) include the remaining segment of $\mathbf{b}'$ into A. In an analogous way, preparing the set $P_{\mathbf{p}}$ requires time $\mathcal{O}\left(n \cdot |P|\right)$. Thus preparing the recursive calls in line 9 takes time $\mathcal{O}\left(n \cdot (|P| + |\overline{T}[k]|)\right)$. Recall that $|\overline{T}[k]| = |T[k]| \leq \prod_{i=1}^{r} f_i$.

Now we will show that $|P| \leq \prod_{i=1}^{r} f_i$. We shall define an injective function from P into the set of proper partial labelings for the instance (G, Λ', t), using only label 1. For any vector $\mathbf{p} \in P$ representing an independent set X in G we assign a partial labeling in which vertices of X receive labels 1 and the remaining vertices are unlabeled (there are no conflicts, because they may appear only on edges with both endvertices already labeled). Clearly this assignment is an injection. Each labeling obtained in this way (for different sets X) has a different encoding in the set $T[1]$ (for lists Λ'). Thus the size of P does not exceed $\prod_{i=1}^{r} f_i$.

Finally, executing line 10 requires appending every vector $\mathbf{v} \in Q$ to $\mathbf{a}$, which can be done it time $\mathcal{O}(|Q|) = \mathcal{O}(|T[k+1]|)$.

So the time $F'(G, \mathcal{S})$ needed to compute $\overline{T}[k] \oplus P$ in the way shown by the Algorithm 2 is given by the recursive formula (4.7). Recall that in every step we remove S_1 from $\mathcal{S}$ and the index of every remaining set in $\mathcal{S}$ is reduced by one, so that the first one is still called S_1. Also for $\mathcal{S} = (S_1, S_2, \ldots, S_r)$ by $\mathcal{S} - S_1$ we denote the tuple $(S_2, S_3, \ldots, S_r)$.

$$\begin{aligned}F'(G,\mathcal{S}) \leq & \underbrace{c_1}_{\text{lines } 1-3} + f_1 \cdot \Bigg(\underbrace{c_2}_{\text{lines 4–5, 8}} + \underbrace{c_3 \cdot n \cdot |T[k]|}_{\text{lines 6–7}} + \underbrace{c_4 \cdot n \cdot |P|}_{\text{prepare } P_{\mathbf{P}}} \\ & + \underbrace{F'(G - S_1, \mathcal{S} - S_1)}_{\text{line 9}} + \underbrace{c_5 \cdot |T[k+1]|}_{\text{line 10}} \Bigg) \\ & \leq c_1 + f_1 \cdot \left(c_2 + c_6 \cdot n \cdot \prod_{i=1}^{r} f_i + F'(G - S_1, \mathcal{S} - S_1) \right) = \\ & \leq c_7 + c_8 \cdot n \cdot \prod_{i=1}^{r} f_i + f_1 \cdot F'(G - S_1, \mathcal{S} - S_1) \\ & \leq c' \cdot n \cdot \prod_{i=1}^{r} f_i + f_1 \cdot F'(G - S_1, \mathcal{S} - S_1)\end{aligned} \tag{4.7}$$

for some constants c_i (for $i = 1,..,8$) and c'. We can verify by induction that the above inequality implies that $F'(G,\mathcal{S}) \leq c' \cdot n^2 \cdot \prod_{i=1}^{r} f_i$ for large n. Indeed, by induction hypothesis we have:

$$\begin{aligned}F'(G,\mathcal{S}) & \leq c' \cdot n \cdot \prod_{i=1}^{r} f_i + f_1 \cdot F'(G - S_1, \mathcal{S} - S_1) \\ & \leq c' \cdot n \cdot \prod_{i=1}^{r} f_i + \left(f_1 \cdot c' \cdot (n - s_1)^2 \cdot \prod_{i=2}^{r} f_i \right) \\ & = c' \cdot n^2 \cdot \prod_{i=1}^{r} f_i \cdot \left(\frac{1}{n} + \frac{(n - s_1)^2}{n^2} \right) \\ & = c' \cdot n^2 \cdot \prod_{i=1}^{r} f_i \cdot \underbrace{\left(1 - \frac{(2s_1 - 1)n + s_1^2}{n^2} \right)}_{\leq 1} \leq c' \cdot n^2 \cdot \prod_{i=1}^{r} f_i\end{aligned}$$

Recall that $\overline{T}[k]$ can be obtained from $T[k]$ in time $\mathcal{O}(|T[k]| \cdot n^2) = \mathcal{O}(n^2 \cdot \prod_{i=1}^{r} f_i)$. Since we need to compute sets $T[k]$ for all $k \leq \Lambda_{max}$, we obtain the following bound $F(G, \mathcal{S})$ on the total running time of Algorithm 1 ($c_1, c_2, .., c_5$ are constants).

$$\begin{aligned}
F(G,\mathcal{S}) \leq & \underbrace{c_1 \cdot n^2 \cdot |P|}_{\text{line 1}} + \underbrace{c_2}_{\text{line 2}} + \Lambda_{max} \cdot \left(\underbrace{c_3 \cdot n^2 \cdot \prod_{i=1}^{r} f_i}_{\text{line 4}} + \underbrace{F'(G,\mathcal{S})}_{\text{line 5}} \right) + \underbrace{c_4 \cdot n \cdot |T[\Lambda_{max}]|}_{\text{lines 6–7}} \\
\leq & c_1 \cdot n^2 \cdot \prod_{i=1}^{r} f_i + c_2 + \Lambda_{max} \cdot \left(c_3 \cdot n^2 \cdot \prod_{i=1}^{r} f_i + c' \cdot n^2 \cdot \prod_{i=1}^{r} f_i \right) + c_4 \cdot n \cdot \prod_{i=1}^{r} f_i \\
\leq & c_5 \cdot \left(\Lambda_{max} \cdot n^2 \cdot \prod_{i=1}^{r} f_i \right).
\end{aligned}$$

Recall that by (4.2) we have $\Lambda_{max} \leq (2\tau + 1)(\tau + 2)n^2$. Thus we obtain the following bound (c is some constant):

$$F(G,\mathcal{S}) \leq c \cdot n^4 \cdot \prod_{i=1}^{r} f_i. \tag{4.8}$$

The space complexity of the algorithm is determined by the total size of vectors in the sets $T[k]$ and the set P. Each of those sets contains at most $\prod_{i=1}^{r} f_i$ vectors of length n each. The number of sets $T[k]$ is Λ_{max}. Thus the space complexity is bounded by $c'' \cdot n^3 \cdot \prod_{i=1}^{r} f_i$ (for a constant c'').

Remark 4.6. By some additional remembering (i.e. sets $T[k]$ should contain pairs consisting of a vector and a corresponding partial labeling instead of just vectors) we can easily adapt the algorithm to find a proper labeling (if there exists one). The space and computational complexity of the modified version is asymptotically the same as of the basic one, up to a polynomial factor of n.

4.2.2 Complexity bounds

In this section we shall consider several possible partitions $\mathcal{S}$ of the vertex set and use them to bound the complexity of the algorithm described in the preceding section with functions of various invariants of G. Let us start with the simplest case, i.e. a partition into singletons.

Theorem 4.7 ([59]). *The algorithm Solve-GLTC solves the τ-bounded generalized list T-coloring problem for a graph G with n vertices in time and space $\mathcal{O}^*((\tau+2)^n)$.*

Proof. Let $v_1, v_2, \ldots, v_n$ be an arbitrary ordering of vertices of G and let $S_i = \{v_i\}$ for $i \in \{1, .., n\}$. For any graph G we trivially get $f_i \leq \tau + 2$, as there are at most $\tau + 2$ vectors $\mathbf{a}$ of length 1, which are feasible for S_i (for any $i = 1, 2, .., n$). Using the formula (4.8) we obtain the bound for the complexity $F(G, \mathcal{S}) \leq c \cdot n^4 \cdot \prod_{i=1}^{n} (\tau + 2) = c \cdot n^4 \cdot (\tau + 2)^n$. $\square$

Although the algorithm by Cygan and Kowalik [21] is designed for the channel assignment problem, it can be adapted to solve the generalized T-coloring problem within the same time bound, i.e. $\mathcal{O}^*((\tau+2)^n)$. However, we can improve the time complexity of our algorithm by an appropriate construction of the partition $\mathcal{S}$. Let us start with $\mathcal{S}$ being a star partition (i.e. partition $\mathcal{S} = (S_1, S_2, .., S_r)$ such that for every $i \in \{1, 2, .., t\}$ the graph $G[S_i]$ has a spanning subgraph, which is a star, see Section 3.5.2 in Chapter 3).

Lemma 4.8 ([59]). *Let G be a graph with n vertices and let $\mathcal{S} = \{S_1, S_2, \ldots, S_r\}$ be its a star partition such that $2 \leq s_i \leq D + 1$ for some constant D and each $i \in 1, 2, \ldots, r - 1$ and $s_r \leq D'$ for some constant D'. The algorithm Solve-GLTC solves the τ-bounded generalized list T-coloring problem on G in time and space*

$$\mathcal{O}^* \left((\tau^2 + 3\tau + 4)^{n/2} + (2(\tau+2)^D + \tau(\tau+1)^D)^{\frac{n}{D+1}} \right).$$

Proof. Let us consider the subgraph $G[S_i]$ for some $i \in \{1, 2, .., r\}$. Without loss of generality let v_j be the central vertex of a spanning star of $G[S_i]$ and $S_i = \{v_j, .., v_{j+s_i-1}\}$. Note that by the definition of $T[k]$, for any $\mathbf{a} \in T[k]$, if $\mathbf{a}_j = h \in \{2, .., \tau + 1\}$, then $\mathbf{a}_{j'} \neq h$ for $j' \in \{j + 1, .., j + s_i - 1\}$, since h represents a single label and each label of any feasible (possibly partial) labeling induces an independent set in G. Hence the number of vectors, which are feasible for S_i is at most

$$f_i \leq \underbrace{2(\tau+2)^{s_i-1}}_{\mathbf{a}_j \in \{0,1\}} + \underbrace{\tau(\tau+1)^{s_i-1}}_{\mathbf{a}_j \notin \{0,1\}}.$$

This inequality combined with the formula (4.8) gives us the following bound on the complexity (c and c_1 are constants):

$$\begin{aligned}
F(G,\mathcal{S}) \leq & c \cdot n^4 \cdot \prod_{i=1}^{r} f_i \\
\leq & c \cdot n^4 \cdot \prod_{i=1}^{r} \left(2(\tau+2)^{s_i-1} + \tau(\tau+1)^{s_i-1}\right) \\
\leq & c \cdot n^4 \cdot \prod_{i=1}^{r-1} \left(2(\tau+2)^{s_i-1} + \tau(\tau+1)^{s_i-1}\right) \underbrace{\left(2(\tau+2)^{D'-1} + \tau(\tau+1)^{D'-1}\right)}_{\text{constant}} \\
\leq & c_1 \cdot n^4 \cdot \prod_{i=1}^{r-1} \left(2(\tau+2)^{s_i-1} + \tau(\tau+1)^{s_i-1}\right).
\end{aligned}$$

One can verify that the value of $\prod_{i=1}^{r-1} \left(2(\tau+2)^{s_i-1} + \tau(\tau+1)^{s_i-1}\right)$ is maximized if all but at most one of $s_1, s_2, \ldots, s_{r-1}$ have value 2 or $D+1$. Consider this case and define p (q, respectively) to be the number of s_i's equal to 2 ($D+1$, respectively). Formally, we have $p := |\{i \in \{1,2,..,r-1\} \colon s_i = 2\}|$ and $q := |\{i \in \{1,2,..,r-1\} \colon s_i = D+1\}|$. Clearly $2p + (D+1)q \leq n$ and thus $p \leq \frac{n-(D+1)q}{2}$. Thus we obtain the following inequality (c' is a constant):

$$\begin{aligned}
F(G,\mathcal{S}) \leq & c_1 \cdot n^4 \prod_{i=1}^{r-1} \left(2(\tau+2)^{s_i-1} + \tau(\tau+1)^{s_i-1}\right) \\
\leq & c' \cdot n^4 \cdot \left(\prod_{i=1}^{p} (2(\tau+2) + \tau(\tau+1))\right) \left(\prod_{i=1}^{q} \left(2(\tau+2)^D + \tau(\tau+1)^D\right)\right) \\
= & c' \cdot n^4 \cdot (2(\tau+2) + \tau(\tau+1))^p \left(2(\tau+2)^D + \tau(\tau+1)^D\right)^q \\
\leq & c' \cdot n^4 \cdot (2(\tau+2) + \tau(\tau+1))^{\frac{n-(D+1)q}{2}} \left(2(\tau+2)^D + \tau(\tau+1)^D\right)^q
\end{aligned}$$

Using standard methods one can see that depending on the values of D and τ, the expression above is either increasing or decreasing (with respect to q). Thus the maximum value is obtained either for $q = 0$ or for $q = \frac{n}{D+1}$ and we can bound $F(G,\mathcal{S})$ by a sum of those two potential maximal values.

$$F(G,\mathcal{S}) \leq c' \cdot n^4 \cdot \left((\tau^2 + 3\tau + 4)^{n/2} + (2(\tau+2)^D + \tau(\tau+1)^D)^{\frac{n}{(D)}}\right)$$

□

Here we shall use the results we proved in Section 3.5 in Chapter 3.

Corollary 4.9 (from Lemma 3.32, [59]). *The algorithm Solve-GLTC solves the τ-bounded generalized list T-coloring problem on G with maximum degree bounded by a constant Δ in time and space*

$$\mathcal{O}^*\left((\tau^2+3\tau+4)^{n/2}+(2(\tau+2)^{\Delta-1}+\tau(\tau+1)^{\Delta-1})^{n/\Delta}\right).$$

Theorem 3.33 gives us a a significantly better bound for regular graphs (in fact in works for much wider class of graph G with $\Delta(G) \leq 2\delta(G)$). By some technical calculations we obtain the following bound.

Corollary 4.10 (from Theorem 3.33, [59]). *The algorithm Solve-GLTC solves the τ-bounded generalized list T-coloring problem on a regular graph with n vertices in time and space $\mathcal{O}^*\left((\tau^2+3\tau+4)^{n/2}\right)$.*

Now let us turn our attention to $K_{1,d}$-free graphs.

Corollary 4.11 (from Lemma 3.35, [59]). *The algorithm Solve-GLTC solves the τ-bounded generalized list T-coloring problem on a $K_{1,d}$-free graph with n vertices in time and space*

$$\mathcal{O}^*\left((\tau^2+3\tau+4)^{n/2}+(2(\tau+2)^{d-2}+\tau(\tau+1)^{d-2})^{n/(d-1)}\right).$$

When we apply this bound to claw-free graphs (i.e. $K_{1,3}$-free graphs), we obtain the following.

Corollary 4.12 ([59]). *The algorithm Solve-GLTC solves the τ-bounded generalized list T-coloring problem on a claw-free graph with n vertices in time and space $\mathcal{O}^*\left((\tau^2+3\tau+4)^{n/2}\right)$.*

Another class of graphs we want to mention here are unit disk graphs, i.e. intersection graphs of unit disks on a plane (see for example Clark *et al.* [19] for more information). They are particularly interesting due to their applications in modelling of ad-hoc networks. It is widely known that unit disk graphs are $K_{1,6}$-free [42]. By some technical calculations, we obtain the following corollary.

Corollary 4.13 ([59]). *The algorithm Solve-GLTC solves the τ-bounded generalized list T-coloring problem on a unit disk graph with n vertices in time and space $\mathcal{O}^*\left((\tau^2+3\tau+4)^{n/2}\right)$.*

Subgraphs other than stars may also be useful in constructing the partition $\mathcal{S}$. Hell and Kirkpatrick [47] studied the problem of partitioning the vertex set of a graph into cliques. Let a *clique packing* into a graph G be a subgraph of G, whose every connected component is a clique with at least 2 vertices. Let $\rho(G)$ denote the order of the largest (in terms of the number of vertices) clique packing in G. Note that a matching is a special case of a clique packing of G. Therefore, if m is the size of maximum matching in G, then we have: $2m \leq \rho(G)$. A spanning clique packing is called a *clique partition*. Hell and Kirkpatrick [47] presented an elegant structural theorem allowing to compute the value of $\rho(G)$. Moreover, they described graphs G that have a clique partition.

Theorem 4.14 ([59]). *The algorithm Solve-GLTC solves the τ-bounded generalized list T-coloring problem for a graph G with n vertices in time and space $c \cdot n^4 \cdot \left((\tau^2 + 3\tau + 4)^{\rho(G)/2}(\tau + 2)^{n-\rho(G)}\right)$, for some constant c.*

Proof. Let H be the largest clique packing into G. Consider a spanning subgraph H' of H in which every connected component is isomorphic to K_2 or K_3. One can clearly obtain such a subgraph – the vertex set of a connected component of H with even number of vertices is partitioned into single edges (a perfect matching), while each component with odd number of vertices is partitioned into single edges and exactly one triangle.

Let p be the number of connected components of H', which are isomorphic to K_2 and let q be the number of connected components of H', which are isomorphic to K_3. Clearly $2p+3q = \rho(G)$. Let $\mathcal{S} = \{S_1, .., S_r\}$ be a partition of $V(G)$, such that the sets S_i for $i \leq p$ correspond to K_2-components in H' and the sets S_i for $p < i \leq p+q$ correspond to K_3-components of H'. The remaining sets S_i, for $i > p+q$, contain the remaining vertices of G, one vertex per set.

Let us consider the subgraph $G[S_i]$ for some $i \leq p+q$. Since it is a clique, each vertex has to be labeled with a different label. Therefore in any vector which is feasible for S_i, each element from $\{2, .., \tau + 1\}$ may appear at most once. Recall that for S_i inducing an edge (i.e. for $i \leq p$) we have at most $(\tau + 2)^2 - \tau = \tau^2 + 3\tau + 4$ feasible vectors.

Now let us consider vectors, which are feasible for S_i, corresponding to triangles (i.e. $p < i \leq p+q$). There are:

- $2^3 = 8$ vectors with no element from $\{2, .., \tau + 1\}$,
- $3\tau \cdot 2^2 = 12 \cdot \tau$ vectors with exactly one element from $\{2, .., \tau + 1\}$,

- $3\tau(\tau-1)\cdot 2 = 6\cdot\tau(\tau-1)$ vectors with exactly two elements from $\{2,..,\tau+1\}$,
- $\tau(\tau-1)(\tau-2)$ vectors with exactly three elements from $\{2,..,\tau+1\}$.

For each of the remaining sets S_i (containing single vertices, i.e. for $i > p+q$), there are $\tau+2$ feasible vectors. Hence

$$f_i \leq \begin{cases} \tau^2+3\tau+4 & \text{for } i \leq p \\ \tau^3+3\tau^2+8\tau+8 & \text{for } p < i \leq p+q \\ \tau+2 & \text{for } i > p+q. \end{cases}$$

Again, using formula (4.8), we obtain the following.

$$\begin{aligned} F(G,\mathcal{S}) \leq & c\cdot n^4 \cdot \prod_{i=1}^{r} f_i \\ \leq & c\cdot n^4 \cdot \left((\tau^2+3\tau+4)^p(\tau^3+3\tau^2+8\tau+8)^q(\tau+2)^{n-\rho(G)}\right) \\ = & c\cdot n^4 \cdot \left((\tau^2+3\tau+4)^{\frac{\rho(G)-3q}{2}}(\tau^3+3\tau^2+8\tau+8)^q(\tau+2)^{n-\rho(G)}\right) \end{aligned}$$

This expression is maximized for $q=0$. So finally we obtain the bound:

$$F(G,\mathcal{S}) \leq c\cdot n^4 \cdot \left((\tau^2+3\tau+4)^{\rho(G)/2}(\tau+2)^{n-\rho(G)}\right).$$

$\square$

When we apply this bound to a graph with a clique partition (e.g. a perfect matching), we obtain the following.

Corollary 4.15. *The algorithm Solve-GLTC solves the τ-bounded generalized list T-coloring problem for a graph G with n vertices, which has a clique partition, in time and space $\mathcal{O}^*\left((\tau^2+3\tau+4)^{n/2}\right)$.*

The Table 4.1 compares the complexity bounds (more precisely, the bases of the exponential factor) of the algorithm Solve-GLTC applied to various graph classes for some values of τ.

τ	general graphs	graphs with $\Delta \leq 3$	regular graphs	graphs with a clique partition	claw-free graphs	unit disk graphs
1	3.0000	2.8021	2.8285			
2	4.0000	3.6841	3.7417			
3	5.0000	4.6105	4.6905			
4	6.0000	5.5613	5.6569			
5	7.0000	6.5266	6.6333			

Table 4.1: Comparison of bases of the exponential factor in the complexity bound.

4.3 Exact algorithm for the $L(2,1)$-labeling problem

Recall that the $L(2,1)$-labeling problem can be seen as a special case of the 1-bounded generalized list T-coloring problem (see Section 4.1). Thus Theorem 4.7 immediately gives us a complexity bound of $\mathcal{O}^*(3^n)$ for the $L(2,1)$-labeling problem. However, we can improve it significantly with a better analysis of possible segments that can appear in $T[k]$. We shall use previously proven results on covering a graph with connected subgraphs (see Section 3.5.1) and the bounds for the number of proper pairs in a graph (see Section 3.3). The following theorem is the main result of this section.

Theorem 4.16 ([53]). *The $L(2,1)$-labeling problem on a graph with n vertices can be solved in time and space $\mathcal{O}(2.6488^n)$.*

Proof. Assume that G is a connected graph (otherwise we can label its connected components separately). Let ℓ be a large fixed constant (we will explain later what exactly the value of ℓ should be). By Theorem 3.31 there exist connected subgraphs $G_1, G_2, \ldots, G_q$ of G, such that:

1. $V(G) \subseteq \bigcup_{i=1}^{q} V(G_i)$,
2. $\ell \leq |V(G_i)| \leq 2\ell$ for $i \in \{1, 2, .., q-1\}$ and $|V(G_q)| \leq 2\ell$,
3. $n' := \sum_{i=1}^{q} |V(G_i)| \leq n(1 + \frac{1}{\ell})$.

Let $S_i = V(G_i)$ for every $i \in \{1, 2, .., q\}$ and let $\mathcal{S} = (S_1, S_2, .., S_q)$. For $i \in \{1, 2, .., q\}$, define $s_i := |S_i|$. Note that some vertices may appear in more than one set S_i, so $\mathcal{S}$ does not have to be a partition of $V(G)$. We form a sequence of vertices of G according to $\mathcal{S}$, just as we did before (see (4.3)) in Section 4.2). The length of this sequence is n' and some vertices may appear in it several times. However, this is not a problem for our algorithm. When constructing partial labeling, we need to check if all values of coordinates corresponding to the same vertex are equal. If not, the algorithm just removes such a vector from the current set.

Again, we shall process vertices in groups corresponding to sets S_i. From the formula (4.8) we obtain that the time complexity of this procedure is bounded by:

$$F(G,\mathcal{S}) \leq c \cdot n^4 \cdot \prod_{i=1}^{q} f_i, \tag{4.9}$$

where f_i is defined by the formula (4.6) and c is a constant.

Moreover, observe that a single label in an (possibly partial) $L(2,1)$-labeling of G induces a 2-packing. Thus the set P will now contain characteristic vectors of all 2-packings in G. Moreover, considering a vector $\mathbf{a} \in \{0,1,2\}^{s_i}$ being a k-feasible for S_i, notice that the set $\{v_j \colon \mathbf{a}_j = 2\}$ is a 2-packing in G_i (since the the value 2 corresponds to a single label). The labels 0 and 1 can be distributed in an arbitrary way on the remaining vertices of G_i. Thus we obtain the following bound:

$$f_i \leq \sum_{j=0}^{s_i} u_j(G_i) \cdot 2^{s_i - j} \tag{4.10}$$

(recall that $u_j(G_i)$ denotes the number of j-element 2-packings in G_i).

Observe that the right hand side of (4.10) is exacly the expresion (3.4), appearing in the definition $pp(G_i)$, i.e. it is the number of proper pairs in G_i. Thus,

$$f_i \leq \sum_{j=0}^{s_i} u_j(G_i) \cdot 2^{s_i - j} = pp(G_i). \tag{4.11}$$

Recall from Theorem 3.19 that $pp(G_i) \leq x^{s_i}$ for $x = 2.64878$, provided that $j \leq s_i$ is large enough. Combining formulae (4.9) and (4.11), we obtain the following complexity bound for G with n vertices:

$$\begin{aligned} F(G,\mathcal{S}) \leq & c \cdot n'^4 \cdot \prod_{i=1}^{q} f_i \leq c \cdot n'^4 \cdot \prod_{i=1}^{q} pp(G_i) \leq c \cdot n'^4 \cdot \prod_{i=1}^{q} x^{s_i} \\ \leq & c \cdot n'^4 \cdot x^{n'} = c \cdot (n(1+1/\ell))^4 \cdot \left(x^{n(1+1/\ell)}\right). \end{aligned} \tag{4.12}$$

We arrive at our main result by choosing the constant ℓ so big, that $x^{1+1/\ell} = 2.64878^{1+1/\ell} \leq 2.6488$. For such ℓ we obtain the following bound (for any connected graph G with n vertices):

$$F(G, \mathcal{S}) \leq c \cdot n^4 \cdot 2.6488^n.$$

$\square$

Recall that in Section 3.3 we have found better bounds for the maximum number of proper pairs in claw-free and r-dominated graphs. These results, by formula (4.12), automatically allow us to solve the $L(2,1)$-labeling problem faster (we proceed in a way analogous to the proof of Theorem 4.16).

Corollary 4.17 (from Theorem 3.21, [53]). *The $L(2,1)$-labeling problem on a claw-free graph with n vertices can be solved in time and space $\mathcal{O}(2.5944^n)$.*

Corollary 4.18 (from Corollary 3.22, [53]). *The $L(2,1)$-labeling problem on an r-dominated graph with n vertices can be solved in time and space $\mathcal{O}^*\left(2^{n-r}\left(2+\frac{n}{r}\right)^r\right)$.*

Observe that if $\frac{n}{r} > 3.7729$, the bound from Corollary 4.18 is better than the bound from Theorem 4.16.

4.4 Exact algorithm for the graph homomorphism problem

In this section we focus on another generalization of many graph coloring problems, i.e. the *graph homomorphism* problem.

If φ is a homomorphism from G to H, then we will write $\varphi\colon G \to H$ shortly. A *partial homomorphism* from G to H is a homomorphism from an induced subgraph of G to H (in other words, we allow that some vertices of G are not mapped to any vertices of H).

The *bandwidth* of a graph $G = (V, E)$, denoted by $\mathrm{bw}(G)$, is the minimum of the values $\max\{|i-j|\colon v_iv_j \in E(G)\}$ over all orderings $(v_1, v_2, \dots, v_n)$ of V. Informally speaking, we want to place the vertices of G in integer points of a number line in such a way, that the longest edge is as short as possible (see for example [16, 18]).

The following theorem is the main result of this section.

Theorem 4.19 ([78]). *The existence of a homomorphism from G to H can be decided in time $\mathcal{O}^*\left((\mathrm{bw}(\overline{H})+2)^n\right)$, where n is the number of vertices of G and $\mathrm{bw}(\overline{H})$ is the bandwidth of the complement of H.*

Proof. Let $V(G) = \{v_1, v_2, .., v_n\}$ and $V(H) = \{h_1, h_2, .., h_m\}$. The ordering of vertices in G is arbitrary. The vertices of H are arranged in the order corresponding to the bandwidth of $\overline{H}$, i.e. in such a way that the value $\max(\{|i-j|\colon h_ih_j \notin E(H)\}$ is minimum possible (by the definition, this minimum value is equal to $\mathrm{bw}(\overline{H})$). Let $\beta = \mathrm{bw}(\overline{H}) + 1$ and let $H_k = H[\{h_1, h_2, \dots, h_k\}]$ for any $k \in \{1, 2, \dots, m\}$.

Analogically to the algorithm for the Generalized list T-coloring, for every $k \in \{1, .., m\}$ we introduce the set of vectors $T[k] \subseteq [\beta]^n$. Now $\mathbf{a} \in T[k]$ if and only if there exists a partial homomorphism $\varphi\colon G \to H_k$, such that:

$$\mathbf{a}_i = \begin{cases} 0 & \text{if } v_i \text{ is not mapped,} \\ 1 & \text{if } \varphi(v_i) = h_\ell, \text{ where } 1 \le \ell \le k - \beta + 1, \\ \ell + \beta - k & \text{if } \varphi(v_i) = h_\ell, \text{ where } \ell > k - \beta + 1. \end{cases}$$

Observe that if $\ell > k - \beta + 1$, then $\mathbf{a}_i \in \{2, .., \beta\}$. Again, we define $T[0] := \{0^n\}$. Note that vectors $\mathbf{a}$ with no 0's correspond to homomorphisms $G \to H_k$. Therefore there exists a homomorphism $G \to H$ if and only if $T[m] \cap \{1, 2, \dots, \beta\}^n \neq \emptyset$.

The set P still contains characteristic vectors of all independent sets in G. One more thing we need to change is the definition of $\overline{T}[k]$. Let $k \in \{1, \ldots, m-1\}$ be fixed and assume we have already computed the set $T[k]$. Let $\mathbf{a}$ be a vector from $T[k]$. Then we define $\overline{\mathbf{a}} \in [\![\beta]\!]^n$ as follows:

$$\overline{\mathbf{a}}_i = \begin{cases} 0 & \text{if } \mathbf{a}_i = 0 \text{ and there is no } v_j \in N_G(v_i), \text{ such that} \\ & \quad \mathbf{a}_j \geq 2 \text{ and } h_{k-\beta+\mathbf{a}_j} \notin N_H(h_{k+1}), \\ \bar{0} & \text{if } \mathbf{a}_i = 0 \text{ and there exists } v_j \in N_G(v_i), \text{ such that} \\ & \quad \mathbf{a}_j \geq 2 \text{ and } h_{k-\beta+\mathbf{a}_j} \notin N_H(h_{k+1}), \\ \mathbf{a}_i & \text{if } \mathbf{a}_i \in \{1, .., \beta\}. \end{cases}$$

Consider $\mathbf{a} \in T[k]$ (for $k \leq m-1$) and a partial homomorphism φ corresponding to $\mathbf{a}$. Notice that if v_i is not mapped by φ, then $\overline{\mathbf{a}}_i$ is either equal to 0 or to $\bar{0}$. We have $\overline{\mathbf{a}}_i = 0$ if and only if φ can be extended by mapping v_i to h_{k+1}. In the other case, $\overline{\mathbf{a}}_i$ is equal to $\bar{0}$. Since the vertices of H are arranged „according to $\mathrm{bw}(\overline{H})$", all non-neighbors of h_{k+1} are in $\{h_{k-\mathrm{bw}(\overline{H})+1}, h_{k-\mathrm{bw}(\overline{H})+2}, \ldots, h_k\}$. This is why we do not have to distinguish the vertices mapped to $h_1, h_2 \ldots, h_{k-\mathrm{bw}(\overline{H})}$ in our representation of partial homomorphisms.

The definition of $\oplus$ remains unchanged (with $\tau+1$ replaced by β). Also the way of computing $\overline{T}[k] \oplus P$ is the same as it was introduced in Section 4.2. For simplicity, we shall only consider a partition $\mathcal{S}$ of $V(G)$ to singletons (so $s_i = 1$ for $i = 1, 2, .., n$). Thus in our case, the formula (4.4) we use for computation has the following form:

$$\begin{aligned} \overline{T}[k] \oplus P &= \bigcup_{\substack{b \in [\![\beta]\!], p \in \{0,1\} \\ \text{s.t. } b \oplus p \text{ is defined}}} (b \oplus p)(\overline{T}[k]_b \oplus P_p) \\ &= 0 \left[(\overline{T}[k]_0 \cup \overline{T}[k]_{\bar{0}}) \oplus P_0 \right] \cup 1 \left[\left(\overline{T}[k]_1 \cup \overline{T}[k]_2 \right) \oplus P_0 \right] \\ &\cup \bigcup_{a \in \{2, .., \beta-1\}} a \left[\overline{T}[k]_{a+1} \oplus P_0 \right] \cup \beta \left[\overline{T}[k]_0 \oplus P_1 \right]. \end{aligned}$$

Using this formula we see that to compute $\oplus$ on two sets of vectors of length n, we have to compute $\oplus$ on $\beta+1$ pairs of sets of vectors of length $n-1$. The size of P is at most 2^n and the size of $T[k]$ is at most $(\beta+1)^n$. Note that since $\beta = \mathrm{bw}(\overline{H})+1$, we have $\beta+1 = \mathrm{bw}(\overline{H})+2 \geq 2$. Recall that computing $\overline{T}[k]$ takes time at most $\mathcal{O}(|T[k]| \cdot n^2) = \mathcal{O}(n^2 \cdot (\beta+1)^n)$. Using the argumentation similar to the one we used in Section 4.2.1, one can verify by induction that the time complexity of our algorithm is:

$$F(n) = \mathcal{O}^*\big((\mathrm{bw}(\overline{H}) + 2)^n\big).$$

□

We have mentioned in Section 4.1 that the problem of (m, k)-coloring of a graph G is equivalent to the problem of determining an existence of a homomorphism from G to $\overline{C}_m^{k-1}$. Since the complement of $\overline{C}_m^{k-1}$ is C_m^{k-1} and $\mathrm{bw}(C_m^{k-1}) = 2(k-1)$ (if $v_0, v_1, v_2, .., v_{m-1}$ is the ordering of vertices along C_m, then the optimal arrangements is $v_0, v_1, v_{m-1}, v_2, v_{m-2}, ..., v_{\lceil m/2\rceil}$, see for example [18]), we obtain the following.

Corollary 4.20 ([78])**.** *The (m, k)-coloring problem on a graph G with n vertices can be solved in time $\mathcal{O}^*((2k)^n)$.*

4.4.1 Exact algorithm for the locally injective graph homomorphism problem

Now we shall show how to further modify the the algorithm from the preceding section, to obtain an algorithm for the locally injective homomorphism problem, working within the same time bound.

Theorem 4.21 ([78])**.** *The existence of a locally injective homomorphism from G to H can be decided in time $\mathcal{O}^*\big((\mathrm{bw}(\overline{H}) + 2)^n\big)$, where n is the number of vertices of G and $\mathrm{bw}(\overline{H})$ is the bandwidth of the complement of H.*

Proof. We observe that the vertices of G that can be mapped to a single vertex of H (in a locally injective manner) must be at distance at least 3 from each other. Therefore they form a 2-packing. Since it is the only additional requirement for locally injective homomorphisms, the only thing that has to be changed in the algorithm from the preceding section is the initialization of the set P. Now it has to contain characteristic vectors of all 2-packings sets. □

Recall (see Theorem 4.1) that $H(2, 1)$-labelings are exactly locally injective homomorphisms to $\overline{H}$. Since the complement of $\overline{H}$ is H, we obtain the following corollary.

Corollary 4.22 ([78]). *For any graphs G and H we can solve the $H(2,1)$-labeling problem in time $\mathcal{O}^*\left((\mathrm{bw}(\overline{H})+2)^n\right)$, where n is the number of vertices of G.*

Let us see how this bound works for the $L_c(2,1)$-labeling problem (see Section 4.1 for the definitions of the problems). It is equivalent to finding the smallest m, such that the input graph admits a $C_{m+1}(2,1)$-labeling. We shall check the existence of such a labeling for $m = 3, .., 2n$ and stop when we find one. Since $\mathrm{bw}(\overline{C_m}) = 2$, we obtain the following.

Corollary 4.23 ([78]). *The $L_c(2,1)$-labeling problem on a graph G with n vertices can be solved in time $\mathcal{O}^*\left(4^n\right)$.*

Remark 4.24. The bounds presented in this section can be slightly improved, using the methods presented in Sections 4.2 and 4.3. However, it requires many technical calculations and the improvement gets smaller as $\mathrm{bw}(\overline{H})$ grows.

Let us conclude this section with some comparison of our algorithm with the previously known algorithms for the graph homomorphism problem [37] and the locally injective graph homomorphism problem [45]. Recall from Section 1.3 that the complexities of algorithms described in these papers are bounded by $\mathcal{O}^*((2\,\mathrm{tw}(H)+1)^n)$ and $\mathcal{O}^*((\Delta(H)-1)^n)$, respectively. Observe that both parameters in these bounds, i.e. the treewidth and the maximum degree, grow rapidly as the density of H increases. Therefore if H is extremely dense (for example obtained from a complete graph by removing a constant number of edges or a matching), both those algorithms may work very slowly. But in such cases the graph $\overline{H}$ is very sparse and its bandwidth is small, so our algorithm outperforms the algorithms described in [37] and [45] significantly. On the other hand, if H is sparse (so $\overline{H}$ is dense), $\mathrm{bw}\,\overline{H}$ is large and so is the complexity bound for our algorithm.

Chapter 5

Algorithms using polynomial space

In this chapter we present an exact algorithm for the $L(2,1)$-labeling problem, using only polynomial space, presented in the joint paper co-authored with K. Junosza-Szaniawski, J. Kratochvíl and Liedloff [54].

5.1 $L(2,1)$-labeling problem

The following theorem is the main result of this section.

Theorem 5.1 ([54])**.** *The $L(2,1)$-span of a graph on n vertices can be computed in time $\mathcal{O}(7.4920^n)$ and polynomial space.*

To prove this theorem, we shall consider the $L(2,1)$-labeling problem generalized to red-black graphs (see Section 3.4).

Generalized problem

For a red-black graph H we define a *k-L-labeling* as a function $\varphi : V(H) \to \{1,\dots,k\}$ (here we do not use 0 as a label for technical reasons) fulfilling the following conditions:

1. $|\varphi(v) - \varphi(w)| \geq 1$ for all $v, w \in V(G)$ such that $vw \in R(G)$
2. $|\varphi(v) - \varphi(w)| \geq 2$ for all $v, w \in V(G)$ such that $vw \in B(G)$.

Notice that a $(k+1)$-L-labeling of the R-closure of a graph G corresponds to a k-$L(2,1)$-labeling of G. More precisely, the following is true.

Remark 5.2. If ψ is a k-$L(2,1)$-labeling of a graph G and H is an R-closure of G, then a labeling φ defined by $\varphi(v) = \psi(v) + 1$ is a $(k+1)$-L-labeling of H and vice-versa.

For technical reasons we need to consider two more restrictions. Suppose that the given instance is a red-black graph G and sets P, Q. A labeling $\varphi\colon V(G) \to \{1, \dots, k\}$ is a *k-L_Q^P-labeling* of G if it is a k-L-labeling of G, such that $Q \cap \varphi^{-1}(1) = \emptyset$ and $P \cap \varphi^{-1}(k) = \emptyset$. A function φ is an L_Q^P-labeling of G if it is a k-L_Q^P-labeling of G for some k. Note that every L_Q^P-labeling of G is in fact a $L_{Q\cap V(G)}^{P\cap V(G)}$-labeling of G. However, we will not restrict the definition to sets $P, Q \subseteq V(G)$, as it makes the description of the algorithm simpler.

Let $\lambda_Q^P(G)$ denote the smallest possible $k \geq 0$, for which a k-L_Q^P-labeling of G exists. In particular, for a red-black graph with no vertices we have $\lambda_Q^P((\emptyset, P, Q)) \overset{def.}{=} 0$ for any sets P, Q.

The considered generalized problem asks to compute $\lambda_Q^P(G)$. Any k-L_Q^P-labeling of G with $k = \lambda_Q^P(G)$ is called *optimal*. We observe that even if φ is an optimal L_Q^P-labeling of G, then any of the sets $\varphi^{-1}(1)$ and $\varphi^{-1}(\lambda_Q^P(G))$ may be empty. In the extremal case, if $P = Q = V(G)$, then $\varphi^{-1}(1) = \varphi^{-1}(k) = \emptyset$ for all k and feasible k-L_Q^P-labelings φ of G.

Notice that if H is an R-closure of a graph G then $\lambda_\emptyset^\emptyset(H) = \lambda(G) + 1$, by Remark 5.2. In this way we shall use our algorithm to compute the $L(2,1)$-span of a given input graph.

Algorithm

Before we start, we shall need just two more definitions. A triple of sets (X, Y, Z) is a *balanced partition* of a red-black graph G if

1. the sets X, Y, Z form a partition of $V(G)$,
2. the set Y is independent,
3. all sets X, Y, Z are non-empty,
4. $|X| \leq \frac{|V(G)|}{2}$ and $|Z| \leq \frac{|V(G)|}{2}$.

A triple of sets (X, Y, Z) is a *correct partition* of G if it is a balanced partition of G and Y is an R-maximal independent set.

Observe that there is a close relationship between correct partitions and R-maximal proper pairs, introduced in Section 3.4. Namely, if (X, Y, Z) is a correct partition of G, then (Y, X) is an R-maximal proper pair. Hence we obtain the following Remark.

Remark 5.3. The number of R-maximal proper pairs in G is an upper bound for the number of correct partitions of G.

The main idea of the algorithm is based on two key observations, described in Lemmas 5.4 and 5.5.

Lemma 5.4. *Let φ be a k-L_Q^P-labeling of G. Let $h = \min\{j \in \{1, \dots, k\} \colon |\bigcup_{i=1}^{j} \varphi^{-1}(i)| \geq |V(G)|/2\}$. Define $X := \bigcup_{i=1}^{h-1} \varphi^{-1}(i)$, $Y := \varphi^{-1}(h)$ and $Z := \bigcup_{i=h+1}^{k} \varphi^{-1}(i)$. One of the following cases occurs:*

1. *$X = \emptyset$ and $|Y| \geq |V(G)|/2$,*
2. *$Z = \emptyset$ and $|Y| \geq |V(G)|/2$,*
3. *the triple (X, Y, Z) is a balanced partition of G.*

Proof. Suppose that neither the case 1 nor the case 2 occurs. Notice that the sets X, Y and Z clearly form a partition of $V(G)$ and Y is independent. Clearly $Y \neq \emptyset$, because otherwise we would choose $h - 1$ instead of h. Also by the definition of h, we have $|X| \leq |V(G)|/2$. Since the cases 1 and 2 did not occur, $X \neq \emptyset$ and $Z \neq \emptyset$. Moreover, the fact that $|X \cup Y| \geq |V(G)|/2$ implies that $|Z| \leq |V(G)|/2$. $\square$

Let φ be a k-L_Q^P-labeling of G (for some $k < L_Q^P(G)$) and let $h \in \{2, \dots, k-1\}$. We say that a k-L_Q^P-labeling φ' of G is an *h-maximization of φ* if the following conditions are fulfilled:

1. the set $\varphi'^{-1}(h)$ is R-maximal,
2. $\varphi^{-1}(h) \subseteq \varphi'^{-1}(h)$,
3. $\varphi'^{-1}(i) \subseteq \varphi^{-1}(i)$ for all $i \in \{1, \dots, k\} \setminus \{h\}$.

Lemma 5.5. *Let φ be a k-L_Q^P-labeling of G (for some $k < L_Q^P(G)$). For every $h \in \{2, \dots, k-1\}$, there exists an h-maximization of φ.*

Proof. If $\varphi^{-1}(h)$ is R-maximal, we are done. Otherwise, there exists at least one vertex v such that:

1. $\varphi(v) \neq h$,
2. $N_R(v) = N(v)$,
3. $\varphi(w) \neq h$ for all $w \in N(v)$.

Note that we can change the label of v to h obtaining another k-L_Q^P-labeling of G. Applying such a relabeling as many times as possible, we finally obtain a labeling φ', which is an h-maximization of φ. □

We observe that if a graph is disconnected, we can label each of its connected components separately. Hence for all graphs G and sets P, Q we have the following equality:

$$\lambda_Q^P(G) = \max\{\lambda_Q^P(C)\colon C \text{ is a connected component of } G\}. \tag{5.1}$$

Therefore we may assume that the input graph G is connected.

The algorithm partitions the vertex set $V(G)$ into all possible triples of sets X, Y, Z, which form a correct partition of G. The graphs $G[X]$ and $G[Z]$ are then labeled recursively. Note that $G[X]$ and $G[Z]$ may not be connected, so we can assume the connectivity only in the first step. Due to restrictions related to the sets P and Q, the cases when $X = \emptyset$ or $Z = \emptyset$ have to be considered separately.

The labeling of the whole graph G is constructed from the labelings found in the recursive calls. The sets of labels used on the sets X and Z are separated from each other by the label used for the R-maximal independent set Y. This allows to solve the subproblems for $G[X]$ and $G[Z]$ independently from each other. Iterating over all such partitions of $V(G)$, the algorithm computes the minimum k admitting the existence of a k-L_Q^P-labeling of G. By definition, such k is equal to $\lambda_Q^P(G)$. The pseudo-code of our procedure is given by Algorithm 3.

Remark 5.6. Observe that the algorithm can be easily adapted to construct an optimal labeling of G (not only to find $\lambda_Q^P(G)$). Basically, after obtaining optimal labelings of $C[X]$ and $C[Z]$ (recursive calls in lines 13 and 14), we construct the optimal labeling of C by shifting the labels used on Z by k_X+1.

Algorithm 3: Find-Lambda-Poly(G, P, Q)

```
Input: Red-black graph $G$, Sets $P, Q$
if $V(G) = \emptyset$ then return 0
if $\lambda_Q^P(G) \leq 3$ then
    return $\lambda_Q^P(G)$
foreach connected component $C$ of $G$ do
    $k[C] \leftarrow \infty$
    foreach independent set $Y \subseteq V(C)$ such that $|Y| \geq |V(C)|/2$ do
        $k_1 \leftarrow$ Find-Lambda-Poly($C - Y, N_B(Y), Q$)
        $k_2 \leftarrow$ Find-Lambda-Poly($C - Y, P, N_B(Y)$)
        if $Y \cap P \neq \emptyset$ then $k_1 \leftarrow k_1 + 1$
        if $Y \cap Q \neq \emptyset$ then $k_2 \leftarrow k_2 + 1$
        $k[C] \leftarrow \min(k[C], k_1 + 1, k_2 + 1)$
    foreach correct partition $(X, Y, Z)$ of $C$ do
        $k_X \leftarrow$ Find-Lambda-Poly($C[X], N_B(Y), Q$)
        $k_Z \leftarrow$ Find-Lambda-Poly($C[Z], P, N_B(Y)$)
        $k[C] \leftarrow \min(k[C], k_X + 1 + k_Z)$
return $\max\{k[C] \colon C$ is a connected component of $G\}$
```

Lemma 5.7. *For a red-black graph $G = (V, \emptyset, \emptyset)$ and any sets P, Q, we have that $\lambda_Q^P(G) \leq 3$.*

Proof. The labeling $\varphi\colon V \to \{1, 2, 3\}$ such that $\varphi(v) = 2$ for every $v \in V$ is a 3-L_Q^P-labeling of G. □

Lemma 5.8. *For any red-black graph G and sets P, Q, the algorithm call **Find-Lambda-Poly(**G, P, Q**)** returns $\lambda_Q^P(G)$.*

Proof. The proof proceeds by the induction on $|V(G)|$. If $V(G) = \emptyset$, the correct result is given in line 1 (by the definition of $\lambda_Q^P(\emptyset, P, Q)$). If $\lambda_Q^P(G) \leq 3$, the result is found in line 3, so from now on we assume that $\lambda_Q^P(G) > 3$. Notice that if $|V(G)| \leq 1$, then $\lambda_Q^P(G) \leq 3$ by Lemma 5.7.

Let $n > 1$ and assume that the statement is true for all graphs G' and all sets P', Q', such that $|V(G')| < n$. Let G be a red-black graph on n vertices and let P, Q be sets. Let k be the value returned by the algorithm call **Find-Lambda-Poly(**G, P, Q**)**. By (5.1), to show that $k = \lambda_Q^P(G)$ it is enough to show that that $k[C] = \lambda_Q^P(C)$ for every connected component C (for $k[C]$ defined as in the algorithm).

Let C be a connected component of G. First we will show that $k[C] \geq \lambda_Q^P(C)$, i.e. there exists a $k[C]$-L_Q^P-labeling of C. Assume that $k[C]$ was set in the line 11. Consider the independent set Y and the iteration of the loop in lines 6–11 for which $k[C]$ was set. Let $k_1' = \textbf{Find-Lambda-Poly}(C - Y, N_B(Y), Q)$ and $k_2' = \textbf{Find-Lambda-Poly}(C - Y, P, N_B(Y))$. By the induction hypothesis there exist a k_1'-$L_Q^{N_B(Y)}$-labeling φ' of $C - Y$ and a k_2'-$L_{N_B(Y)}^P$-labeling φ'' of $C - Y$. Notice that $k[C] \in \{k_1'+1, k_1'+2, k_2'+1, k_2'+2\}$ and at least one of the following cases occurs.

Case 1: $k[C] = k_1' + 1$ and $Y \cap P = \emptyset$.

In this case we can extend φ' in the following way

$$\varphi(v) = \begin{cases} \varphi'(v) & \text{if } v \in V(C) \setminus Y \\ k_1' + 1 & \text{if } v \in Y \end{cases}$$

obtaining a $k[C]$-L_Q^P-labeling φ of C.

Case 2: $k[C] = k_1' + 2$ and $Y \cap P \neq \emptyset$.

In this case we can extend φ' in the following way

$$\varphi(v) = \begin{cases} \varphi'(v) & \text{if } v \in V(C) \setminus Y \\ k_1' + 1 & \text{if } v \in Y \end{cases}$$

obtaining a $k[C]$-L_Q^P-labeling φ of C (note that due to restriction on P the label $k_1' + 2$ is counted as used, despite the fact that no vertex has this label).

Case 3: $k[C] = k_2' + 1$ and $Y \cap Q = \emptyset$.

In this case we can extend φ'' in the following way

$$\varphi(v) = \begin{cases} 1 & \text{if } v \in Y \\ \varphi''(v) + 1 & \text{if } v \in V(C) \setminus Y \end{cases}$$

obtaining a $k[C]$-L_Q^P-labeling φ of C.

Case 4: $k[C] = k_2' + 2$ and $Y \cap Q \neq \emptyset$.

In this case we can extend φ'' in the following way

$$\varphi(v) = \begin{cases} 2 & \text{if } v \in Y \\ \varphi''(v) + 2 & \text{if } v \in V(C) \setminus Y \end{cases}$$

obtaining a $k[C]$-L_Q^P-labeling φ of C.

Now assume that that $k[C]$ was set in line 15. Consider the correct partition (X, Y, Z) and the iteration of the loop in lines 12–15 for which $k[C]$ was set. Let k_X and k_Z be defined as in lines 13 and 14 for this iteration. Hence $k[C] = k_X + 1 + k_Z$. By the induction hypothesis there exists k_X-$L_Q^{N_B(Y)}$-labeling φ_X of $C[X]$ and k_Z-$L^P_{N_B(Y)}$-labeling φ_Z of $C[Z]$. We can define a $k[C]$-L_Q^P-labeling of C in the following way:

$$\varphi(v) = \begin{cases} \varphi_X(v) & \text{if } v \in X, \\ k_X + 1 & \text{if } v \in Y, \\ k_X + 1 + \varphi_Z(v) & \text{if } v \in Z. \end{cases}$$

Notice that sets X, Y, Z are non-empty since (X, Y, Z) is a correct partition of C. Hence $k_X, k_Z \geq 1$ and $\varphi^{-1}(1) \cap Q = \varphi^{-1}(k_X + 1 + k_Z) \cap P = \emptyset$.

Now we will show that $k[C] \leq \lambda_Q^P(C)$. Let φ be an optimal L_Q^P-labeling of C. Recall that $\lambda_Q^P(C) > 3$. One of the following cases occurs.

Case 1: $|\varphi^{-1}(1)| \geq |V(C)|/2$.

Consider the iteration of the loop in lines 6–11 for $Y = \varphi^{-1}(1)$. By the induction hypothesis the algorithm **Find-Lambda-Poly** sets $k_2 = \lambda^P_{N_B(Y)}(C - Y)$. Notice that $\varphi^{-1}(1) \cap Q = \emptyset$ and $\lambda^P_{N_B(Y)}(C - Y) = \lambda_Q^P(C) - 1$. Hence the condition in line 10 is not fulfilled and k_2 is equal to $\lambda_Q^P(C) - 1$. By the condition in line 11 we have $k[C] \leq k_2 + 1 = \lambda_Q^P(C)$.

Case 2: $\varphi^{-1}(1) = \emptyset$ and $|\varphi^{-1}(2)| \geq |V(C)|/2$.

Consider the iteration of the loop in lines 6–11 for $Y = \varphi^{-1}(2)$. By the induction hypothesis the algorithm **Find-Lambda-Poly** sets $k_2 = \lambda^P_{N_B(Y)}(C - Y)$. Notice that $\varphi^{-1}(2) \cap Q \neq \emptyset$ since otherwise we could decrease the label of every vertex by one, obtaining $(\lambda_Q^P(C) - 1)$-L_Q^P-labeling of C. Hence $\lambda^P_{N_B(Y)}(C - Y) = \lambda_Q^P(C) - 2$ and the algorithm **Find-Lambda-Poly** in line 10 sets k_2 to $\lambda^P_Q(C) - 2 + 1$. By the condition in line 11 we have $k[C] \le k_2 + 1 = \lambda^P_Q(C) - 2 + 1 + 1 = \lambda^P_Q(C)$.

Case 3: $|\varphi^{-1}(\lambda^P_Q(C))| \ge |V(C)|/2$.
This case is analogous to the Case 1.

Case 4: $\varphi^{-1}(\lambda^P_Q(C)) = \emptyset$ and $|\varphi^{-1}(\lambda^P_Q(C) - 1)| \ge |V(C)|/2$.
This case is analogous to the Case 2.

Notice that neither the case $\varphi^{-1}(1) = \varphi^{-1}(2) = \emptyset$ nor $\varphi^{-1}(\lambda^P_Q(C)) = \varphi^{-1}(\lambda^P_Q(C) - 1) = \emptyset$ may occur, since the labeling φ is optimal. Hence the only case remaining is:

Case 5: Both conditions:

- $0 < |\varphi^{-1}(1)| < |V(C)|/2$ or $0 < |\varphi^{-1}(1) \cup \varphi^{-1}(2)| < |V(C)|/2$ and
- $0 < |\varphi^{-1}(\lambda^P_Q(C))| < |V(C)|/2$ or $0 < |\varphi^{-1}(\lambda^P_Q(C)) \cup \varphi^{-1}(\lambda^P_Q(C) - 1)| < |V(C)|/2$.

are fulfilled. Notice that we can assume that the conditions above are satisfied for every optimal L^P_Q-labeling φ of G, since otherwise we are done by Cases 1–4.

Let $h = \min\{j \in \{1, \dots, \lambda^P_Q(C)\} \colon |\bigcup_{i=1}^{j} \varphi^{-1}(i)| \ge |V(C)|/2\}$. Note that $1 < h < \lambda^P_Q(C)$, because of the conditions of the Case 5. By Lemma 5.4, the triple $(\bigcup_{i=1}^{h-1} \varphi^{-1}(i), \varphi^{-1}(h), \bigcup_{i=h+1}^{\lambda^P_Q(C)} \varphi^{-1}(h))$ is a balanced partition of G.

Let φ' be a h-maximization of φ (its existence is guaranteed by Lemma 5.5). Let $X = \bigcup_{i=1}^{h-1} \varphi'^{-1}(i)$, $Y = \varphi'^{-1}(h)$ and $Z = \bigcup_{i=h+1}^{\lambda^P_Q(C)} \varphi'^{-1}(h)$. Notice that by the definition of h-maximization, we have the following:

- $\emptyset \ne \varphi^{-1}(h) \subseteq Y$ implies that $Y \ne \emptyset$,
- $X \subseteq \bigcup_{i=1}^{h-1} \varphi^{-1}(i)$ implies that $|X| \le |V(C)|/2$,
- $Z \subseteq \bigcup_{i=h+1}^{\lambda^P_Q(C)} \varphi^{-1}(i)$ implies that $|Z| \le |V(C)|/2$.

Moreover $X \ne \emptyset$ and $Z \ne \emptyset$, because otherwise we are done by Case 2 or Case 4 for φ'. Hence (X, Y, Z) is a correct partition of C and it is considered in some iteration of the loop in lines 12–15. In the iteration for (X, Y, Z) the algorithm sets $k_X = \lambda^{N_B(Y)}_Q(C[X]) = h - 1$ in line 13 and

$k_Z = \lambda^P_{N_B(Y)}(C[Z]) = \lambda^P_Q(C) - h$ in line 14 (by the induction hypothesis). Hence $k[C] \leq k_X + 1 + k_Z = h - 1 + 1 + \lambda^P_Q(C) - h = \lambda^P_Q(C)$, which completes the proof. □

A direct estimation of the computation complexity of our algorithm gives a running-time $\mathcal{O}((9+\epsilon)^n)$. However, by using some results from Section 3.4, we can improve this running-time upper bound, as claimed in the next Lemma.

Lemma 5.9. *The algorithm **Find-Lambda-Poly** computes $\lambda^P_Q(G)$ of a red-black graph in time $\mathcal{O}((8+\epsilon)^n)$ and polynomial space, where n is the number of vertices in G and ϵ is an arbitrarily small positive constant.*

Proof. Verifying if a given set Y is independent can be performed in polynomial time. We can check if a given function $\varphi\colon V(G) \to \mathbb{N}$ is an L^P_Q-labeling of G in polynomial time as well. The algorithm **Find-Lambda-Poly** first checks in constant time if $V(G) = \emptyset$. Then it exhaustively checks if there exists a k-L^P_Q-labeling of G for $k \in \{1,2,3\}$. There are 3^n functions $\varphi\colon V(G) \to \{1,2,3\}$, so this step is performed in time $n^{\mathcal{O}(1)} \cdot 3^n$.

Then for every connected component C of G the algorithm checks all independent sets of size at least $|V(C)|/2$ (there are no more than 2^n such sets) and all correct partitions of C (by Theorem 3.27 and Remark 5.3 there are at most $\sqrt{8}^n \cdot n^{\mathcal{O}(1)}$ considered partitions and they can be enumerated within this time bound, using polynomial space). For each connected component of G, the algorithm is called recursively for at most two subgraphs of order at most $n/2$ each. Hence we obtain the following inequality for the complexity:

$$T(n) \leq (2^n + \sqrt{8}^n) n^{\mathcal{O}(1)} T(n/2) \tag{5.2}$$

This recursive inequality implies $T(n) = \mathcal{O}(8^n n^{\mathcal{O}(1)\log n}) = \mathcal{O}(8^n 2^{\mathcal{O}(1)\log^2 n})$, which is bounded by $\mathcal{O}((8+\epsilon)^n)$, for all $\epsilon > 0$. The space complexity of the algorithm is clearly polynomial. □

Proof of Theorem 5.1. Notice that if we are looking for a $L(2,1)$-labeling of G, we can assume that G is connected (otherwise we would label each of its components separately) and the initial graph given to the algorithm is an R-closure of G (an R-closure of a graph can be found in polynomial time). Recall that by Observation 3.29, the number of R-maximal proper pairs in a red-black graph on n vertices, which is an R-closure of some connected graph, is bounded by $\mathcal{O}(2.6488^n)$.

Hence the complexity of the algorithm is bounded by

$$T'(n) \leq (2^n + 2.6488^n) n^{\mathcal{O}(1)} T(n/2), \tag{5.3}$$

where T is given by the inequality (5.2). By (5.3) we get $T'(n) = \mathcal{O}(7.4920^n)$, which completes the proof. □

Remark 5.10. Notice that the algorithm **Find-Lambda-Poly** can be easily generalized for the ℓ-bounded channel assignment problem (for an integer $\ell \geq 2$). To adapt the algorithm, we have to consider the partition of the vertex set into $\ell + 1$ sets $X, Y_1, \ldots, Y_{\ell-1}, Z$, such that $|Y| \leq n/2$, $|Z| \leq n/2$ and $Y_1, \ldots, Y_{\ell-1}$ are independent. Then we run the algorithm recursively on $G[X]$ and $G[Y]$. This algorithm has time complexity $\mathcal{O}^*((\ell+1)^{2n})$ and polynomial space complexity. The complexity bound can be slightly improved by noticing that each of the sets $Y_1, Y_2, \ldots, Y_{\ell-1}$ is independent and considering only such partitions. However, this requires much technical effort and the improvement is very little (and becomes smaller as ℓ grows).

5.2 k-$L(2,1)$-labeling problem with k fixed

In this section we shall adapt the algorithm **Find-Lambda-Poly** to solve the k-$L(2,1)$-labeling problem if k is a fixed integer. Again, we shall start with solving k-L_Q^P-labeling problem for red-black graphs.

We observe that k-L_Q^P-labeling problem is a special case of the so-called $(d,2)$-*CSP* (constrained satisfaction problem). An instance of $(d,2)$-CSP consists of:

1. a set of vertices V,
2. sets of possible labels C_v for each vertex v, such that $|C_v| \leq d$,
3. a set $\mathcal{C}$ of constraints, each of the form $\{(v,c_v),(w,c_w)\}$, where $v,w \in V$ and $c_v \in C_v$ and $c_w \in C_w$.

The problem asks if there exists a labeling φ of vertices, such that

1. $\varphi(v) \in C_v$ for all $v \in V$,
2. $\{(v,\varphi(v)),(w,\varphi(w))\} \notin \mathcal{C}\}$ for all $v,w \in V$.

Theorem 5.11 summarizes complexity bounds for the exact algorithms for the $(d,2)$-CSP.

Theorem 5.11 (Beigel, Eppstein [8], Angelsmark [7])**.** *The $(d,2)$-CSP problem can be solved in polynomial space and time*

1. *$\mathcal{O}(1.3645^n)$ for $d = 3$,*
2. *$\mathcal{O}((0.4518 \cdot d + \epsilon)^n)$ (where ϵ is any positive constant) for all $d \geq 4$.*

Now let us explain how to encode an instance of k-L_Q^P-labeling problem as an instance of $(d,2)$-CSP. For each vertex we consider the following list of possible labels:

$$C_v = \begin{cases} \{2,3,\ldots,k-1\} & \text{if } v \in P \cap Q \\ \{2,3,\ldots,k\} & \text{if } v \in Q \setminus P \\ \{1,2,\ldots,k-1\} & \text{if } v \in P \setminus Q \\ \{1,2,\ldots,k\} & \text{otherwise.} \end{cases}$$

The set of constraints is defined as follows:

$$\begin{aligned}\mathcal{C} = &\{\{(v,c),(w,c)\} \text{ for } vw \in E(G), c \in C_v \cap C_w\} \cup \\ &\{\{(v,c),(w,c+1)\} \text{ for } vw \in B(G), c \in C_v, c+1 \in C_w\} \cup \\ &\{\{(v,c),(w,c-1)\} \text{ for } vw \in B(G), c \in C_v, c-1 \in C_w\}.\end{aligned}$$

For $k \leq 8$ we shall use this approach to solve the k-L_Q^P-labeling problem in the same time as in Theorem 5.11. For the larger number of labels, we proceed in a similar way as with the algorithm **Find-Lambda-Poly**, but the way of partitioning the vertex set into three subsets is slightly different. A similar reasoning as in the proof of Lemma 5.5 shows that we can assume that the set Y labeled with the label $\lceil k/2 \rceil$ is R-maximal. We partition the set of remaining vertices into two sets X and Z, to be labeled with labels $1, 2, .., \lceil k/2 \rceil - 1$ and $\lceil k/2 \rceil + 1, .., k$, respectively. The labeling of X (Z, respectively) is an $L_Q^{N(Y)}$-labeling ($L_{N(Y)}^P$-labeling, respectively) of the graph $G[X]$ ($G[Y]$, respectively). Algorithm 4 shows the pseudocode of this procedure.

```
Algorithm 4: Determine-Lambda($G, P, Q, k$)
  Input: Red-black graph $G$, Sets $P, Q$, Integer $k$
1 if $k \leq 8$ then
2     if $\lambda_Q^P(G) \leq k$ then return YES
3     else return NO
4 foreach $R$-maximal independent set $Y$ do
5     foreach $(X, Z)$ being a partition of $V(G) \setminus Y$ do
6         if Determine-Lambda($G[X], N(Y), Q, \lceil k/2 \rceil - 1$) =YES and
7         Determine-Lambda($G[Z], P, N(Y), k - \lceil k/2 \rceil$) =YES then
8             return YES
9 return NO
```

Let $T'_k(n)$ denote the worst-case complexity of the algorithm **Determine-Lambda**, used to solve the k-L_Q^P-labeling problem on a red-black graph on n vertices. By $\overline{r}_\ell(n)$ let us denote the time needed to enumerate all ℓ-element R-maximal independent sets in a red-black graph with n vertices. We obtain the following formula for the complexity for $k > 8$:

$$T'_k(n) \leq \sum_{\ell=0}^{n} \overline{r}_\ell(n) \sum_{m=0}^{n-\ell} \binom{n-\ell}{m} \left(T'_{\lceil k/2 \rceil -1}(m) + T'_{\lfloor k/2 \rfloor}(n-\ell-m)\right). \tag{5.4}$$

Lemma 5.12. *Let $k > 8$. If $T'_{\lfloor k/2 \rfloor}(n) = \mathcal{O}^*(\alpha^n)$ for some $\alpha > \frac{11}{7}$, then $T'_k(n) = \mathcal{O}^*\left(\left(\frac{4}{3}(1+\alpha)^n\right)\right)$.*

Proof. Let $\beta := \alpha + 1$. Let G be a red-black graph with n vertices, being the worst-case instance for the algorithm **Determine-Lambda**. Let $n' := |V(G_B)|$. From the formula (5.4) we obtain the following.

$$\begin{aligned}
T'_k(n) \leq & \sum_{\ell=0}^{n} \overline{r}_\ell(n) \sum_{m=0}^{n-\ell} \binom{n-\ell}{m} \left(T'_{\lceil k/2 \rceil -1}(m) + T'_{\lfloor k/2 \rfloor}(n-\ell-m)\right) \\
\leq & 2 \sum_{\ell=0}^{n} \overline{r}_\ell(n) \sum_{m=0}^{n-\ell} \binom{n-\ell}{m} T'_{\lfloor k/2 \rfloor}(n-\ell-m) \\
\leq & n^{\mathcal{O}(1)} \sum_{\ell=0}^{n} \overline{r}_\ell(n) \sum_{m=0}^{n-\ell} \binom{n-\ell}{m} \alpha^{n-\ell-m} \\
= & n^{\mathcal{O}(1)} \sum_{\ell=0}^{n} \overline{r}_\ell(n) (1+\alpha)^{n-\ell} = n^{\mathcal{O}(1)} \sum_{\ell=0}^{n} \beta^{n-\ell} \cdot \overline{r}_\ell(n)
\end{aligned}$$

Now we apply the bound for $\overline{r}_\ell(n)$ from Lemma 3.26.

$$\begin{aligned}
T'_k(n) \leq & n^{\mathcal{O}(1)} \sum_{\ell=0}^{n} \beta^{n-\ell} \cdot \left(n^{\mathcal{O}(1)} \cdot (4/3)^{n-n'} \left(\frac{81}{64}\right)^{\ell} \sum_{\ell'=0}^{\ell} \binom{n'/2}{\ell'} \left(\frac{128}{81}\right)^{\ell'} \right) \\
= & n^{\mathcal{O}(1)} \left(\frac{4}{3}\right)^{n-n'} \beta^n \sum_{\ell=0}^{n} \sum_{\ell'=0}^{\ell} \binom{n'/2}{\ell'} 2^{\ell'} \left(\frac{81}{64}\right)^{\ell-\ell'} \beta^{-\ell} \\
= & n^{\mathcal{O}(1)} \left(\frac{4}{3}\right)^{n-n'} \beta^n \sum_{\ell'=0}^{n} \sum_{\ell=\ell'}^{n} \binom{n'/2}{\ell'} 2^{\ell'} \left(\frac{81}{64}\right)^{\ell-\ell'} \beta^{-\ell} \\
= & n^{\mathcal{O}(1)} \left(\frac{4}{3}\right)^{n-n'} \beta^n \sum_{\ell'=0}^{n} \binom{n'/2}{\ell'} \left(\frac{128}{81}\right)^{\ell'} \sum_{\ell=\ell'}^{n} \left(\frac{81}{64\beta}\right)^{\ell}
\end{aligned}$$

Since we assumed that $\alpha > \frac{11}{7}$ and thus $\beta > \frac{18}{7}$, the value of $\left(\frac{81}{64\beta}\right)^{\ell}$ is smaller than 1. Hence,

$$\begin{aligned}
T'_k(n) \leq & n^{\mathcal{O}(1)} \left(\frac{4}{3}\right)^{n-n'} \beta^n \sum_{\ell'=0}^{n} \binom{n'/2}{\ell'} \left(\frac{128}{81}\right)^{\ell'} (n-\ell'+1) \left(\frac{81}{64\beta}\right)^{\ell'} \\
\leq & n^{\mathcal{O}(1)} \left(\frac{4}{3}\right)^{n-n'} \beta^n \sum_{\ell'=0}^{n} \binom{n'/2}{\ell'} \left(\frac{2}{\beta}\right)^{\ell'} = n^{\mathcal{O}(1)} \left(\frac{4}{3}\right)^{n-n'} \beta^n \sum_{\ell'=0}^{n'/2} \binom{n'/2}{\ell'} \left(\frac{2}{\beta}\right)^{\ell'} \\
= & n^{\mathcal{O}(1)} \left(\frac{4}{3}\right)^{n-n'} \beta^n \left(1+\frac{2}{\beta}\right)^{n'/2} = n^{\mathcal{O}(1)} \left(\frac{4\beta}{3}\right)^{n} \left(\frac{3}{4}\cdot\sqrt{\frac{\beta+2}{\beta}}\right)^{n'}.
\end{aligned}$$

Since $\beta > \frac{18}{7}$, the value of $\frac{3}{4} \cdot \sqrt{\frac{\beta+2}{\beta}}$ is smaller than 1. Therefore we obtain the following bound:

$$T'_k(n) \leq n^{\mathcal{O}(1)} \left(\frac{4\beta}{3}\right)^{n} = n^{\mathcal{O}(1)} \left(\frac{4(1+\alpha)}{3}\right)^{n}.$$

□

Table 5.1 presents the bounds for $T'_k(n)$ for some values of k. The bounds for $k \leq 8$ follow from Theorem 5.11, while the bounds for $k \geq 9$ follow from Lemma 5.12.

If we want to solve a k-$L(2,1)$-labeling problem, we can again assume that the initial graph is an R-closure of some connected graph. Therefore, by Observation 3.23, in the first step we choose the set Y, which is a 2-packing. Thus we can use the bound for the time needed to enumerate all ℓ-element 2-packings in a connected graph with n vertices (let us denote it by $\overline{u}_\ell(n)$), which is given by Theorem 3.6. This gives us the following bound on $T_k(n)$, being the complexity of the modified algorithm for $k > 8$ (recall that in the k-$L(2,1)$-labeling we start labeling with 0, so the number of labels available is $k+1$).

$$T_k(n) \leq \sum_{\ell=0}^{n} \overline{u}_\ell(n) \sum_{m=0}^{n-\ell} \binom{n-\ell}{m} \left(T'_{\lceil k/2\rceil -1}(m) + T'_{\lfloor k/2\rfloor}(n-\ell-m)\right)$$

k	complexity
3	1.3645
4	1.8072
5	2.2591
6	2.7109
7	3.1627
8	3.6147
9	3.7430
10	4.3455
..	
15	5.5503
16	6.1530

Table 5.1: Bases of the exponent in the bound for $T'_k(n)$

Let $T'_{\lfloor k/2 \rfloor}(n) = \mathcal{O}^*(\alpha^n)$ for some α (see Table 5.1). From the formula above, we obtain the following by a similar reasoning as in the proof of Lemma 5.12.

$$\begin{aligned}
T_k(n) \leq & \sum_{\ell=0}^{n} \overline{u}_\ell(n) \sum_{m=0}^{n-\ell} \binom{n-\ell}{m} \left(T'_{\lceil k/2 \rceil -1}(m) + T'_{\lfloor k/2 \rfloor}(n-\ell-m)\right) \\
\leq & n^{\mathcal{O}(1)} \sum_{\ell=0}^{n} \overline{u}_\ell(n) \sum_{m=0}^{n-\ell} \binom{n-\ell}{m} \alpha^m \\
= & n^{\mathcal{O}(1)} \sum_{\ell=0}^{n} \overline{u}_\ell(n)(1+\alpha)^{n-\ell} = n^{\mathcal{O}(1)} \sum_{\ell=0}^{n} \binom{n-\ell+1}{\ell} (1+\alpha)^{n-\ell}.
\end{aligned}$$

Now, for every fixed k we can estimate the bound for $T_k(n)$ using the above formula. Table 5.2 shows obtained bounds on $T_k(n)$ for small k. Observe that for $k \leq 31$ there are better than the bound for the complexity of the algorithm **Find-Lambda-Poly** from Theorem 5.1.

The values in the second column of Table 5.2 are obtained by a more careful analysis of the algorithm for the k-$L(2,1)$-labeling problem by Havet *et al.* [45].

k	Havet *et al.* [45]	new bound
4	1.3006	
5	2.4495	
6	3.4642	3.1219
7	4.4722	3.5894
8	5.4773	3.5894
9	6.4808	4.0616
10	7.4833	4.0616
11	8.4853	4.5301
	...	
31	28.4957	7.4317
32	24.4959	8.0424

Table 5.2: Bases of the exponent in the bound for $T_k(n)$

Chapter 6

Open problems

In this section we state some open problems that seem worth investigating.

6.1 Problems concerning Chapter 3

In Section 3.1 we showed an upper bound for the number $u_k(n)$ of k-element 2-packings in a connected graph. However, we find this bound unsatisfactory and do not use it when estimating the number of proper pairs (in Section 3.3). Thus we state the first open problem.

Problem 6.1. Find an upper bound for $u_k(n)$, better than $\binom{n-k+1}{k}$.

By analogy to the case of independent sets, it seems natural to consider the maximum number of *maximal* 2-packings in a graph G with n vertices. Note that if G does not have to be connected, the correct bound is the same as for maximal independent set, i.e. $3^{n/3}$ (see [75, 76, 87]). The problem gets more interesting is we restrict ourselves to connected graphs.

Problem 6.2. What is the maximum number of maximal 2-packings in a connected graph with n vertices?

We think this problem is interesting even for trees. It is worth mentioning that a tight bound for the maximum number of maximal independent sets in a tree has been showed by Sagan [79].

Problem 6.3. What is the maximum number of maximal 2-packings in a tree with n vertices?

6.2 Problems concerning Chapter 4

Very recently, Kowalik and Socała [62] published an algorithm for the ℓ-bounded channel assignment problem working in time $\mathcal{O}(2^n(\ell+2)^{n/2} \cdot n^2)$. This is the first algorithm whose complexity is smaller than $(c \cdot \ell)^n$ for any constant c. However, the generalization of this result to the generalized list T-coloring problem does not seem easy.

Problem 6.4. Design an exact exponential algorithm for the τ-bounded generalized list T-coloring problem with complexity smaller than $(c \cdot \ell)^n$ for any constant c.

There are also many open questions concerning exact algorithms for the (locally constrained) graph homomorphism problems. As we mentioned before, there are no non-trivial exact algorithms deciding the existence of a locally surjective or a locally bijective graph homomorphism either.

Problem 6.5. Design a non-trivial exact exponential algorithm for the locally surjective and the locally bijective graph homomorphism problem.

Kowalik and Socała [62] and Socała [81] proved that, assuming the ETH (the Exponential Time Hypothesis, see Section 1.3), there is no algorithm with complexity $\mathcal{O}^*(c^n)$ for any constant c, solving the channel assignment problem or the non-list generalized T-coloring problem. It would be interesting to show similar results for the list version of the problem and the graph homomorphism and the locally injective graph homomorphism problem as well.

Problem 6.6. Assuming the ETH, show that there is no exact algorithm for the τ-bounded generalized list T-coloring with time complexity bounded by $\mathcal{O}^*(c^n)$ for a constant c (or design such an algorithm).

Problem 6.7. Assuming the ETH, show that there is no exact algorithm for the (locally injective) graph homomorphism problem with time complexity bounded by $\mathcal{O}^*(c^n)$ for a constant c (or design such an algorithm).

6.3 Problems concerning Chapter 5

Recall that in Theorem 4.16 we have shown that the $L(2,1)$-labeling problem can be solved in time $\mathcal{O}(2.6488^n)$ (where n is the number of vertices of the input graph), using exponential space. If only a polynomial space is available, by Theorem 5.1, the problem can be solved in time $\mathcal{O}(7.4922^n)$. The gap between those bounds is very large. On the other hand, Björklud *et al.* [9] showed that the graph coloring problem can be solved in time $\mathcal{O}^*(2^n)$ using an exponential space or in time $\mathcal{O}(2.2461^n)$, using polynomial space. They use an algorithm based on the inclusion-exclusion principle. Similar approach has been used by Cygan and Kowalik [21] to design an exact algorithm for the channel assignment problem. When we apply this algorithm to the case of the $L(2,1)$-labeling, we obtain the time complexity $\mathcal{O}^*(3^n)$ and exponential space complexity. However, we can see no easy way to adapt this algorithm so that it uses only polynomial space.

It would be interesting to find an algorithm for the $L(2,1)$-labeling problem, whose time complexity is not much larger than $\mathcal{O}(2.6488^n)$, but using only polynomial space.

Problem 6.8. Design an exact algorithm for the $L(2,1)$-labeling problem using only polynomial space and working in time $\mathcal{O}(c^n)$ for a constant c close to 3.

Bibliography

[1] J. Abello, M. Fellows, and J. Stillwell. On the complexity and combinatorics of covering finite complexes. *Australian Journal of Combinatorics*, 4:103 – 112, 1991.

[2] J. Akiyama and M. Kano. *Factors and Factorizations of Graphs*, volume 2031 of *Lecture Notes in Mathematics*. Springer, 2011.

[3] N. Alon. personal communication.

[4] N. Alon. Independent sets in regular graphs and sum-free subsets of finite groups. *Israel Journal of Mathematics*, 73:247 – 256, 1991.

[5] N. Alon and A. Zaks. T - choosability in graphs. *Discrete Applied Mathematics*, 82:1 – 13, 1998.

[6] A. Amahashi and M. Kano. On factors with given components. *Discrete Mathematics*, 42:1 – 6, 1982.

[7] O. Angelsmark. Constructing Algorithms for Constraint Satisfaction and Related Problems. Methods and Applications. PhD Thesis, Linkp̈ing universiteit.

[8] R. Beigel and D. Eppstein. 3-coloring in time $O(1.3289^n)$. *Journal of Algorithms*, 54:168 – 204, 2005.

[9] A. Björklund, T. Husfeldt, and M. Koivisto. Set Partitioning via Inclusion-Exclusion. *SIAM Journal on Computing*, 39:546 – 563, 2009.

[10] K. Bodlaender, T. Kloks, R. Tan, and J. van Leeuwen. Approximations for lambda-Colorings of Graphs. *Computer Journal*, 47:193 – 204, 2004.

[11] A. Brandstädt, V. B. Le, and J. P. Spinrad. *Graph classes: a survey.* Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1999.

[12] Y. Bu, D. Cranston, M. Montassier, A. Raspaud, and W. Wang. Star coloring of sparse graphs. *Journal of Graph Theory*, 62:201 – 219, 2009.

[13] J. M. Byskov. Enumerating maximal independent sets with applications to graph colouring. *Operations Research Letters*, 32:547–556, 2004.

[14] T. Calamoneri. The $L(h, k)$-Labelling problem: An updated Survey and Annotated Bibliography. *The Computer Journal*, 54:1344 – 1371, 2011.

[15] G. Chang and D. Kuo. The $L(2, 1)$-labeling problem on graphs. *SIAM Journal on Discrete Mathematics*, 9:309 – 316, 1996.

[16] P. Chinn, J. Chvátalová, A. Dewdney, and N. Gibbs. The bandwidth problem for graphs and matrices – a survey. *Journal of Graph Theory*, 6:223 – 254, 1982.

[17] F. Chung, R. Graham, P. Frankl, and J. Shearer. Some intersection theorems for ordered sets and graphs. *Journal of Combinatorial Theory, Series A*, 43:23 – 37, 1986.

[18] J. Chvátalová, A. Dewdney, N. Gibbs, and R. Korfhage. The bandwidth problem for graphs, a collection of recent results. *Dept. of Comp. Science, University of Werstent Ontario*, Research Resport #24, 1975.

[19] B. N. Clark, C. J. Colbourn, and D. S. Johnson. Unit disk graphs. *Discrete Mathematics*, 86:165 – 177, 1990.

[20] J.-F. Couturier, P. Golovach, D. Kratsch, M. Liedloff, and A. Pyatkin. Colorings with few Colors: Counting, Enumeration and Combinatorial Bounds. *Theory of Computing Systems*, 52:645 – 667, 2012.

[21] M. Cygan and Ł. Kowalik. Channel assignment via fast zeta transform. *Information Processing Letters*, 111:727 – 730, 2011.

[22] R. Diestel. *Graph Theory.* Graduate Texts in Mathematics. Springer, 2005.

[23] N. Eggeman, F. Havet, and S. Noble. k-$L(2,1)$-Labelling for Planar Graphs is NP-Complete for $k \geq 4$. *Discrete Applied Mathematics*, 158:1777 – 1788, 2010.

[24] D. Eppstein. Small maximal independent sets and faster exact graph coloring. *Journal on Graph Algorithms and Applications*, 7:131 – 140, 2003.

[25] P. Erdős, A. Rubin, and H. Taylor. Choosability in graphs. *Proc. West Coast Conference on Combinatorics, Graph Theory and Computing, Arcata, Congressus Numerantium*, 26:125 – 157, 1979.

[26] J. Fiala, P. Golovach, and J. Kratochvíl. Parameterized complexity of coloring problems: Treewidth versus vertex cover. *Theoretical Computer Science*, 412:2513–2523, 2011.

[27] J. Fiala, T. Kloks, and J. Kratochvíl. Fixed - parameter complexity of λ-labelings. *Discrete Applied Mathematics*, 113:59 – 72, 2001.

[28] J. Fiala, D. Král', and R. Skrekovski. A Brooks-Type Theorem for the Generalized List T-Coloring. *SIAM Journal on Discrete Mathematics*, 19:588 – 609, 2005.

[29] J. Fiala and J. Kratochvíl. Complexity of partial covers of graphs. *Proc. of ISAAC 2001, LNCS*, 2223:537 – 549, 2001.

[30] J. Fiala and J. Kratochvíl. On the Computational Complexity of the $L(2,1)$-Labeling Problem for Regular Graphs. *Proc. of ICTCS 2005, LNCS*, 3701:228 – 236, 2005.

[31] J. Fiala and J. Kratochvíl. Locally Injective Graph Homomorphism: Lists Guarantee Dichotomy. *Proc. of WG 2006, LNCS*, 4271:15 – 26, 2006.

[32] J. Fiala and J. Kratochvíl. Locally constrained graph homomorphisms – structure, complexity, and applications. *Computer Science Review*, 2:97 – 111, 2008.

[33] J. Fiala, J. Kratochvíl, and A. Pór. On the computational complexity of partial covers of theta graphs. *Discrete Applied Mathematics*, 156:1143 – 1149, 2008.

[34] J. Fiala and D. Paulusma. A complete complexity classification of the role assignment problem. *Theoretical Computer Science*, 349:67 – 81, 2005.

[35] J. Fiala and R. Škrekovski. List distance labelings of graphs. *KAM Series*, 530, 2001.

[36] F. Fomin, F. Grandoni, and D. Kratsch. A measure and conquer approach for the analysis of exact algorithms. *Journal of the ACM*, 56:1 – 32, 2009.

[37] F. Fomin, P. Heggernes, and D. Kratsch. Exact Algorithms for Graph Homomorphisms. *Theory of Computing Systems*, 41:381 – 393, 2007.

[38] M. Garey, J. D.S., and S. L. Some simplified NP-complete graph problems. *Theoretical Computer Science*, 1:237 – 267, 1976.

[39] M. R. Garey and D. S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness.* W. H. Freeman & Co., New York, NY, USA, 1990.

[40] D. Gon**c**calves. On the $L(p, 1)$-labelling of graphs. *Discrete Mathematics*, 308:1405 – 1414, 2008.

[41] J. Griggs and R. K. Yeh. Labeling graphs with a condition at distance two. *SIAM Journal on Discrete Mathematics*, 5:586 – 595, 1992.

[42] W. Hale. Frequency assignment: theory and applications. *Proc. IEEE*, 68:1497 – 1514, 1980.

[43] T. Hasunuma, T. Ishii, H. Ono, and Y. Uno. A Linear Time Algorithm for $L(2, 1)$-Labeling of Trees. *Proc. of ESA 2009, LNCS*, 5757:35 – 46, 2009.

[44] F. Havet, M. Klazar, J. Kratochvíl, D. Kratsch, and M. Liedloff. Exact Algorithms for $L(p, q)$-labelings of graphs. manuscript.

[45] F. Havet, M. Klazar, J. Kratochvíl, D. Kratsch, and M. Liedloff. Exact algorithms for $L(2, 1)$-labelling. *Algorithmica*, 59:169 – 194, 2011.

[46] F. Havet, B. Reed, and J. S. Sereni. Griggs and Yeh's Conjecture and $L(p, 1)$-labelings. *SIAM Journal on Discrete Mathematics*, 26:145 – 168, 2012.

[47] P. Hell and D. G. Kirkpatrick. Packings by cliques and by finite families of graphs. *Discrete Mathematics*, 49:45 – 59, 1984.

[48] P. Hell and J. Nešetřil. On the complexity of H-coloring. *Journal of Combinatorial Theory, Series B*, 48:92 – 110, 1990.

[49] P. Hell and J. Nešetřil. *Graphs and Homomorphisms.* New Oxford University Press, 2004.

[50] R. Impagliazzo and R. Paturi. On the complexity of k-SAT. *Journal of Computer and System Sciences*, 62:367 – 375, 2001.

[51] R. Janczewski, A. Kosowski, and M. Małafiejski. The complexity of the $L(p, q)$-labeling problem for bipartite planar graphs of small degree. *Discrete Mathematics*, 309:3270 – 3279, 2009.

[52] T. R. Jensen and B. Toft. *Graph Coloring Problems.* John Wiley and Sons, 1995.

[53] K. Junosza-Szaniawski, J. Kratochvíl, M. Liedloff, P. Rossmanith, and P. Rzążewski. Fast exact algorithm for $L(2, 1)$-labeling of graphs. *Theoretical Computer Science*, 505:42 – 54, 2013.

[54] K. Junosza-Szaniawski, J. Kratochvíl, M. Liedloff, and P. Rzążewski. Determining the $L(2, 1)$-span in Polynomial Space. *Discrete Applied Mathematics*, 161:2052 – 2061, 2012.

[55] K. Junosza-Szaniawski and P. Rzążewski. Determining $L(2, 1)$-span in Polynomial Space. *arXiv:1104.4506v1 [cs.DM]*, 2011.

[56] K. Junosza-Szaniawski and P. Rzążewski. On Improved Exact Algorithms for $L(2, 1)$-Labeling of Graphs. *Proc. of IWOCA 2010, LNCS*, 6460:34 – 37, 2011.

[57] K. Junosza-Szaniawski and P. Rzążewski. On the complexity of exact algorithm for $L(2, 1)$-labeling of graphs. *Information Processing Letters*, 111:697 – 701, 2011.

[58] K. Junosza-Szaniawski and P. Rzążewski. On the number of 2-packings in a connected graph. *Discrete Mathematics*, 312:3444 – 3450, 2012.

[59] K. Junosza-Szaniawski and P. Rzążewski. An Exact Algorithm for the Generalized List T-Coloring Problem. *Discrete Mathematics and Theoretical Computer Science*, 16:77 – 94, 2014.

[60] J. Kahn. An entropy approach to the hard - core model on bipartite graphs. *Combinatorics, Probability and Computing*, 10:219 – 237, 2001.

[61] R. Karp. Reducibility Among Combinatorial Problems. *Complexity of Computer Computations*, page 85–103, 1972.

[62] Ł. Kowalik and A. Socała. Assigning channels via the meet-in-the-middle approach. *Proc. of SWAT 2014, LNCS*, 8503:282 – 293, 2014.

[63] D. Král'. An exact algorithm for the channel assignment problem. *Discrete Applied Mathematics*, 145:326 – 331, 2005.

[64] D. Král'. The channel assignment problem with variable weights. *SIAM Journal on Discrete Mathematics*, 20:690 – 704, 2006.

[65] J. Kratochvíl. Perfect codes in graphs. Proceedings of VII Hungarian Colloquium on Combinatorics.

[66] J. Kratochvíl. Regular codes in regular graphs are difficult. *Discrete Mathematics*, 133:191 – 205, 1994.

[67] J. Kratochvíl, A. Proskurowski, and J. Telle. Covering regular graphs. *Journal of Combinatorial Theory, Series B*, 71:1 – 16, 1997.

[68] E. Lawler. A note on the complexity of the chromatic number problem. *Information Processing Letters*, 5:66 – 67, 1976.

[69] D. F. Liu and X. Zhu. Circular distance two labeling and the λ-number for outerplanar graphs. *SIAM Journal on Discrete Mathematics*, 19:281 – 293, 2005.

[70] L. Lu. Graph Labeling with Distance Conditions and the Delta Squared Conjecture. Senior Thesis, University of South Carolina.

[71] T. Łuczak. personal communication.

[72] C. McDiarmid. Discrete mathematics and radio channel assignment. *CMS Books Math. Ouvrages Math. SMC*, 11:27–63, 2003.

[73] C. McDiarmid. On the span in channel assignment problems: bounds, computing and counting. *Discrete Mathematics*, 266:387 – 397, 2003.

[74] A. Meir and J. W. Moon. Relations between packing and covering numbers of a tree. *Pacific Journal of Mathematics*, 61:225 – 233, 1975.

[75] R. E. Miller and D. E. Muller. A problem of maximum consistent subsets. *IBM Research Report, Thomas J. Watson Research Center*, RC - 240, 1960.

[76] J. W. Moon and L. Moser. On cliques in graphs. *Israel Journal of Mathematics*, 3:23 – 28, 1965.

[77] P. Rossmanith. personal communication.

[78] P. Rzążewski. Exact Algorithm for Graph Homomorphism and Locally Injective Graph Homomorphism. *Information Processing Letters*, 114:387 – 391, 2014.

[79] B. Sagan. A note on independent sets in trees. *SIAM Journal on Discrete Mathematics*, 1:105 – 108, 1988.

[80] P. Seymour. Colouring series-parallel graphs. *Combinatorica*, 10:379 392, 1989.

[81] A. Socała. Tight lower bound for the channel assignment problem. *arXiv:1407.7162v1 [cs.DS]*, 2014.

[82] V. Strassen. Gaussian Elimination is not Optimal. *Numerische Mathematik*, 13:354 – 356, 1969.

[83] D. Sumner. Graphs with 1-Factors. *Proceedings of the AMS*, 42:8 – 12, 1974.

[84] J. M. M. van Rooij, H. L. Bodlaender, and P. Rossmanith. Dynamic Programming on Tree Decompositions Using Generalised Fast Subset Convolution. *Proc. of ESA 2009, LNCS*, 5757:566 – 577, 2009.

[85] V. Vizing. Vertex colorings with given colors. *Metody Diskret. Analiz. (in Russian)*, 29:3 – 10, 1976.

[86] M. Wahlström. A tighter bound for counting max-weight solutions to 2SAT instances. *Proc. of IWPEC 2008, LNCS*, 5018:202 – 213, 2008.

[87] D. Wood. On the number of maximal independent sets in a graph. *Discrete Mathematics and Theoretical Computer Science*, 13:17 – 20, 2011.

[88] R. K. Yeh. A survey on labeling graphs with a condition at distance two. *Discrete Mathematics*, 306:1217 – 1231, 2006.

[89] Y. Zhao. The Number of Independent Sets in a Regular Graph. *Combinatorics, Probability and Computing*, 19:315 – 320, 2010.

[90] X. Zhu. Circular chromatic number, a survey. *Discrete Mathematics*, 229:371 – 410, 2001.