INTRACTABILITY RESULTS FOR CERTAIN GRAPH–THEORETIC OPTIMIZATION AND APPROXIMATION PROBLEMS

A Thesis

By

G. VENKATESAN

Under the guidance of

Prof. C. Pandu Rangan

Submitted in partial fulfilment of the requirements
for the award of the degree of

Bachelor of Technology
in
Computer Science and Engineering

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY, MADRAS.

May 1997

CERTIFICATE

This is to certify that the thesis entitled **Intractability Results for Certain Graph–theoretic Optimization and Approximation Problems**, submitted by **G. Venkatesan** in partial fulfilment of the requirements for the award of the degree of **Bachelor of Technology**, is a bonafide record of the work carried out by him under my supervision and guidance at the Department of Computer Science and Engineering, Indian Institute of Technology, Madras.

Chennai 600 036
Date : May 1997

Prof. C. Pandu Rangan
Dept of Comp. Sci. & Engg.
Indian Institute of Technology,
Madras

Acknowledgements

We know too much for one man to know much.
- J. Robert Oppenheimer, *The Open Mind*

I would like to express my sincere thanks to my guide Prof. C. Pandu Rangan for his constant support, encouragement and advice, and for giving me complete freedom in choosing topics to work on. I can never thank him enough for all the trouble he took to make all required references available to me. I may forget the Algorithmics he taught me, but I will never forget the lo..ng chats I had with him on a variety of topics.

I thank Dr. S. A. Choudum and Dr. P. Sreenivasa Kumar for their excellent teaching of the courses Graph Theory and TFCS-II, which are probably the two best courses I have done during my B.Tech, even when seen without theoretical spectacles.

I would like to thank Dr. M. S. Chang and Dr. G. J. Chang for their help in my investigations relating to graph packing problems. I thank Dr. Balaji Raghavachari for his visit to our lab, which was one of the main factors that motivated me to learn more about the recent developments on (in)approximability.

I thank Madanlal for co-authoring my first paper (which is always a special one). I thank Niranjan for being a great friend and well-wisher and for his help during my initial adventures with LaTeX. I should also thank all the seniors who have passed out of the TCS lab, for their outstanding contributions have created an excellent reputation for the lab. It was a great challenge to try and live up to this reputation.

The Class of '97 was a great group to spend four years in. The bunch was comprised of predominantly South Indian members, but it offered enough variety to make it exciting and interesting, and I enjoyed the company of each and every one of its members. Of course, special mention should go to the co-inhabitants of my (the 5th) wing and my immediate neighbors Pisku, Sada and Vasan for making my everyday life in the hostel something to look forward to (though I did not see much of Pisku during the final year, understandably so:-). The K_3 formed by myself, Pisku and Nibble will probably hang about in my memory for some time.

The TCS lab was a great place to work and latex in. I thank all its members whose stay in the lab overlapped with mine, for being part of its wonderful atmosphere.

How can any acknowledgement be complete if I do not thank my parents and my sister whose love I breathe every moment?

Abstract

Intractability results for optimization problems complement algorithm design techniques by proving what cannot be computed efficiently and therefore suggests the use of alternative simplifications of the problem in order to efficiently solve it. The rationale behind this approach is that problems are seldom encountered in their utmost generality in practice. Two particularly relevant approaches to tackle "hard" problems are: (i) Restrict the problem to the case of better structured inputs, and (ii) Allow sub-optimal (but provably good) *approximate* solutions. The purpose of this thesis is to prove intractability results even under these two simplifications.

Graphs are the most pervasive discrete structures used to model computational problems and our work focuses on graph–theoretic problems. The first part of our work proves **NP**-hardness results corresponding to situation (i) above, where the input graphs are restricted to belong to special classes like bipartite, planar, chordal, split, bounded–degree etc. More specifically, we consider two optimization problems: (a) Minimum Graph Spanners, and (b) Graph Packing Problems.

Regarding graph spanners, we prove that the problem of finding a t-spanner with fewest possible edges is **NP**-hard on chordal and bounded–degree graphs for $t \geq 2$, on bipartite graphs for $t \geq 3$, and on split graphs for $t = 2$. Regarding packing problems, we prove that the problem of finding a maximum number of pairwise vertex–disjoint triangles is **NP**-hard on chordal, planar, line and total graphs, but can be solved in polynomial time on split graphs. We also prove that the problems of ascertaining the existence of a P_3-factor in degree-3 bipartite graphs and of a K_4-factor in comparability graphs are **NP**-complete.

The second part of the work is concerned with (in)approximability of optimization problems, developments relating to which are arguably the most significant ones in complexity theory during this decade. We first provide an example of a family of optimization problems which behave better and better with respect to approximability, but none of which are likely to admit a polynomial time approximation scheme. The family of problems we consider is the independent set problem on k–iterated total graphs for $k \geq 1$.

The other work along these lines concerns the approximability of domination problems which are among the most well-studied and useful graph–theoretic problems. Domination problem has several variants; while domination, connected domination and total domination can be approximated to a factor of $O(\log N)$, where N is the number of vertices, the other variants like independent and clique domination appear to be much harder to approximate. We first prove that independent domination cannot be approximated to a constant factor unless **P**=**NP** using a property called "self–improvement" which has found use for other problems like CLIQUE as well. Then, using problem–specific tech-

niques, we prove that approximating independent and clique dominating sets to a factor of $N^{1-\epsilon}$ is **NP**-hard for any $\epsilon > 0$. While this extreme behavior in terms of inapproximability is also shared by the CLIQUE problem (due to a recent result by Hastad [28]), we prove a still stronger negative result which states that both these problems cannot be approximated even to a factor of $N/\log^{1+\epsilon} N$, for any $\epsilon > 0$, unless the **NP**-complete problem 3SAT can be decided in sub–exponential deterministic time (which is considered unlikely). This is the **strongest negative result** we are aware of regarding the approximability of an unweighted **NP**-complete graph optimization problem.

Contents

List of Figures

Chapter 1

Introduction

The study of the difficulty of computational problems has two parts. The theory of algorithms is concerned with the design of efficient algorithms which prove upper bounds on the amount of computational resources required to solve a specific problem. Complexity theory, on the other hand, is concerned with proving the corresponding lower bounds. Our work is *mainly* part of the latter endeavor, i.e to prove problems computationally difficult, or *hard*.

Despite some notable success, lower bound research is still in a stage of infancy and fundamental separation questions like $\mathbf{P} \stackrel{?}{=} \mathbf{NP}$ remain unresolved. The interesting question that arises is what can one say about the difficulty of a problem if one assumes $\mathbf{P} \neq \mathbf{NP}$ or any other such complexity–theoretic statement. It is in this aspect that complexity theory has been extremely successful by introducing the notion of *complete* problems for *syntactic* complexity classes like **NP**. The most popular notion is that of **NP**–completeness. Complete problems capture the essence and difficulty of a class, they are a link that keeps complexity classes alive and anchored in computational practice. For example, the existence of important, natural problems that are complete for a class lends the class a significance that may not be immediately clear from its definition (**NP** is a case in point, and we discuss the significance of **NP**-complete problems next).

1.1 NP-complete Problems

An intuitive way of understanding **NP** is the following: each problem in **NP** has the property that any "yes" instance x of the problem has at least one *succinct certificate* (or *polynomial witness*) y of its being a "yes" instance. It is this characterization that explains why **NP** is inhabited by such a tremendous wealth of practically important, natural computational problems.

The study of the complexity of computational problems concerns itself for the most part with specimens in **NP**; it basically tries to sort out which problems can be solved in polynomial time, and which cannot. In this context, **NP**-completeness is a most important tool. A problem Π in **NP** is **NP**-complete if every problem in **NP** reduces in polynomial time to Π. Showing that the problem being studied is **NP**-complete establishes that it is among the least likely to be in **P**, those that can be solved in polynomial time only if **P**=**NP**. In this sense, **NP**-completeness is a valuable component of our methodology, one that complements algorithm design techniques. Once a problem has been shown **NP**-complete, it seems reasonable to direct our efforts to alternative approaches like designing approximation algorithms, attacking special cases etc[1].

1.2 Tackling NP-complete problems

As remarked in the previous section, it seems judicious to attempt to counter **NP**-complete problems through alternative approaches. Two of the most popular methods are:

- Find sub-optimal (but provably good) approximate solutions.
- Solve the problem for restricted sets of (better structured) inputs.

We consider **NP**-complete graph–theoretic problems (which form a sizeable fraction of known **NP**-complete problems owing to the pervasiveness of graphs in various applications), and prove hardness results even for these alternative approaches. For graph-theoretic problems, restricting inputs means that the input graph is constrained to belong to some special class of graphs; the classes of graphs that are considered in this thesis are described in section 1.4.

1.3 Graph–theoretic Preliminaries

All graphs we deal with, unless otherwise stated, are finite, simple and undirected. A graph G is usually denoted as $G = (V, E)$ where V is the set of vertices of G and E is the set of edges of G.

Two vertices u and v such that $(u, v) \in E$ are said to be adjacent (or neighbors) in G. The neighborhood of a vertex v, denoted $N_G(v)$, is the set of vertices in G adjacent to v, i.e $N_G(v) = \{u : (v, u) \in E\}$. The *closed* neighborhood of v, denoted $N_G[v]$, is given by $N_G[v] = N_G(v) \cup \{v\}$. The degree of v in G, denoted $d_G(v)$, is the number of neighbors of v in G, i.e $d_G(v) = |N_G(v)|$. We omit the subscript G when no confusion arises. We denote by $\Delta(G)$ and $\delta(G)$ the maximum and minimum degrees of a vertex in G, i.e $\Delta(G) = \max\{d_G(v) : v \in V\}$ and $\delta(G) = \min\{d_G(v) : v \in V\}$.

A path in G is a sequence of distinct vertices $v_1, v_2, \ldots, v_k$ such that $(v_i, v_{i+1}) \in E$ for $1 \leq i < k$. A cycle is a sequence of vertices $v_1, v_2, \ldots, v_k, v_1$ such that $v_1, v_2, \ldots, v_k$ is a path and $(v_k, v_1) \in E$. P_n (C_n) stands for the path (cycle) on n vertices. A graph is *connected* if there exists a path between any two vertices.

Given a subset S of V, the subgraph induced by S, denoted $G[S]$, is given as $G[S] = (S, E_S)$ where $E_S = \{(u, v) : u, v \in S \text{ and } (u, v) \in E\}$. A set $X \subseteq V$ is a *clique* if any

[1]There is nothing wrong with trying to prove that **P=NP** by developing a polynomial time algorithm for an **NP**-complete problem. The point is that without an **NP**-completeness proof we would be trying the same thing *without knowing it!*.

two vertices in X are adjacent, it is an *independent set* (or just independent) if no two vertices in X are adjacent and it is a vertex-cover if every edge in E has at least one end point in X, i.e if $V - X$ is an independent set. We denote by $\omega(G)$, $\alpha(G)$ and $\gamma(G)$ the size of a largest clique, largest independent set and smallest vertex-cover in G. Note that $\gamma(G) = |V| - \alpha(G)$ and $\omega(G) = \alpha(G^c)$ where G^c denotes the complement of G, i.e the graph $G^c = (V, E')$ where $e \in E'$ iff $e \notin E$. K_n stands for the complete graph (or clique) on n vertices.

1.4 Special Classes of Graphs

Any problem if sufficiently generalized will become **NP**-hard or worse, and any problem has special cases which fall in **P**. The interesting question therefore is to find the dividing line between the tractable and intractable versions of the problem. For graph–theoretic problems, the pertinent question is to find out the dividing line between **P** and **NP**-complete problems along a hierarchy of special classes of graphs. This section defines the special classes of graphs that are considered in this thesis, an excellent introduction to several of these classes of graphs can be found in [25].

A graph is *planar* if it can be embedded in the plane such that no two of its edges cross each other. A graph is *bipartite* if its vertex set can be partitioned into two independent sets, and it is *split* if its vertex set can be partitioned into a clique and an independent set. A graph is *chordal* if it has no induced cycle of length greater than three, it is easy to see that all split graphs are also chordal. A graph is an *interval* graph if it is the intersection graph of a family of intervals on the real line.

A *comparability* graph is a graph whose edges can be transitively oriented, a *cocomparability* graph is one whose complement is comparability. A *permutation* graph is a graph which is simultaneously comparability and cocomparability. It is also known that the class of interval graphs is precisely the class of chordal, cocomparability graphs [25].

If $G = (V, E)$, then let $E' = \{(e, v) : e \in E, v \in V$ and v is incident with $e\}$, and let $E'' = \{(e, f) :$ e and f are adjacent edges of $E\}$. The *line graph* of G, denoted $L(G)$, is defined as $L(G) = (E, E'')$ and the *total graph* of G, denoted $T(G)$, is defined as $T(G) = (V \cup E, E \cup E' \cup E'')$. A graph G is said to be a *bounded–degree* graph if there exists a *constant* K such that $\Delta(G) \leq K$.

1.5 Organization of the Thesis

The rest of the thesis is organized as follows. Chapter 2 considers the minimum spanner problem, mentions its current status on various classes of graphs and proves **NP**-hardness results on bipartite, chordal, split and bounded–degree graphs. Chapter 3 considers packing problems on graphs, which form a natural generalization of classical matching problems. It defines a parameter called the packing–hardness number of a class of graphs that serves as a measure of the difficulty of generalized matching problems on that class of graphs, and estimates the packing–hardness number of chordal, planar, bipartite, line and total graphs.

Chapters 4 and 5 deal with proving hardness results regarding the approximations of **NP**-hard problems. Chapter 4 provides an example of a hierarchical family of optimization problems which behave better and better with respect to approximations, but none of which are likely to admit a polynomial time approximation scheme. Chapter 5 considers the several variants of domination problems and proves hardness results for approximating these. Notable among these results are the strong negative results for independent and clique domination, which leads one to believe that these problems admit nothing other the trivial factor $O(N)$ approximation. Finally, chapter 6 concludes the thesis with some remarks and open problems.

Chapter 2

Complexity of Minimum Spanner Problems

2.1 Introduction

We prove hardness results relating to the concept of minimum graph spanners that has arisen in the theoretical analysis of computer networks. For a connected graph $G = (V, E)$, $H = (V, E')$ is a *spanning subgraph* iff $E' \subseteq E$. A t-spanner of a graph G is a spanning subgraph H of G in which the distance between every pair of vertices in H is at most t times their distance in G. A minimum t-spanner of a graph G is a t-spanner of G with minimum number of edges [43]. A tree t-spanner of G is a t-spanner that is a tree [11]. A tree t-spanner of G, if one exists, is clearly a minimum t-spanner of G.

The key ideas behind the notion of spanners is to approximate the pairwise distances in the original graph by a *sparse* spanning subgraph. Therefore, one of the fundamental problems in the study of spanners is to find a minimum t-spanner of a graph where $t \geq 1$ is a fixed integer (stretch factor). Finding a minimum 2-spanner is **NP**-hard for general graphs [43]. Later, Cai [12] proved that minimum t-spanner is **NP**-hard for each $t \geq 3$ using a rather complicated reduction from 3SAT. MINIMUM t-SPANNER, for each fixed $t \geq 2$, remains **NP**-hard on graphs with maximum degree equal to nine [13]. Because of these intractability results, the complexity of MINIMUM t-SPANNER for useful subclasses of graphs becomes interesting. Linear algorithms exist for the MINIMUM t-SPANNER problem on interval and permutation graphs for each fixed $t \geq 3$ [37], however the case $t = 2$ remains open for both these classes of graphs. Also, a linear algorithm for the MINIMUM 2-SPANNER problem on graphs with bounded degree ≤ 4 is presented in [13].

In this chapter, we prove hardness results for the minimum spanner problem on bipartite graphs, split graphs, chordal graphs and bounded–degree graphs. We denote by $s_t(G)$ the number of edges in a minimum t-spanner of G. Clearly, if G is connected, $s_t(G) \geq |V(G)| - 1$. For the rest of the discussion in this chapter, we assume that G is a connected graph.

2.2 A useful NP-hard problem

In this section we prove the **NP**-hardness of a useful problem which we will use in several reductions later on. Define the $2K_3$-edge cover of a graph G to be a set of edges S of G

such that every triangle in G has at least two edges from S. Also let $t_2(G)$ denote the size of a minimum $2K_3$-edge cover of G. We now prove:

Lemma 2.1 *It is* **NP***-hard to determine* $t_2(G)$ *of a general graph* G.

Proof : The reduction is from the minimum vertex cover problem on a general graph $H = (V', E')$ where $V' = \{1, 2, \ldots, n\}$ and $E' = \{(f_i, g_i) : 1 \leq i \leq m\}$ i.e f_i and g_i are the distinct ends of edge e_i for $1 \leq i \leq m$ where $m = |E'|$.

Form a graph G by adding a new path of length two between every pair of *adjacent* vertices of H and by adding a new vertex and making it adjacent to *all* vertices of H. More formally, $G = (V, E)$ where:

- $V = V' \cup \{w_i : 1 \leq i \leq m\} \cup z$
- $E = E' \cup \{(w_i, f_i), (w_i, g_i) : 1 \leq i \leq m\} \cup \{(z, i) : i \in V'\}$

It is easy to see that if $S \subseteq E$ is a minimum $2K_3$-edge cover of G, then S may be chosen so that $E' \subseteq S$ and exactly one of (w_i, f_i) and (w_i, g_i) is in S for $1 \leq i \leq m$. Hence if $E_z = \{(z, i) : i \in V'\}$ then $|S| = 2m + |S \cap E_z|$. If $S \cap E_z = \{(z, i_1), (z, i_2), \ldots, (z, i_p)\}$, then it is easy to see that S would be a minimum $2K_3$-edge cover of G if and only if $\{i_1, i_2, \ldots, i_p\}$ is a minimum vertex cover of H. Hence, if $\gamma(H)$ denotes the size of a minimum vertex cover of H, we have:

$$t_2(G) = 2m + \gamma(H)$$

and the result follows. □

2.3 Minimum Spanners on Bipartite Graphs

In this section, we prove that the MINIMUM t-SPANNER problem, for $t \geq 3$ is **NP**-hard on bipartite graphs. Our proof is way simpler than the one presented in [12]. Since a Bipartite graph G has no odd cycle, a subgraph H of G would be a $2k$-spanner of G iff it is $(2k-1)$-spanner of G. Hence if suffices to prove the result for odd $t \geq 3$. Also since the only 1-spanner and 2-spanner of a Bipartite graph G is G itself, the MINIMUM t-SPANNER problem for Bipartite Graphs is in **P** for $t = 1, 2$.

2.3.1 Definitions

Let us first define few terms.

Definition 2.1 *An edge* e *of* G *is a* t*-mandatory edge if the two ends of* e *are connected by a path* Q *of length* t *and all internal vertices of the path have degree 2. The path* Q *will be called a* t*-compelling path.*

Definition 2.2 *An induced path P of a graph G is called a t-mandatory path if all its edges are t-mandatory edges in G.*

We now observe the following lemma that allows us to *force* an edge in a t-spanner of a graph G.

Lemma 2.2 *If S is a MINIMUM t-SPANNER of a graph G, then S may be modified without changing its cardinality so that it will include all the t-mandatory edges and exactly $t-1$ edges at each t-compelling path.*

Proof: Follows immediately from the definition of a t-mandatory edge. □

2.3.2 The Hardness Result

Theorem 2.1 *For any fixed $t \geq 3$, determination of $s_t(G)$ is* **NP***-hard for a bipartite graph G.*

Proof : As mentioned earlier, it suffices to consider odd $t \geq 3$. The reduction is from the edge domination problem on bipartite graphs [44]. Let $H = (X, Y, E)$ be a bipartite graph with $X = \{x_1, x_2, \ldots, x_r\}$, $Y = \{y_1, y_2, \ldots, y_s\}$ and $|E| = m$. Let $\gamma_e(H)$ denote the size of a minimum edge dominating set of H ($MED(H)$).

Construct a graph $G = (V', E')$ as follows: G will initially have H as an induced subgraph. Add two vertices z_X, z_Y, and connect z_X (z_Y) to all vertices in X (Y) by distinct t-mandatory paths of length $(t-1)/2$ (by distinct we mean that the internal vertices of all such paths are distinct). Denote by P_i (Q_j) the t-mandatory path between z_X and x_i (z_Y and y_j). See Figure 2.1 that illustrates the above construction for $t = 3$.

Since t is odd and H is Bipartite, it is easy to see that G has no odd cycles and is also therefore Bipartite. Also G has $2+(r+s)t(t-1)/2$ vertices and $(r+s)(t^2-1)/2+m$ edges.

Let S be a minimum t-spanner of G. By lemma 2.2, we may assume that S is a minimum t-spanner of G that includes all the t-mandatory edges of G and exactly $t-1$ edges from each t-compelling path of G. Hence S will have $s_t(G)$ edges and $E(P_i) \subseteq E(S)$, $E(Q_j) \subseteq E(S)$ for $1 \leq i \leq r$, $1 \leq j \leq s$. So any two x_i's (or y_i's) will certainly be connected by a path of length $t-1$ in S and a necessary and sufficient condition to further ensure that S is a t-spanner is that $E(S) \cap E$ is an edge dominating set of H. Thus we have:

$$s_t(G) = \frac{(r+s)t(t-1)}{2} + \gamma_e(H)$$

and the result follows since G can be constructed in polynomial time from H. □

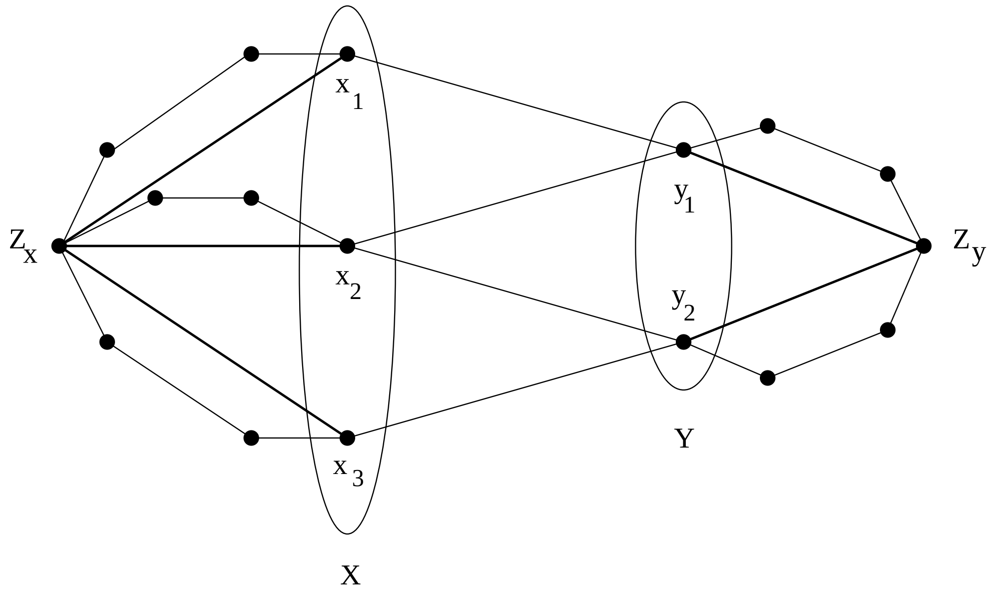

Figure 2.1: Construction of a Bipartite Graph.

2.4 Minimum Spanners on Chordal Graphs

We now obtain **NP**-hardness results on Chordal graphs. However, here we cannot force the choice of an edge in a t-spanner by simply adding a path of length t between its two ends as then we would have an induced cycle - a forbidden subgraph of Chordal graphs.

2.4.1 A useful sequence of graphs

We now define two sequences of graphs: $H_2, H_3, \cdots$ and $F_2, F_3, \cdots$ which will be used in the reductions employed in the **NP**-hardness proofs of this section.

Let H_2 be K_4, (the complete graph on 4 vertices), with a triangle (v_1, v_2, v_3) designated as the *bond triangle* and let the other three edges (c, v_i), $1 \leq i \leq 3$ be called *outer* edges. Let F_2 be K_3, (the complete graph on 3 vertices), with an edge (v_1, v_2) designated as the *bond edge* and the other two edges (c, v_1) and (c, v_2) be the *outer* edges.
Continue the sequence as follows:
Form H_{n+1} from H_n by bonding a new K_3 at each outer edge. The *bond* triangle of H_{n+1} is the same as that of H_n and the *outer* edges of H_{n+1} are the edges that are newly introduced while bonding the triangles. The newly introduced vertices become the outer vertices of H_{n+1}.
F_{n+1} is obtained from F_n in a similar way by bonding a new K_3 at each outer edge (here bond edges remain the same). See Figure 2.2 for an illustration of this procedure.
We observe the following:

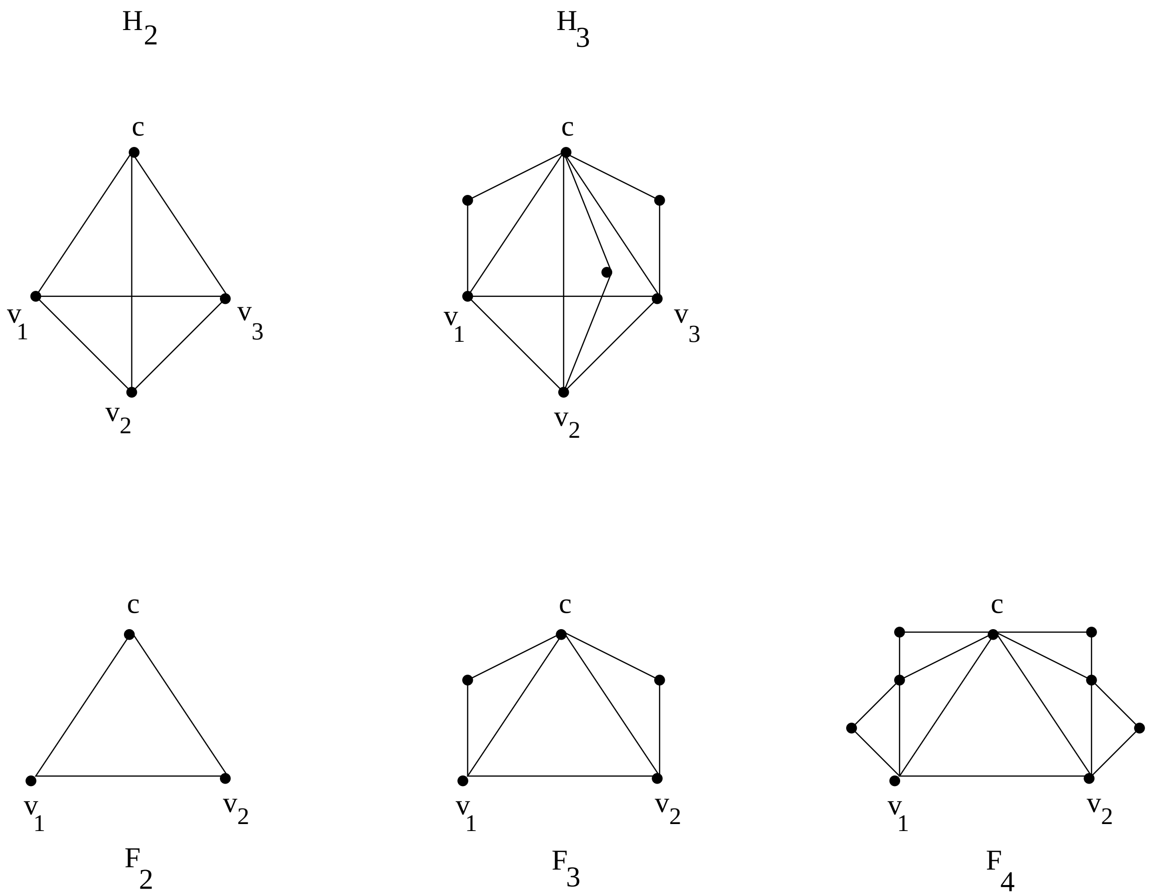

Figure 2.2: Construction of H_i and F_i.

Fact 2.1 *For $n \geq 2$, H_n has $3.2^{n-2}+1$ vertices and 3.2^{n-1} edges of which 3.2^{n-2} are outer edges. Moreover, H_n has 3.2^{n-3} outer vertices for $n \geq 3$.*

Fact 2.2 *For $n \geq 2$, F_n has $2^{n-1}+1$ vertices and 2^n-1 edges of which 2^{n-1} are outer edges. Moreover F_n has 2^{n-2} outer vertices for $n \geq 3$.*

For $t \geq 2$, denote by $B_e(H_t)$ ($B_v(H_t)$) the edges (vertices) of the bond triangle of H_t. Similarly $B_e(F_t)$ and $B_v(F_t)$ are defined (with bond triangle replaced by bond edge)

Lemma 2.3 *Let $G=(V,E)$ be a graph and let $H=(V',E')$ be obtained by identifying the bond edge of a* new *copy of F_t ($t \geq 2$) with an edge $e \in E$. Let S be a subgraph of H with minimum number of edges such that $d_S(x,y) \leq td_H(x,y)$, $\forall x,y \in V(F_t)$. Then:*

1. *S has 2^{t-1} edges.*
2. *S has at least $2^{t-1}-1$ edges from $E(F_t)-\{e\}$.*
3. *S may be chosen so that $e \in E(S)$.*

Proof : Proof proceeds by induction on t. All the three statements are clearly true for $t=2$. For $t>2$, S must have at least one edge incident with each outer vertex of F_t and so S must contain at least 2^{t-2} outer edges of the bonded F_t. Also if S contains both the edges incident at an outer vertex u of the bonded F_t, then one of the edges may be replaced by the third edge (not outer w.r.t F_t) of the unique triangle in F_t containing u. Thus any such S may be chosen to include a set Q of exactly 2^{t-2} outer edges of F_t.

If $O(F_t)$ denotes the set of outer vertices of F_t, then by our construction of $\{F_t\}_{t \geq 2}$, $F_t - O(F_t) \cong F_{t-1}$. Hence it is easy to see that S will satisfy the conditions of the lemma iff $S-O(F_t)$ satisfies the conditions of the lemma with t replaced by $t-1$ everywhere in the lemma.

Hence $|E(S)| = 2^{t-2} + |E(S-O(F_t))| = 2^{t-2} + 2^{(t-1)-1} = 2^{t-1}$ by the induction hypothesis proving 1. Also by the induction hypothesis, $S-O(F_t)$ must contain at least $2^{t-2}-1$ edges from $E(F_{t-1})-e$ and hence

$$|E(S) \cap (E(F_t)-e)| = 2^{t-2} + |E(S-O(F_t)) \cap (E(F_t)-e)| \geq 2^{t-2} + (2^{t-2}-1) = 2^{t-1}-1$$

proving 2. Part 3 follows from the fact that by induction hypothesis $S-O(F_t)$ may be chosen so that $e \in S-O(F_t)$. Note that part 3 implies that bonding of F_t at an edge e serves the purpose of *forcing* the edge in a t-spanner of the graph. □

Lemma 2.4 *Let $G=(V,E)$ be a graph and let $H=(V',E')$ be obtained by identifying the bond triangle of a* new *copy of H_t ($t \geq 2$) with a triangle T of G. Let S be a subgraph of H with minimum number of edges such that $d_S(x,y) \leq td_H(x,y)$, $\forall x,y \in V(H_t)$. Then:*

1. *S has at least $3 \times 2^{t-2}-2$ edges from $E(H_t)-B_e(H_t)$.*

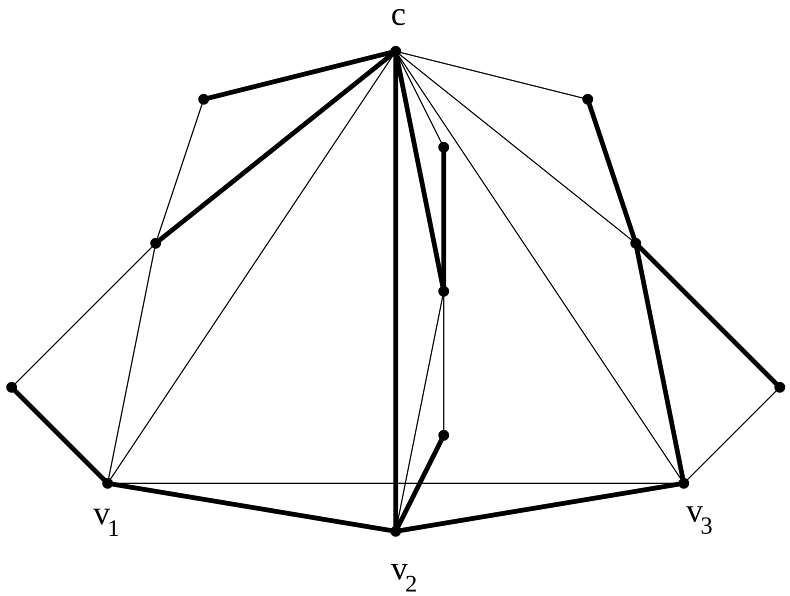

Figure 2.3: Spanner for $t = 4$.

2. S has $3 \times 2^{t-2}$ edges.

3. S may be chosen so that it includes two edges of the triangle T.

Proof : The proof proceeds along the same lines as that of Lemma 2.3. See Figure 2.3 for an illustration for the case $t = 4$ (edges present in S are marked bold). Note that this "bonding" has the effect of *forcing* the choice of at least two edges of T in a minimum spanner. □

2.4.2 NP-hardness on Chordal Graphs

Theorem 2.2 *The MINIMUM t-SPANNER problem is* **NP***-hard on Chordal graphs for all fixed $t \geq 2$.*

Proof : The reduction is from the problem of determining the size of a minimum $2K_3$-edge cover($t_2(H)$) of a general graph H, proved **NP**-hard in Lemma 2.1.

Let $H = (V, E)$ where $V = \{v_1, v_2, \ldots, v_n\}$. Form a graph G as follows:

Initially take a K_{n+1} containing vertices $\{z, u_1, u_2, \ldots, u_n\}$ where z is a special vertex and u_i corresponds to v_i for $1 \leq i \leq n$. For each edge $e_i = (z, u_i)$, $1 \leq i \leq n$, bond a *new* copy of F_t (using its bond edge) at e_i. Finally, for each triangle $(v_{i_1}, v_{i_2}, v_{i_3})$ in H, bond a *new* copy of H_t (using its bond triangle) at the triangle $(u_{i_1}, u_{i_2}, u_{i_3})$. The resulting graph is our desired G. See Figure 2.4 for an illustration of this construction for the case $t = 2$.

Since F_t and H_t are Chordal (by our construction) and all bonding operations are at either 2-cliques or 3-cliques it is straightforward to see that G is Chordal.

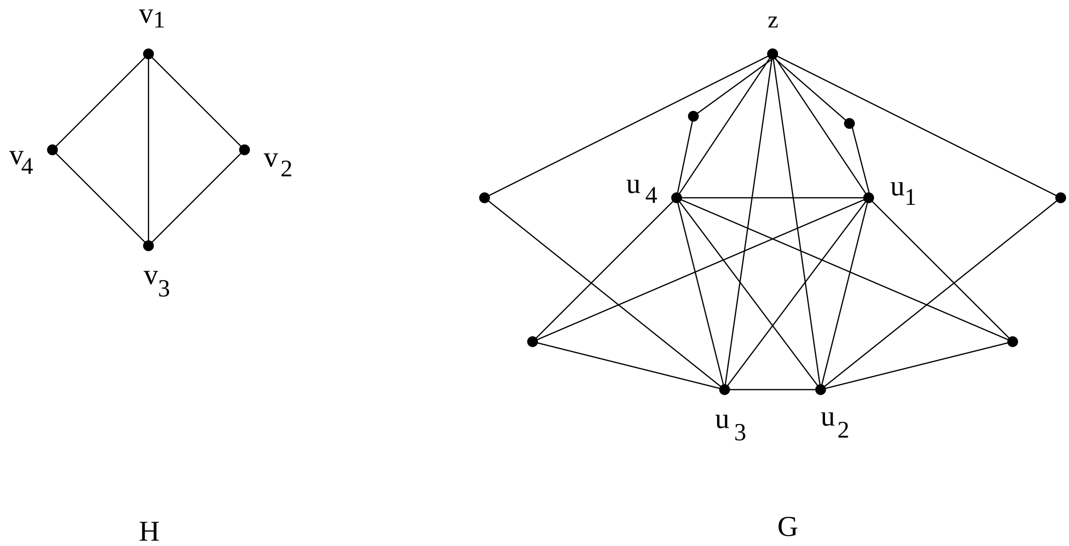

Figure 2.4: The Graph G for the case $t = 2$.

Let S be a minimum t-spanner of G. By Lemmas 2.3 and 2.4, it follows that S may be chosen to include each edge (z, u_i) for $1 \leq i \leq n$ (thus providing a path of length two between u_i and u_j for $1 \leq i < j \leq n$) and at least two edges from each triangle (u_i, u_j, u_k) $(1 \leq i < j < k \leq n)$ in G whenever (v_i, v_j, v_k) induces a K_3 in H. Since S is a minimum t-spanner of G, it will include exactly $t_2(H)$ edges of the form (u_p, u_q) $(1 \leq p < q \leq n)$ and using Lemmas 2.3 and 2.4 it follows that:

$$s_t(G) = |E(S)| = (2^{t-1} - 1)n + (3 \times 2^{t-2} - 2)c_3(H) + n + t_2(H)$$

where $c_3(H)$ denotes the number of triangles in H.

Clearly G can be constructed in polynomial time from H for fixed t and hence the theorem is proved. □

Corollary 2.1 *The MINIMUM 2-SPANNER problem is* **NP**-*hard even when restricted to Split graphs.*

Proof : Follows from the fact that the graph G constructed in the proof of Theorem 2.2 will be a Split graph if $t = 2$. □

Lemma 2.5 *The MINIMUM t-SPANNER problem can be solved in linear time on Split graphs for* $t \geq 3$.

Proof : Every connected split graph admits a tree 3-spanner, so the result follows. □

Combining the results of Theorem 2.2, corollary 2.1 and Lemma 2.5, we now state the following corollaries which completely settle the complexity status of the MINIMUM t-SPANNER problem on Split and Chordal graphs.

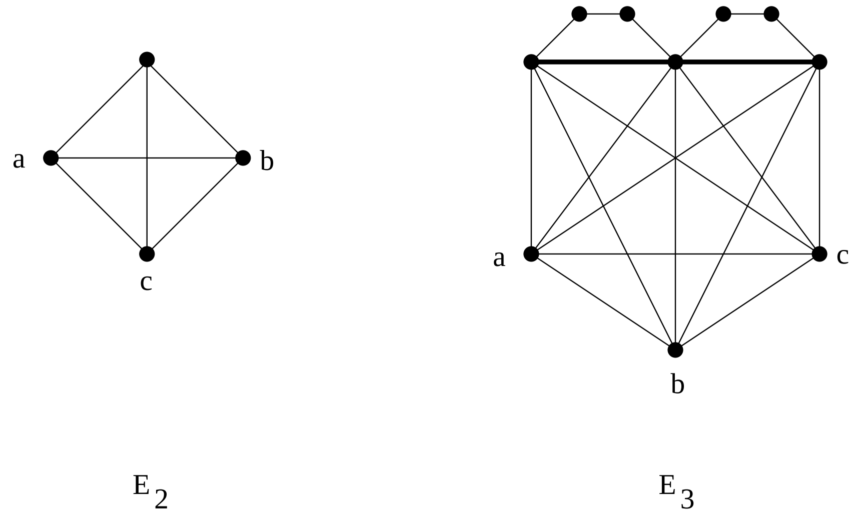

Figure 2.5: The Sequence E_t of Graphs.

Corollary 2.2 *The MINIMUM t-SPANNER problem on Split graphs is* **NP***-hard if* $t = 2$ *and is in* **P** *otherwise.*

Corollary 2.3 *The MINIMUM t-SPANNER problem on Chordal graphs is in* **P** *for* $t = 1$ *and is* **NP***-hard for all integers* $t > 1$*.*

2.5 Minimum Spanners on Bounded Degree Graphs

All **NP**-hardness proofs on the spanner problem presented so far produce graphs that have vertices of unbounded degree. In fact the complexity status of the tree t-spanner problem was left open for the class of Bounded Degree graphs in [11]. We now prove that the minimum spanner problem is **NP**-hard on bounded–degree graphs. Our proof is much simpler than the one presented in [13], but the bound on the degree obtained is poorer.

2.5.1 A useful sequence of graphs

As before, we first define a sequence of graphs $\{E_t\}_{t\geq 2}$ that serve as useful graphs to bond at triangles and force the choice of two out of the three edges of the triangle. For $t \geq 2$, construct E_t as follows: Start with a t-mandatory P_t path of length $2t-4$. Then take a new K_3, called the *bond triangle* of E_t and join the end vertices and the center vertex of P_t to all the three vertices of the bond triangle of E_t. Figure 2.5 shows the graphs E_t for the case $t = 2, 3$. $\{a, b, c\}$ induce the bond triangle. Let $B_e(E_t)$ denote the set of edges in the bond triangle of E_t.

Note that E_t has $2(t-1)^2 + 2$ vertices for $t \geq 2$ and $2(t^2 - t + 4)$ edges for $t \geq 3$. Also, $\Delta(E_t) = 7$ for $t \geq 3$ and $\Delta(E_2) = 3$ – so E_t is degree bounded (for $t \geq 2$).

Lemma 2.6 *Let $G = (V, E)$ be a graph and let $H = (V', E')$ be obtained by identifying the bond triangle of a copy of E_t ($t \geq 2$) with a triangle T of G. Let S be a subgraph of H with minimum number of edges such that $d_S(x, y) \leq t d_H(x, y)$ $\forall x, y \in V(E_t)$. Then S must have at least $(2t-4)t+3$ edges from E_t and may be chosen so that $|E(S) \cap B_e(E_t)| = 2$.*

Proof : Let e, g be the end vertices of the t-mandatory path P_t of E_t and let f be its center vertex. Without loss of generality assume that S has all the $2t-4$ t-mandatory edges and $(2t-4)(t-1)$ edges from the $2t-4$ t-compelling paths in E_t. Thus $|E(S)| \geq t(2t-4)$ and $d_S(e, f)$, $d_S(f, g) \leq t-2$. Since $\{e, f, g\} \times \{a, b, c\} \subset E(E_t)$ (note that $e = f = g$ for the case $t = 2$), it is not difficult to see that the optimal way to choose the remaining edges of S is to choose one of the three sets $\{(f, a), (a, b), (a, c)\}$, $\{(f, b), (b, a), (b, c)\}$ or $\{(f, c), (c, a), (c, b)\}$. The result now easily follows. $\square$

2.5.2 The Hardness Result

We are now ready to prove the main theorem of this section:

Theorem 2.3 *There is a constant k, such that the minimum t-spanner problem, for any fixed $t \geq 2$, is* **NP***-hard on graphs G with $\Delta(G) \leq k$.*

Proof : The reduction is from the vertex cover problem on cubic graphs [24]. Let $H = (V_1, E_1)$ be a cubic graph with $V_1 = \{1, 2, \ldots, n\}$, $E_1 = \{e_j = (p_j, q_j) : 1 \leq j \leq m\}$.

Form a graph $H_2 = (V_2, E_2)$ as follows: Initially take n independent edges $f_i = (x_i, y_i)$, for $1 \leq i \leq n$ as part of H_2 – these edges correspond to the vertices of H_1. Then for $1 \leq j \leq m$, corresponding to the edge e_j of H_2, add the following configuration shown in Figure 2.6 (called T_j) between the edges f_{p_j} and f_{q_j} of H_2. Note that a_j, b_j, c_j and u_{jk} (for $1 \leq k \leq 5$) are *new* vertices. Also $|V_2| = 8m + 2n$, $|E_2| = 19m + n$ and $c_3(H_2) = 10m$.

Clearly $\Delta(H_2) \leq 10$. Also we can ensure that, for each i, $1 \leq i \leq n$, if $\{e_{j_1}, e_{j_2}, e_{j_3}\}$ are the edges of H_1 incident at vertex i, then x_i is adjacent to say c_{j_1} and c_{j_2} and y_i is adjacent to c_{j_3}. This will make sure that $\Delta(H_2) = 9$. Moreover observe that a vertex is present in at most 8 triangles in H_2, $\omega(H_2) = 3$ and every edge of H_2 is present in some triangle of H_2.

Now consider the problem of finding a minimum $2K_3$-edge cover S of H_2, $|S| = t_2(H_2)$. Clearly S may be chosen so that, for the configuration of Figure 2.6, $P_j = \{(a_j, x_{p_j}), (c_j, y_{p_j}), (a_j, b_j), (c_j, x_{q_j}), (b_j, y_{q_j})\} \subset S$. Now in S, at least three more edges are needed from T_j so that the triangles (a_j, b_j, c_j), (a_j, x_{p_j}, y_{p_j}), (a_j, c_j, y_{p_j}), (c_j, b_j, x_{q_j}) and (b_j, x_{q_j}, y_{q_j}) have at least two edges in S and at least two of these edges which are used to *cover* the above five triangles must be from $Q_j = \{(a_j, c_j), (c_j, b_j), (a_j, y_{p_j}), (b_j, x_{q_j})\}$. Thus it is not difficult to see that the optimal way to choose S is that: $P_j \subset S$, $|Q_j \cap S| = 2$ and S includes at least one of f_{p_j} and f_{q_j} for $1 \leq j \leq m$. Hence,

$$|S| = t_2(H_2) = 12m + \gamma(H_1)$$

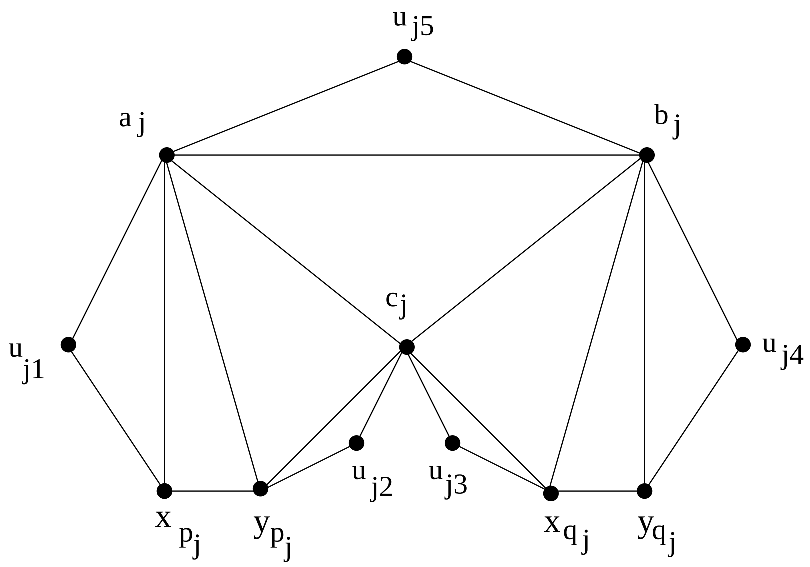

Figure 2.6: Configuration T_j for an edge (p_j, q_j).

where $\gamma(H_1)$ stands for the size of a minimum vertex cover of H_1.

Now form $G = (V, E)$ as follows: G will initially contain H_2 as an induced subgraph, and for each triangle T in H_2, bond a new copy of the graph E_t with H_2 by identifying its bond triangle with T. Let G be the resulting graph after all the bondings are completed. Let L be a minimum t-spanner of G. By Lemma 2.6, L may be chosen so that L has at least two edges from each triangle of H_2. Also since each edge of H_2 is in some triangle of H_2 by our construction, it follows that a $2K_3$-edge cover of H_2 will also be a t-spanner (in fact a 2-spanner) of H_2. Using the above facts and arguing as in theorem 2.2, it is a routine matter to verify that:

$$\begin{aligned} s_t(G) &= ((2t-4)t+1)c_3(H_2) + t_2(H_2) \\ &= (2t^2 - 4t + 1)10m + 12m + \gamma(H_1) \end{aligned}$$

Since a vertex in H_2 is present in at most 8 triangles of H_2, it is easy to check (using the construction of E_t) that $\Delta(G) = 9 + 3 \times 8 = 33$ if $t \geq 3$ and $\Delta(G) = 9 + 8 = 17$ if $t = 2$.

The result now follows with $k = 33$ since G can be constructed in polynomial time from H for fixed $t \geq 2$. $\square$

2.6 Conclusions

We have completely resolved the complexity status of the minimum t-spanner problem for all values of t on bipartite, chordal, split and degree–bounded graphs. The results are summarized in the table below:

Class	$t = 1$	$t = 2$	$t \geq 3$
Bipartite	**P**	**P**	**NP**-hard
Chordal	**P**	**NP**-hard	**NP**-hard
Split	**P**	**NP**-hard	**P**
Bounded-degree	**P**	**NP**-hard	**NP**-hard

The minimum t-spanner is solvable in linear time on interval and permutation graphs for $t \geq 3$ since these classes of graphs are tree 3-spanner admissible [37]. However the complexity of the MINIMUM 2-SPANNER is open on these classes of graphs. The complexity of the minimum t-spanner problem for $t \geq 2$ on **planar** graphs is also an interesting open problem.

Chapter 3

Complexity of Graph Packing Problems

3.1 Introduction and Terminology

The problem of packing (induced) copies of a *pattern* graph H into a given graph G has a long history [18, 29, 30, 35] and includes some of the most celebrated graph problems like maximum matching. These problems are encountered in practical problems like program optimization and examination scheduling [9, 35]. Several other applications are listed in [35] in its introduction.

If $\mathcal{F}$ is a family of *fixed* pairwise non-isomorphic graphs and $G = (V, E)$ is any graph, then $I_{\mathcal{F}}(G)$ stands for the set of all induced subgraphs of G each of which is isomorphic to some member of $\mathcal{F}$. For example if $\mathcal{F} = \{K_3\}$, then $I_{\mathcal{F}}(G)$ will be the set of triangles in G. A set $P \subseteq I_{\mathcal{F}}(G)$ is called a $\mathcal{F}$-packing if any two (induced) subgraphs of G belonging to P are vertex-disjoint. Henceforth, we shall refer to a vertex-disjoint $\mathcal{F}$-packing as just a $\mathcal{F}$-packing. A vertex $v \in V$ is said to be covered by the packing P if v belongs to the vertex set of some subgraph in P. The $\mathcal{F}$-packing problem consists of finding a $\mathcal{F}$-packing that covers the maximum number of vertices. If the vertices of $G = (V, E)$ have nonnegative weights associated with them, then the *weighted* $\mathcal{F}$-packing problem asks for a (vertex-disjoint) packing $P \subseteq I_{\mathcal{F}}(G)$ that covers a set of vertices in G with maximum total weight. If $\mathcal{F}$ consists of a single graph H, then we call the $\mathcal{F}$-packing problem as just the H-packing problem. Clearly, in this case the problem just reduces to finding a maximum number of pairwise vertex-disjoint induced copies of H in a given graph G.

A graph G is said to have a H-factor, if there exists a H-packing of G covering all vertices of G. For instance G has a K_2-factor if and only if G has a perfect matching. The edge-disjoint H-packing problem consists of finding a maximum number of pairwise edge-disjoint *partial* subgraphs isomorphic to H in a given graph G.

Though the classical matching problem can be efficiently solved even on general graphs [19, 40], the general H-packing problem appears to be very difficult. In fact, Kirkpatrick and Hell [35] have shown that if H is any connected graph with at least three vertices, then the H-packing problem is **NP**-complete. Subsequently, Berman *et. al.* [7] generalized this result by proving **NP**-completeness even for planar graphs. Related hardness results for edge-disjoint packing problems were proved by Corneil *et. al.* in [16].

Some work has also been done by restricting $\mathcal{F}$ to certain "common" families – families of certain cliques in [30], complete bipartite graphs in [31] and paths in [39, 14].

In this chapter, instead of fixing the "pattern" family $\mathcal{F}$, we try to estimate the *difficulty* of the H-packing problem when the input graphs are restricted to belong to certain "popular" classes. We do so, by associating with a class $\mathcal{C}$ of graphs, a number called the **packing-hardness number**. The details are mentioned in section 3.2.

3.2 The Packing-Hardness Number

In this section, we define a number associated with each class of graphs, called the packing-hardness number, that is meant to capture the difficulty of generalized matching problems on a particular class of graphs.

Definition 3.1 *Let $\mathcal{C}$ be a class of graphs. The* packing-hardness number *of the class $\mathcal{C}$, denoted $ph(\mathcal{C})$, is defined to be the minimum k such that there exists a k-vertex "pattern" graph H such that the H-packing problem is* **NP***-complete on the class $\mathcal{C}$ of graphs. If no such graph exists for any "fixed" k, then define $ph(\mathcal{C}) = \infty$*

One point needs mention at this moment: When we talk of **NP**-completeness of the H-packing problem for some H, we implicitly mean the decision version of the problem where we are given a goal $K \geq 0$ and the question is whether there are K or more vertex-disjoint copies of H in a given graph $G = (V, E)$.

Note that if $\bar{\mathcal{C}}$ denotes the class of graphs whose complements are in $\mathcal{C}$, then $ph(\bar{\mathcal{C}}) = ph(\mathcal{C})$ since the H-packing problem on $\mathcal{C}$ is polynomially equivalent to the H^c-packing problem on $\bar{\mathcal{C}}$.

Since the maximum matching problem is polynomial time solvable even on general graphs, we have $ph(\mathcal{C}) > 2$ for any class $\mathcal{C}$ of graphs. Hence it there exists a graph H with $|V(H)| = 3$ such that the H-packing problem is **NP**-complete on a class $\mathcal{C}$ of graphs, then we have $ph(\mathcal{C}) = 3$.

As we shall see later in this chapter, the packing-hardness number of several restricted classes of graphs is also finite, implying that the H-packing problem appears indeed to be very "hard" on even restricted classes of graphs. We determine the packing-hardness number of chordal, planar, line, total and bipartite graphs. The packing-hardness number of all these classes of graphs equals three, which seems to be the point where the difficulty of the problem really sets in. Some of the **NP**-hardness results presented generalize the results of [39]. We also present some interesting remarks on the packing-hardness number of split graphs.

3.3 NP-completeness results on K_3-packing

In this section we prove that the K_3-packing problem is **NP**-complete on chordal graphs, planar graphs, line graphs and on total graphs. This will immediately prove the following:

Theorem 3.1 *If $\mathcal{C}$ is the class of chordal, planar, line or total graphs, then $ph(\mathcal{C}) = 3$.*

3.3.1 Chordal Graphs

In the following, we will prove that the K_3-packing problem is **NP**-complete on chordal graphs. It is proved by reducing the 3-satisfiability problem to the K_3-packing problem on chordal graphs.

Problem. The 3-satisfiability problem (3SAT).
Instance: Collection $C = \{c_1, c_2, \cdots, c_m\}$ of clauses on a finite set U of variables such that $|c_i| = 3$ for $1 \leq i \leq m$.
Question: Is there a truth assignment for U that satisfies all the clauses in C?

We will first show how to construct a chordal graph from a 3SAT instance. Given an instance of 3SAT where $C = \{c_1, c_2, \cdots, c_m\}$ and $U = \{u_1, u_2, \cdots, u_n\}$, we will construct a chordal graph.

- For each variable u_i, we construct
  $W_i = \{w[i,j], \bar{w}[i,j] : 1 \leq j \leq m\}$,
  $X_i = \{x[i,j], \bar{x}[i,j] : 1 \leq j \leq m\}$,
  $S_i = \{s[i,j] : 1 \leq j \leq m\}$, and
  $T_i = \{t[i,j] : 1 \leq j \leq m\}$.
  Let
  $W = \cup_{i=1}^{n} W_i$,
  $X = \cup_{i=1}^{n} X_i$,
  $S = \cup_{i=1}^{n} S_i$, and
  $T = \cup_{i=1}^{n} T_i$.
- For each clause c_j, we construct a vertex $y[j]$. Let $Y = \{y[j] : 1 \leq j \leq m\}$.

We will construct a chordal graph $G_C = (X \cup W \cup K, KK \cup WK \cup XW \cup XY)$ where $K = S \cup T \cup Y$ is a clique, both W and X are independent sets, KK is the set of all edges connecting two vertices of K, XW is the set of edges connecting a vertex in X and a vertex in W, XY is the set of edges connecting a vertex in X and a vertex in Y, and WK is the set of edges connecting a vertex in W and another vertex in K. We have shown how to construct S, T, and K. Totally, vertex set K has $2mn + m$ vertices and both vertex sets W and X have $2mn$ vertices each. Next, we show how to construct the edge sets XW, XY, and WK. Recall that every edge of WK connects a vertex of W and a vertex of K and so on.

- Each vertex $w[i,j]$ (resp. $\bar{w}[i,j]$) is connected to a vertex $x[i,j]$ (resp. $\bar{x}[i,j]$). Let $XW = \{(x[i,j], w[i,j]), (\bar{x}[i,j], \bar{w}[i,j]) : 1 \leq i \leq n, 1 \leq j \leq m\}$.
- Each vertex $w[i,j]$ is connected to vertex $s[i,j+1]$ if $1 \leq j < m$ and each vertex $w[i,m]$ is connected to vertex $s[i,1]$. Each vertex $\bar{w}[i,j]$ is connected to vertex $s[i,j]$. Each vertex $t[i,j]$ is connected to vertices $w[i,j]$ and $\bar{w}[i,j]$. That is, for

each variable u_i, we construct the following two sets of edges :

$$\begin{aligned} WS_i &= \{(w[i,j], s[i,j+1]), (\bar{w}[i,j], s[i,j]) : 1 \leq j < m\} \cup \\ &\quad \{(w[i,m], s[i,1]), (\bar{w}[i,m], s[i,m])\} \end{aligned}$$

and

$$WT_i = \{(t[i,j], w[i,j]), (t[i,j], \bar{w}[i,j]) : 1 \leq j \leq m\}.$$

Let
$WS = \cup_{i=1}^{n} WSi$ and $WT = \cup_{i=1}^{n} WTi$.

- For each $u_i \in c_j$ (resp. $\bar{u}_i \in c_j$), $y[j]$ is connected to $w[i,j]$ and $x[i,j]$ (resp. $\bar{w}[i,j]$ and $\bar{x}[i,j]$). We thus have:

$$WY = \{(w[i,j], y[j]) : u_i \in c_j\} \cup \{(\bar{w}[i,j], y[j]) : \bar{u}_i \in c_j\} \text{ and,}$$

$$XY = \{(x[i,j], y[j]) : u_i \in c_j\} \cup \{(\bar{x}[i,j], y[j]) : \bar{u}_i \in c_j\}$$

- Let $WK = WS \cup WT \cup WY$.

It is straightforward to verify that $G_C = (X \cup W \cup K, KK \cup WK \cup XY \cup XW)$ is a chordal graph and can be constructed in polynomial time.

Totally, there are $4mn + m(2n+1)$ vertices in G_C. We now relate $t(G_C)$ with the satisfiability of the given 3SAT instance through the following lemma:

Lemma 3.1 *The 3SAT instance has a truth assignment for U that satisfies all clauses in C if and only if G_C has $m(n+1)$ vertex-disjoint triangles.*

Proof: First, we show that if C is satisfiable, then G_C has a set $\mathcal{T}$ of $m(n+1)$ vertex-disjoint triangles. Suppose u_i is assigned a false value. Then, we have the m triangles, $\{\{w[i,j], s[i,j+1], t[i,j]\} : 1 \leq j < m\} \cup \{\{w[i,m], s[i,1], t[i,m]\}\}$. Suppose u_i is assigned a true value. Then, we have the m triangles, $\{\{\bar{w}[i,j], s[i,j], t[i,j]\} : 1 \leq j \leq m\}$. This process can be used to add mn triangles in $\mathcal{T}$. Now, for each clause c_j, $1 \leq j \leq m$, there exists a variable to satisfy it. Suppose $u_i \in c_j$ and u_i is assigned a true value. Then, $w[i,j]$ is adjacent to both $x[i,j]$ and $y[j]$. Thus, $\{w[i,j], x[i,j], y[j]\}$ is a triangle that can be added to $\mathcal{T}$. Similarly, $\{\bar{w}[i,j], x[i,j], y[j]\}$ is a triangle that can be used in $\mathcal{T}$ if $\bar{u}_i \in c_j$ and u_i is assigned a false value. This gives a further m triangles for $\mathcal{T}$ and thus, there are $m(n+1)$ vertex-disjoint triangles in G_C if C is satisfiable.

On the other hand, suppose G_C has a set $\mathcal{T}$ of $m(n+1)$ vertex-disjoint triangles. Since every triangle in G_C has at least one vertex from K and there are at most m vertex-disjoint triangles in G_C with exactly one vertex from K, there must be *at least* mn triangles in $\mathcal{T}$ that have a minimum of two vertices from K. Thus, it is easy to see that no triangle in $\mathcal{T}$ can have all three vertices from K.

Now observe the following: (1) there does not exist a triangle in $\mathcal{T}$ which contains two vertices in S or two vertices in T, and (2) if a triangle in $\mathcal{T}$ contains exactly one vertex in K, then it is a vertex in Y. Thus, each triangle in $\mathcal{T}$ contains a vertex $y \in Y$ or two vertices in K with one in S and the other in T.

Since there are $m(n+1)$ vertex-disjoint triangles in $\mathcal{T}$, by the preceding discussion, $\mathcal{T}$ must have m triangles each of them containing a vertex in W, a vertex in X and a vertex in Y. Other triangles contains a vertex of W, a vertex of S and a vertex of T. Consider the subgraph $G_C[i]$ induced by $W_i \cup S_i \cup T_i$. $G_C[i]$ must contains m triangles in $\mathcal{T}$. The triangles in $G_C[i]$ can be divided into two sets:

$$\mathcal{T}_i^t = \{\{\bar{w}[i,j], s[i,j], t[i,j]\} : 1 \leq j \leq m\} \text{ and,}$$

$$\mathcal{T}_i^f = \{\{w[i,j], s[i,j+1], t[i,j]\} : 1 \leq j < m\} \cup \{\{w[i,m], s[i,1], t[i,m]\}\}.$$

Since no vertices in $\{s[i,j], t[i,j] : 1 \leq j \leq m\}$ will appear in any triangle of $\mathcal{T}$ outside of $\mathcal{T}_i = \mathcal{T}_i^t \cup \mathcal{T}_i^f$, it is easy to see that $\mathcal{T}$ will have to include exactly m triangles from $\mathcal{T}_i$, either all triangles in $\mathcal{T}_i^t$ or all triangles in $\mathcal{T}_i^f$.

Thus, in general $\mathcal{T}$ specifies a truth assignment for U, with the variable u_i being set true if and only if $\mathcal{T} \cap \mathcal{T}_i = \mathcal{T}_i^t$.

For each clause $c_j \in C$, the set of triangles containing exactly one vertex $y[j]$ in K and the other two vertices in X and W is:

$$\mathcal{C}_j = \{\{w[i,j], x[i,j], y[j]\} : u_i \in c_j\} \cup \{\{\bar{w}[i,j], \bar{x}[i,j], y[j]\} : \bar{u}_i \in c_j\}.$$

Thus $\mathcal{T}$ will have to contain exactly one triangle from $\mathcal{C}_j$. This can be done if some $w[i,j]$ (resp. $\bar{w}[i,j]$) for a literal $u_i \in c_j$ (resp. $\bar{u}_i \in c_j$) does not occur in the triangles in $\mathcal{T} \cap \mathcal{T}_i$, which will be the case if and only if the truth setting determined by $\mathcal{T}$ satisfies clause c_j.

We have shown that (1) if C is satisfiable, then G_C contains $m(n+1)$ vertex-disjoint triangles, and (2) if G_C contains $m(n+1)$ vertex-disjoint triangles, then C is satisfiable. Thus, the lemma is proved. □

By lemma 3.1 and the fact that G_C can be constructed in polynomial time from the 3SAT instance, we get:

Theorem 3.2 *The K_3-packing problem is* **NP**-*complete on chordal graphs.*

3.3.2 Planar Graphs

We now prove that the K_3-packing problem is **NP**-complete on planar graphs. A much more general result appears in [7], however our technique is simpler and also has applications in later hardness results.

Theorem 3.3 *The K_3-packing problem is* **NP**-*complete on planar graphs.*

Proof : The reduction is from the the independent set problem on planar cubic graphs [24]. Let $H = (V, E)$ be an arbitrary planar cubic graph. We reduce the problem of determining $\alpha(H)$ to the problem of finding $t(G)$ – the maximum number of vertex-disjoint triangles in G – for a suitably constructed planar graph G.

Let $|V| = n$ and $|E| = m$. First form $H' = (V', E')$ (H' will not be simple) from H as follows:
For $1 \leq i \leq m$, Insert two new vertices u_i and w_i in the edge e_i of H and add one more edge between u_i and w_i. More precisely, if $V = \{1, 2, \cdots, n\}$ and $E = \{e_1, e_2, \cdots, e_m\}$ and
$e_i = (f(i), g(i))$ for $1 \leq i \leq m$
Define $V' = V \cup \{u_i, w_i | 1 \leq i \leq m\}$
$E_i = \{(f(i), u_i), (w_i, g(i)), p_i, q_i\}$ where p_i, q_i are the two edges between u_i and w_i in H'
$E' = \cup_{i=1}^{m} E_i$

Clearly H' has no triangle, is planar and is 3-regular. Though, line graphs are defined in the strict sense only for simple graphs, we can speak of the line graph of H' - $L(H')$ whose vertices correspond to edges of H' and two vertices in $L(H')$ are adjacent iff the corresponding edges share <u>at least</u> one vertex in common. We set $G = L(H')$. It is not difficult to see that G will also be planar and $\Delta(G) = 4$. Since H' is triangle-free and 3-regular, triangles in $L(H')$ will comprise of the 3 edges incident at a vertex of H'. Also two triangles T_1, T_2 in $L(H')$ will be vertex-disjoint iff the vertices to which T_1 and T_2 correspond are not adjacent in H'. It therefore follows that :
$t(G) = t(L(H')) = \alpha(H')$
Let $\gamma(G)$ denote the size of the minimum vertex-cover of G.
Clearly $\gamma(G) = |V(G)| - \alpha(G)$.
We now prove that $\gamma(H') = \gamma(H) + m$.

Let S be a minimum vertex-cover of H, $S = \{i_1, i_2, \cdots, i_k\}$. In H', for $1 \leq j \leq m$ set $z_j = u_j$ or w_j depending upon whether $g(j) \in S$ or $g(j) \notin S$ (Note that in the latter case $f(j) \in S$ since S is a vertex-cover)
Now, $T = S \cup \{z_i | 1 \leq i \leq m\}$ is clearly a vertex-cover of H' , hence
$\gamma(H') \leq \gamma(H) + m$.

Conversely, let T be a minimum vertex-cover of H'. Clearly $T \cap \{u_i, w_i\} \neq \phi$ for $1 \leq i \leq m$. Also if both $u_i, w_i \in T$, then we can replace u_i by $f(i)$ in T. So we may assume that T is such that $|T \cap \{u_i, w_i\}| = 1$ for $1 \leq i \leq m$. Let $S = T \cap V$. We claim S is a vertex-cover of H. If not there exists $e_i = (f(i), g(i))$ such that $f(i), g(i) \notin S$, but then one of the edges $(f(i), u_i), (w_i, g(i))$ in H' is not covered by any vertex in T, a contradiction. Hence,
$\gamma(H') \geq \gamma(H) + m$.
Thus, $\gamma(H') = \gamma(H) + m$ or equivalently $\alpha(H') = \alpha(H) + m$. Hence $t(G) = \alpha(H) + m$. The conclusion now follows since G is planar with $\Delta = 4$ and can be constructed from H in polynomial time. □

3.3.3 Line Graphs

We now prove that the K_3-packing problem is **NP**-complete on line graphs (Note that line graphs are defined *only* for simple graphs).

Lemma 3.2 *If G is a triangle-free 3-regular simple graph and $L(G)$ is its line graph, then $t(L(G)) = \alpha(G)$.*

Proof : Since G is triangle-free, a triangle T in $L(G)$ will comprise of the three edges incident at a vertex v_T in G. The result now follows since triangles T_1, T_2 will be vertex-disjoint in $L(G)$ iff v_{T_1} and v_{T_2} are not adjacent in G. □

Theorem 3.4 *The independent set problem is* **NP**-*complete on triangle-free 3-regular graphs.*

Proof : The reduction is once again from the independent set problem on planar cubic graphs [24].
Let $H = (V', E')$ be an arbitrary planar cubic graph. We reduce the problem of determining $\alpha(H)$ to the problem of finding $\alpha(G)$ for a suitably constructed triangle-free cubic graph G.
Let $V' = \{1, 2, \cdots, n\}$ and $E' = \{e_1, e_2, \cdots, e_m\}$ and $e_i = (f(i), g(i))$ for $1 \leq i \leq m$
Define $G = (V, E)$ as follows : For $1 \leq i \leq m$, set
$U_i = \{u_{ij} | 1 \leq j \leq 8\}$ and
$E_i = \{(f(i), u_{i1}), (g(i), u_{i8})\} \cup \{(u_{ij}, u_{i,j+1}) | 1 \leq j < 8\} \cup \{(u_{i8}, u_{i1})\}$
$\quad \cup \{(u_{ij}, u_{i,j+3}) | j = 2, 3, 4\}$
Now define :
$E = \cup_{i=1}^{m} E_i$ and
$V = V' \cup (\bigcup_{i=1}^{m} U_i)$
Note that $|V| = 8m + n$ and $|E| = 13m$.
Informally, in G we attach a 8-cycle with certain specific diagonals instead of each edge of H. From the construction it is clear that G is 3-regular and has no triangle.

Using arguments exactly as in the proof of theorem 3.3, we can prove that a minimum vertex-cover S of G may be assumed to include exactly one of the vertices u_{i1}, u_{i8} and exactly four vertices from U_i for $1 \leq i \leq m$. Since S is a minimum vertex-cover, S will therefore include exactly $\gamma(H)$ vertices from V'.
Hence it follows that $\gamma(G) = \gamma(H) + 4m$ or equivalently,
$\alpha(G) = \alpha(H) + 4m$

The result now follows since G can be constructed from H in polynomial time. □

Theorem 3.5 *The K_3-packing problem is* **NP**-*complete on line graphs.*

Proof : Follows from lemma 3.2 and theorem 3.4. □

3.3.4 Total graphs

Theorem 3.6 *The K_3-packing problem is* **NP**-*complete even when the inputs are restricted to total graphs.*

Proof: We employ a reduction from the independent set problem on a 3-regular triangle-free graph $H = (V, E)$ which is **NP**-complete due to theorem 3.4. Let $V = \{v_1, v_2, \ldots, v_n\}$.

From H obtain a graph H' by attaching a new pendant vertex to each vertex of H. More formally, $H' = (V', E')$ where:

$$V' = V \cup \{u_i | 1 \leq i \leq n\} \text{ and}$$

$$E' = E \cup \{q_i = (u_i, v_i) | 1 \leq i \leq n\}$$

Now, let $G = T(H')$ be the total graph of H'. Let P be a set of maximum number of pairwise vertex-disjoint triangles of G. By the nature of the graph H', it is easy to see that P may be modified without reducing its cardinality to include all the triangles T_i of the form (u_i, q_i, v_i) of G for $1 \leq i \leq n$. Since $V' \subset \bigcup_{i=1}^{n} T_i$, clearly the optimum way to find the remaining triangles for P, is to find a set of maximum number of vertex-disjoint triangles of $L(H)$.

Thus, we have:

$$t(T(H')) = n + t(L(H))$$

But, by lemma 3.2, we have $t(L(H)) = \alpha(H)$ and hence,

$$t(T(H')) = n + \alpha(H)$$

The result now follows since determining $\alpha(H)$ is **NP**-complete and $G = T(H')$ can be constructed in polynomial time from H. $\square$

3.4 Bipartite Graphs

We now proceed to determine the packing-hardness number of bipartite graphs. Recall that P_3 stands for the path on three vertices.

Theorem 3.7 *It is* **NP**-*complete to determine whether a bipartite graph with $\Delta = 3$ has a P_3-factor.*

Proof : Presence in **NP** is clear. We prove **NP**-hardness by presenting a polynomial transformation from a restricted version of the EXACT COVER BY 3-SETS (X3C) problem. Recall that the X3C problem asks – given a finite set X with $3n$ elements and a collection C of m 3-element subsets of X, whether C contains an exact cover of X, i.e. a sub-collection $C' \subseteq C$ such that every element of X occurs in exactly one member of C'. This problem is **NP**-complete, even if we add the restriction that no element of X occurs in more than three 3-element subsets in C [24].

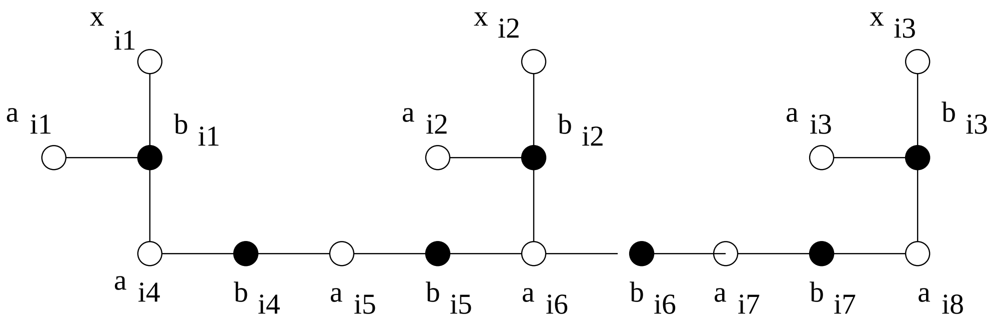

Figure 3.1: The Configuration H_i

Let $I = (X, C)$ be an instance of the restricted version of X3C, with $X = \{x_1, x_2, \ldots, x_{3n}\}$ and $C = \{S_i : 1 \leq i \leq m\}$ with $S_i = \{x_{i1}, x_{i2}, x_{i3}\}$. Construct a graph $G = (V, E)$ as follows:

Initially G will contain X as an independent set. Then for each i, $1 \leq i \leq m$, bond the configuration of figure 3.1 (called H_i) at the vertices x_{i1}, x_{i2} and x_{i3}.

Clearly G has $15m+3n$ vertices and $17m$ edges and can be constructed in polynomial time from the instance I of X3C. Also, G is clearly bipartite with $X \cup \{a_{ij} : 1 \leq i \leq m, 1 \leq j \leq 8\}$ being vertices of one bipartition and $\{b_{ij} : 1 \leq i \leq m, 1 \leq j \leq 7\}$ being the vertices of the other. Since no element of X occurs in more than three S_i's, clearly $\Delta(G) = 3$.

We now prove that G has a P_3-factor if and only if C contains an exact cover for X.
(if) : Let C contain an exact cover C' for X. For $1 \leq i \leq m$, if $S_i \in C'$, then choose the following P_3's from H_i :

$$T_i = \{(a_{ij}, b_{ij}, x_{ij}) : 1 \leq j \leq 3\} \cup \{(a_{ij}, b_{ij}, a_{i,j+1} : j = 4, 7\} \cup \{(b_{i5}, a_{i6}, b_{i6})\}$$

If $S_i \notin C'$, then choose the following P_3's from H_i,

$$F_i = \{(b_{i4}, a_{i5}, b_{i5}), (b_{i6}, a_{i7}, b_{i7}), (a_{i1}, b_{i1}, a_{i4}), (a_{i2}, b_{i2}, a_{i6}), (a_{i3}, b_{i3}, a_{i8})\}$$

Since C' is an exact cover for X, the various P_3's chosen are vertex-disjoint and cover all the vertices of G, i.e. G has a P_3-factor.
(only if): Conversely suppose G has a P_3-factor P. From the structure of H_i, it is easy to deduce that the only way to cover all vertices in $V(H_i) - \{x_{i1}, x_{i2}, x_{i3}\}$ using vertex-disjoint P_3's in G, is to choose either all the P_3's in T_i or all those in F_i. For instance, if (a_{i1}, b_{i1}, x_{i1}) is chosen in P, this forces the choice of (a_{i4}, b_{i4}, a_{i5}) in P. Now, if $(b_{i5}, a_{i6}, b_{i2}) \in P$, then a_{i2} cannot be covered by P, a contradiction, so $(b_{i5}, a_{i6}, b_{i6}) \in P$. This forces the choice of (a_{i2}, b_{i2}, x_{i2}), $(a_{i7}, b_{i7}, a_{i8}$ and (a_{i3}, b_{i3}, x_{i3}) in P. Thus we are forced to choose all P_3's in T_i. Similarly, it can be shown that if $(a_{i1}, b_{i1}, x_{i1}) \notin P$, then we will be forced to choose all P_3's in F_i for the P_3-factor P.

Hence if P is a P_3-factor of G, then $\forall i$, $1 \leq i \leq m$, either $T_i \subseteq P$ or $F_i \subseteq P$. Form an exact cover C' for X as follows: for $1 \leq i \leq m$, place S_i in C' if and only if $T_i \subseteq P$.

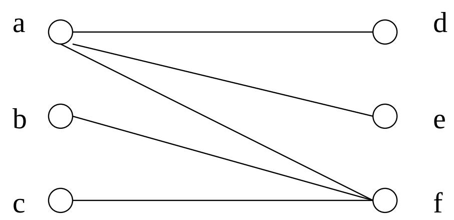

Figure 3.2: An Example where factor 1/2 is achieved

Thus G has a P_3-factor if and only if C contains an exact cover for X and the proof is completed. $\square$.

As a corollary, we obtain:

Corollary 3.1 *The packing-hardness number of the class of bipartite graphs equals three.*

However, we now prove that the P_3-packing problem on bipartite graphs can be approximated to a constant factor in polynomial time. More specifically, we prove:

Theorem 3.8 *There exists a polynomial time factor 1/2 approximation algorithm for the P_3-packing problem on bipartite graphs.*

Proof: Let $G = (X, Y, E)$ be a bipartite graph. The matching-based approach which will be later used to solve the K_3-packing on split graphs (see section 3.5) can easily be adapted to find a packing S_1 (resp. S_2) of maximum number of vertex-disjoint induced P_3's in G each of which has only one vertex from X (resp. Y). Using the same approach, find a packing T_1 (resp. T_2) of maximum number of vertex-disjoint induced P_3's in $G - \{v : v \text{ is covered by } S_1\}$ (resp. $G - \{v : v \text{ is covered by } S_2\}$). Output the packing $S_1 \cup T_1$ or $S_2 \cup T_2$ – whichever has larger cardinality. Let k be the size of an optimum P_3-packing. Clearly $\max\{|S_1|, |S_2|\} \geq k/2$. Also evidently the set output by the proposed algorithm has size $\geq \max\{|S_1|, |S_2|\}$. The result therefore follows. It is also easy to prove the efficacy analysis is tight by providing an example where it is possible that $|T_1| = |T_2| = 0$ and $|S_1| = |S_2| = k/2$. Indeed for the graph of figure 3.2, where $X = \{a, b, c\}$ and $Y = \{d, e, f\}$, we clearly have $k = 2$, however it is possible that $S_1 = \{(a, d, f)\}$ and $S_2 = \{(f, b, c)\}$. $\square$

3.5 Split Graphs

We now proceed to determine the possible values of the packing-hardness number of the class of split graphs.

Theorem 3.9 *The K_4-packing problem is* **NP**-*complete on split graphs.*

Proof: This is a straightforward reduction from the triangle cover problem on a general graph $G = (V, E)$ with $|V| = 3n$, [24]. The triangle cover problem asks whether there exist n vertex-disjoint triangles in G i.e whether G has a K_3-factor.

Consider the split graph $H = (S \cup K, E')$ where $K = V$ is a clique and vertices in the independent set S correspond to triangles in G. Connect each vertex $s \in S$ to the three vertices of the K_3 in G to which s corresponds. It is clear that H has K_4-packing of size n if and only if G has a triangle cover. □

However, we now prove that if the input graph is restricted to split graphs, the K_3-packing problem can be solved in polynomial time. This is one of the rare occasions when an algorithm is presented in this thesis.

K_3-Packing on Split Graphs

Suppose $G = (V, E)$ is a split graph where V is the *disjoint* union of an independent set S and a clique K. Clearly it is enough to find the maximum number of vertex-disjoint triangles in G such that each triangle has one vertex from S and two from K (the optimal way to find some more triangles all of whose vertices are in K to form the optimal triangle packing is obvious) and so we will consider this problem from now on. Henceforth, we shall denote by $tS(G)$ the maximum number of vertex-disjoint triangles in G each of which has one vertex from S.

Since K induces a clique, the following lemma is obvious (recall that $t(G)$ denotes the maximum number of vertex-disjoint triangles in G):

Lemma 3.3 *If $G = (V, E)$ is a split graph, then:*

$$t(G) = tS(G) + \left\lfloor \frac{|K| - 2 \times tS(G)}{3} \right\rfloor$$

We now proceed to map the vertex-disjoint problem on a split graph $G = (V, E)$ with $V = S \cup K$ as above, to the *maximum matching* problem on a suitably defined graph H. The construction of the graph H is described below:

For each $u \in S \subseteq V$, add a new vertex u' that is adjacent to all and only vertices to which u is adjacent in G. Also "join" u and u' by an edge. Also make K an independent set in H. Let H be the resulting graph. More formally, $H = (V', E')$ where:

$$\begin{aligned} V' &= V \cup \{u' | u \in S \subseteq V\} \\ E' &= \{(u, u') | u \in S\} \cup \{(u, v), (u', v) | u \in S, v \in K \text{ and } (u, v) \in E\} \end{aligned}$$

Lemma 3.4 *If $m(H)$ denotes the matching number of H, then $tS(G) = m(H) - |S|$.*

Proof: Let M be a maximum matching of H, so that $|M| = m(H)$. A matching M is said to saturate a vertex v, if v is one of the ends of some edge in M. If for some $u \in S \subseteq V$, M saturates u without saturating u' or saturates u' without saturating u, then clearly the matching M may be modified (without changing its cardinality) so that it includes the edge (u, u'). Thus repeating this process we will obtain a matching M' in which for each pair of vertices $\{u, u'\}$ where $u \in S$, either the edge (u, u') is used by M' or both u, u' are saturated by *different* edges of M'.

By our construction, the neighborhoods of u, u' coincide in H, it therefore follows that $tS(G)$ equals the number of pairs (u, u') such that u and u' are saturated by *different* edges of M' and this can be easily seen to be

$$|M'| - |S| = |M| - |S| = m(H) - |S| \quad \square$$

Theorem 3.10 *The vertex-disjoint K_3-packing problem can be solved in $O(m\sqrt{n})$ time on split graphs.*

Proof: Clearly it suffices to consider connected split graphs. Let $G = (V, E)$ be a connected split graph with $V = S \cup K$ where K is a maximum clique, S is an independent set and $S \cap K = \phi$. Let $|V| = n$ and $|E| = m$. Construct the graph $H = (V', E')$ as mentioned above. Clearly $|V'| = |V| + |S|$ and $|E'| \leq |E| + |S|$. Thus, $|V'| \in O(n)$ and $|E'| \in O(m)$. Hence $m(H)$ and a maximum matching M of H can be determined in $O(m\sqrt{n})$ time [40]. Based on the proof of lemma 3.4, we find that a maximum number of pairwise vertex-disjoint triangles in G may be obtained from the matching M as follows:

1. For each $u \in S$ such that u and u' are both saturated by *different* edges (u, v_1) and (u', v_2) of M where v_1, v_2 are *distinct* vertices in K, add the triangle (u, v_1, v_2) to the triangle-packing.
2. Pack as many triangles, all of whose three vertices will lie in K, using the vertices of K left unused by step 1.

Clearly the process can be done in linear time, if for each vertex $v \in V(H)$, its neighbor if any in M is stored and presented along with M.
Thus the entire process can be implemented in $O(m\sqrt{n})$ time. $\square$

The packing-hardness number

Since the K_4-packing problem is **NP**-complete on split graphs, the packing-hardness number of the class of split graphs is at most 4. From the above results it is easy to deduce that the packing-hardness number of split graphs equals 3 or 4 according as whether the P_3-packing problem on split graphs is **NP**-complete or polynomial time solvable. Determining whether it 3 or 4 remains a challenging *and* interesting open problem.

3.6 Comparability Graphs

Since bipartite graphs form a subclass of comparability graphs, it follows (using corollary 3.1), that the packing-hardness number of comparability graphs is also 3. However, the determination of the complexity status of K_p-packing problem for various values of $p > 2$ is still an interesting problem (recall that the result on bipartite graphs considered P_3-packing). We now proceed to prove that K_4-packing is **NP**-hard on comparability graphs – the idea of the proof is due to Bodlaender and Jansen [8] – though we present it here in a slightly simplified manner.

Theorem 3.11 *It is* **NP***-complete to determine whether a comparability graph has a K_4-factor.*

Proof: The reduction is from the *NUMERICAL 3-DIMENSIONAL MATCHING* (N3DM) problem. An instance of N3DM consists of three disjoint n elements sets: $X = \{x_1, x_2, \ldots, x_n\}$, $Y = \{y_1, y_2, \ldots, y_n\}$ and $Z = \{z_1, z_2, \ldots, z_n\}$, a bound $B \in \mathcal{N}$ and a size $s(a) \in \mathcal{N}$ $\forall$ $a \in X \cup Y \cup Z$ such that $\sum_{a \in X \cup Y \cup Z} s(a) = nB$. The question is whether $X \cup Y \cup Z$ can be partitioned into n disjoint sets A_i such that each A_i has exactly one element from each of X, Y and Z and such that $\sum_{a \in A_i} s(a) = B$ for $1 \leq i \leq n$. WLOG assume that $1 < s(x_i), s(y_i), s(z_i) < B/2$ $\forall i$, $1 \leq i \leq n$ and that $B > 2$.

We first specify a family of (closed) intervals on the real line.

- $A = \{a_i = [0, s(x_i)] : 1 \leq i \leq n\}$
- $B = \{b_{ij} = [s(x_i) + 1, s(x_i) + s(y_j) + jB] : 1 \leq i, j \leq n\}$
- $C = \{c_{jk} = [(j+1)B - s(z_k) + 1, (n+1)B + k] : 1 \leq j, k \leq n\}$
- $D = \{d_k = [(n+1)B + k + 1, (n+3)B + 1] : 1 \leq k \leq n\}$
- $E = \{e_{il} = [1, s(x_i)] : 1 \leq i \leq n, 1 \leq l < n\}$
- $F = \{f_{il} = [0, 0] : 1 \leq i \leq n, 1 \leq l < n\}$
- $G = \{g_{jl} = [(j+1)B, (n+3)B + 1] : 1 \leq j \leq n, 1 \leq l < n\}$
- $H = \{h_{jl} = [0, jB] : 1 \leq j \leq n, 1 \leq l < n\}$
- $I = \{i_{kl} = [(n+1)B + k + 1, (n+3)B] : 1 \leq k \leq n, 1 \leq l < n\}$
- $J = \{j_{kl} = [(n+3)B + 1, (n+3)B + 1] : 1 \leq k \leq n, 1 \leq l < n\}$

Set $V = A \cup B \cup C \cup D \cup E \cup F \cup G \cup H \cup I \cup J$, $|V| = 8n^2 - 4n$. Construct $G = (V, E)$ as follows: $p, q \in V$ are adjacent in G i.e $(p, q) \in E$ if $p \cap q = \phi$ as sets on the real line. Clearly G^c is an interval graph and hence G is a comparability graph.

We proceed to prove that G has a K_4-factor if and only if the instance of N3DM has a solution.

Suppose for instance let $A_i = \{x_{f_i}, y_{g_i}, z_{h_i}\}$, $1 \leq i \leq n$, be a solution to N3DM with $s(x_{f_i}) + s(y_{g_i}) + s(z_{h_i}) = B$ for $1 \leq i \leq n$. For $1 \leq i \leq n$, set $U_i = \{a_{f_i}, b_{f_i g_i}, c_{g_i h_i}, d_{h_i}\}$, since $s(x_{f_i}) + s(y_{g_i}) + s(z_{h_i}) = B$, we have $s(x_{f_i}) + s(y_{g_i}) + g_i B < (g_i + 1)B - s(z_{h_i}) + 1$, so clearly U_i is a K_4. Now let $B' = B - \bigcup_{i=1}^{m} U_i$, if $b_{ij} \in B'$, then put f_{il}, e_{il}, b_{ij} and g_{jl} into a K_4 (note that $(b_{ij}, g_{jl}) \in E$ since $s(x_i) + s(y_j) < B$). This construction is correct because each index i and j appears exactly $n-1$ times in B'. Similarly cover the vertices in $C' = C - \bigcup_{i=1}^{m} U_i$ using K_4's that consist of one vertex each from H, C', I and J. After these steps $2n^2 - n$ vertex-disjoint K_4's that cover all the vertices in V have been chosen. In other words, G has a K_4-factor.

Conversely, let G have a K_4-factor $\mathcal{F}$. First note that the following sets are all independent sets in G: $A \cup F \cup H$, $A \cup E \cup H$, $B \cup H$, $C \cup G$, $D \cup G \cup I$ and $D \cup G \cup J$.

Thus if a K_4 in $\mathcal{F}$ contains a vertex $h \in H$, then it must contain one vertex each from H, C, I and J. Similarly if a K_4 in $\mathcal{F}$ contains a vertex $g \in G$, then it must contain a vertex each from G, B, E and F.

Since $|H| = |G| = n(n-1)$, after removing the vertices in the above $2n(n-1)$ K_4's we will have only $4n$ vertices left – n vertices each from A, B, C and D. Thus the K_4-factor $\mathcal{F}$ must have n K_4's of the form $U_i = \{a_i, b_{ij_i}, c_{j'_i k_i}, d_{k'_i}\}$ for $1 \le i \le n$.

Claim: If $U \in \mathcal{F}$ and $\{c_{jk}, i_{k'l}\} \subset U$ or $\{c_{jk}, d_{k'}\} \subset U$, then we must have $k = k'$.

Proof: If not, let c_{jk} be a vertex with minimum value of k which lies in a K_4 U of $\mathcal{F}$ with $i_{k'l}$ or $d_{k'}$ for some $k' \neq k$. Since U is a clique, by our construction of intervals we must clearly have $k < k'$. But then each c_{jq} with $q < k$ is present in cliques with i_{ql} or d_q only. So there must exist a vertex in $\{i_{kl} : 1 \le l < n\} \cup \{d_k\}$ which is in a K_4 of $\mathcal{F}$ along with $c_{j'k''}$ where $k'' > k$, a contradiction since the corresponding intervals will overlap. □

Similarly we can prove that if $\{b_{ij}, g_{j'l}\} \subset U$ or $\{b_{ij}, c_{j'k}\} \subset U$ for some K_4 U in $\mathcal{F}$, then we must have $j = j'$. Using the above claims, we conclude that U_i, $1 \le i \le n$, is of the form $U_i = \{a_i, b_{i'j_i}, c_{j_i k_i}, d_{k_i}\}$ where $\{j_i : 1 \le i \le n\} = \{k_i : 1 \le i \le n\} = \{1, 2, \ldots, n\}$.

Now since the n^2 vertices of B are covered by the K_4-factor $\mathcal{F}$ through n K_4's P_i which contain a_i and $n(n-1)$ K_4's Q_{il} which contain e_{il}, we can conclude that vertices a_i or e_{il} are together with $b_{i'j}$ only if $s(x_i) = s(x_{i'})$. It is then possible to swap elements between the K_4's P_i and Q_{il} so that each a_i and each e_{il} lies in a K_4 with one b_{ij}. Thus we may assume that, for $1 \le i \le n$, $U_i = \{a_i, b_{ij_i}, c_{j_i k_i}, d_{k_i}\}$. Since U_i is a clique, $(b_{ij_i}, c_{j_i k_i}) \in E$ and hence, $s(x_i) + s(y_{j_i}) + j_i B < (j_i + 1)B - s(z_{k_i}) + 1$. Thus, we have for $1 \le i \le n$,

$$s(x_i) + s(y_{j_i}) + s(z_{k_i}) \le B$$

Since $\{j_i : 1 \le i \le n\} = \{k_i : 1 \le i \le n\} = \{1, 2, \ldots, n\}$, we have,

$$\sum_{i=1}^{n} s(x_i) + s(y_{j_i}) + s(z_{k_i}) = nB$$

and hence we can conclude that $s(x_i) + s(y_{j_i}) + s(z_{k_i}) = B$ for $1 \le i \le n$. Thus, $A_i = \{x_i, y_{j_i}, z_{k_i}\}$, $1 \le i \le n$, is a solution to the instance of N3DM. □

The reduction in the proof of the above theorem actually produces a comparability graph (in fact complement of an interval graph) with clique number at most five. Thus we have:

Corollary 3.2 *The K_4-factor problem is* **NP***-complete on comparability graphs with clique number at most five.*

Corollary 3.3 *The packing-hardness number of the class of interval graphs is either 3 or 4.*

We actually believe that the proof of theorem 3.11 can be modified to settle the following question.
Conjecture: *The K_3-packing problem is* **NP***-complete on comparability graphs with clique-number at most 4.*

3.7 Concluding Remarks

In this chapter, we have proved hardness results for graph packing problems on a wide variety of graphs. Our work marks the first comprehensive attempt to understand generalized matching problems on special classes of graphs. The results presented include proofs of the fact that the packing-hardness number of chordal, planar, line, total and bipartite graphs all equal three.

Several questions remain unanswered, some of the most prominent, interesting and possibly easier ones are:

- What is the packing-hardness number of split graphs? This question is equivalent to settling the complexity status of the P_3-packing problem on split graphs. This problem seems to be pretty interesting and challenging. The approach used in section 3.4 can be modified to give a factor 1/2 approximation to the optimal P_3-packing of a split graph.
- What is the packing-hardness number of unrestricted trees? Note that any fixed tree can be packed in polynomial time into a given tree [38], however the case of arbitrary forests is not so simple.
- Prove or disprove the conjecture in section 3.6.

One other possible line of research is to investigate the complexity of approximate H-packing for a fixed pattern graph H. Any maximal (vertex-disjoint) H-packing trivially achieves an approximation ratio of $1/|V(H)|$. It remains to be seen by how much this naive method can be improved.

Chapter 4

Problems with arbitrarily Small Approximation Thresholds

The area of approximation algorithms for **NP**-hard optimization problems has received a lot of attention over the past two decades. Although some notable positive results have been obtained, it was for long not clear how to prove even approximation problems to be *hard* as was suspected to be the case for many problems. When seen from the perspective of approximability, **NP**-complete problems maintain a healthy, confusing diversity. In order to prove hardness results relating to approximability, the traditional Cook–Karp–Levin type of reduction is grossly inadequate, we require approximation–preserving reductions. One kind of approximation–preserving reduction, called L–Reduction, is described in [41, 42]. We describe it here for the sake of completeness.

Suppose that A and B are optimization problems. An L–Reduction from A to B is a pair of functions R and S, both polynomial-time computable, with the following two properties: First, if x is an instance of A with optimum cost $OPT(x)$, then $R(x)$ is an instance of B with optimum cost that satisfies $OPT(R(x)) \leq \alpha \cdot OPT(x)$ for some constant $\alpha > 0$. Second, if s is any feasible solution of $R(x)$, then $S(s)$ is a feasible solution of x such that

$$|OPT(x) - c(S(s))| \leq \beta \cdot |OPT(R(x)) - c(s)|$$

for some constant $\beta > 0$. That is, S is guaranteed to return a feasible solution of x which is not much more suboptimal than the given solution of $R(x)$. For most L–Reductions (and the ones used in this chapter), we will have $\beta = 1$.

The key property of L–Reductions is that they preserve approximability: if there is an L–Reduction from A to B and B admits a PTAS, then there is a PTAS for A as well.

A complexity class called **MAX SNP**, that was defined in [42], serves as the analog of **NP** with respect to approximation problems: it is a broad class of *optimization* problems, and contains many important complete ones. Though it was defined in a different manner, for our purposes, it is enough to know the equivalent characterization of **MAX SNP** as the class of all optimization problems with approximation threshold strictly less than one, i.e those which admit a factor k approximation algorithm for some constant $k > 0$. A problem is said to be **MAX SNP**-hard if all problems in **MAX SNP** are L–reducible to it, for instance MAX3SAT is known to be **MAX SNP**-complete.

If any **MAX SNP**-hard problem admits a PTAS, then so does every problem in **MAX SNP**, so the question of whether **MAX SNP**-hard problems admit a PTAS is an important one. A sequence of deep and striking recent results, beginning with those in [21], has established the following:

Theorem 4.1 *[3]* **MAX SNP**-*hard problems have polynomial time approximation schemes if and only if* **P**=**NP**.

Connecting the approximability issue with the $\mathbf{P} \stackrel{?}{=} \mathbf{NP}$ question required the development of a very interesting *alternative characterization* of **NP** based on probabilistically checkable proofs which culminated in the now famous PCP theorem that states that **NP** $\subseteq$ PCP$(\log n, 1)$ [3, 4].

The purpose of this chapter is to provide a concrete example of a family of natural graph-theoretic problems which behave better and better with respect to approximability but none of which are likely to admit a polynomial time approximation scheme. The family of problems we give is the independent set problem on k–iterated total graphs, $\forall$ $k \geq 1$.

4.1 Introduction

An optimization problem A is characterized by a set of feasible solutions, call it $F(x)$, for each instance x, and a positive integer cost $c(s)$ for each solution $s \in F(x)$. The optimum cost is defined as $OPT(x) = \min_{s \in F(x)} c(s)$ or $OPT(x) = \max_{s \in F(x)} c(s)$ according as A is a minimization or maximization problem respectively. A heuristic algorithm M, which given any instance x of A returns a feasible solution $M(x) \in F(x)$ that satisfies,

$$|c(M(x)) - OPT(x)| \leq \epsilon \max\{OPT(x), c(M(x))\}$$

for some $\epsilon \geq 0$, is called an ϵ-*approximation algorithm* (this is following the conventions of [41] that treats maximization and minimization problems uniformly – for both kinds of problems ϵ takes on values between 0 and 1 – this notation is used only in this chapter). For a maximization problem, an ϵ-approximation algorithm returns solutions that are never smaller than $(1 - \epsilon)$ times the optimum. For minimization problems, it returns a solutions that are never more than $1/(1 - \epsilon)$ times the optimum. A heuristic is an *asymptotic* ϵ-*approximation algorithm* if there exists a constant $\delta > 0$ such that for all instances x,

$$|c(M(x)) - OPT(x)| \leq \epsilon \max\{OPT(x), c(M(x))\} + \delta$$

The *approximation threshold* of an optimization problem A is the greatest lower bound of all $\epsilon > 0$ such that there is a polynomial-time ϵ-approximation algorithm for A. The *asymptotic approximation threshold* is defined similarly.

An optimization problem A is said to admit a polynomial time approximation scheme (PTAS) if for each fixed $\epsilon > 0$, there is a polynomial time ϵ-approximation algorithm

M_ϵ for A. Clearly, the approximation threshold of a problem equals zero if and only if it admits a PTAS.

For convenience sake, we sometimes use the following terminology: A heuristic M is said to be a factor k-approximate algorithm if $c(M(x)) \leq kOPT(x)$ if A is a minimization problem and if $c(M(x)) \geq kOPT(x)$ if A is a maximization problem.

In this chapter, we provide examples of "natural" graph-theoretic problems with arbitrarily small but yet positive approximation thresholds (assuming $\mathbf{P} \neq \mathbf{NP}$). That is, given any $\delta > 0$, we give concrete examples of optimization problems A_δ that have approximation thresholds that is at most δ, however, unless $\mathbf{P} = \mathbf{NP}$, A_δ does not admit a polynomial time approximation scheme.

There do not seem to exist many such natural families $\{A_\delta : 0 < \delta < 1\}$ of optimization problems. However, one of them is the family $\{Q_k : k \geq 2\}$ where Q_k is the version of MAXSAT where each clause has at least k *distinct* literals. It is known that the approximation threshold of Q_k, say δ_k, satisfies $0 < \delta_k \leq \frac{1}{2^k}$ unless $\mathbf{P} = \mathbf{NP}$ [41].

One can formulate many more such problems by allowing problems where one specifies the "cost" of the optimum solution in the problem input itself and the problem is to find an optimum solution with that cost. For example, consider the Minimum Degree Spanning Tree (MDST) problem where one looks for a spanning tree with lowest maximum degree. This problem is $\mathbf{NP}$-hard through a trivial reduction from the Hamiltonian path problem – it also remains hard if we are given the value of the optimum solution (in this case the maximum degree of a spanning tree with smallest maximum degree) – this is because finding a Hamiltonian path is $\mathbf{NP}$-hard even if the input graph is known (or given) to be Hamiltonian. Now, consider the family of problems $\mathcal{Q}_k$, $k \geq 2$ where:

Problem $\mathcal{Q}_k$: Given a graph $G = (V, E)$ such that the minimum degree spanning tree of G has maximum degree *exactly* k, find a minimum degree spanning tree of G.

The above problems are all $\mathbf{NP}$-hard; however for problem $\mathcal{Q}_k$, a spanning tree of G with maximum degree at most $k+1$ can be found in polynomial time using the algorithm of [22]. Clearly this is the best that can be achieved in polynomial time unless $\mathbf{P}$=$\mathbf{NP}$. Thus it follows that the approximation threshold of $\mathcal{Q}_k$ is *exactly* $1/(k+1)$ (assuming $\mathbf{P} \neq \mathbf{NP}$). Thus, the family $\mathcal{Q}_k$ is also of the desired kind. However, note that the problems $\mathcal{Q}_k$ are not very natural ones since we are forced to mention the value of the optimum solution in the problem input itself. The family of problems that we give here are much more natural *and* concrete ones.

4.2 Definitions

Let $G = (V, E)$ be a graph with $|V| = n$ and $|E| = m$. An edge $e = (u, v)$ is said to dominate all edges that have v or u as an end-vertex, including e itself. A set of edges M is said to be independent (a matching) if no two of its edges have a vertex in common.

A set of edges D is said to be an edge dominating set ($ED(G)$) of G if D dominates all edges of G. The **edge domination problem** consists of finding an edge dominating set of minimum cardinality. Note that an independent edge dominating set $IED(G)$ is just a maximal matching. It is known that the size of a minimum independent edge dominating set $MIED(G)$ of G is equal to the size of a minimum edge dominating set $MED(G)$ of G [44]. We denote by $\gamma'(G)$ the size of a $MED(G)$ i.e $\gamma'(G) = |MED(G)| = |MIED(G)|$. Note that $\gamma'(G)$ stands for the size of a minimum maximal matching of G. The edge domination problem is in general a very hard combinatorial problem; it remains **NP**-hard even when restricted to planar bipartite graphs [32].

An edge $e = (u, v)$ *saturates* vertices u and v. Given a set of edges D, we denote by V_D the set of vertices saturated by the edges of D. Recall that $T(G)$ stands for the total graph of G. A vertex of $T(G)$ corresponding to edge (u, v) of G is called an e-vertex and is denoted by $[u, v]$. For $k \geq 0$, the k-iterated total graph of G, denoted $T_k(G)$ is defined as follows:

$$T_k(G) = \begin{cases} G & \text{if } k = 0 \\ T(T_{k-1}(G)) & \text{otherwise} \end{cases}$$

In this chapter, we consider the family P_k of problems defined by:

Problem P_k : Given a graph $G = (V, E)$ and an integer $k \geq 1$, determine a maximum independent set in the k-iterated total graph $T_k(G)$ of G.

We prove the following theorem:

Theorem 4.2 *Given any $\delta > 0$, one can find a constant k (that depends on δ) such that the approximation threshold ϵ_k of P_k satisfies $0 < \epsilon_k \leq \delta$ (assuming* **P** $\neq$ **NP***).*

Thus, the family P_k of problems provides a concrete example of problems with arbitrarily small, yet positive, approximation thresholds.

4.3 Preliminaries

In this section, we state a few preliminary results that will be useful in subsequent sections.

Lemma 4.1 *[44] If M is a maximal matching of $G = (V, E)$, then $M \cup (V - V_M)$ is a maximal independent set of size $|V| - |M|$ in $T(G)$.*

Lemma 4.2 *[44] If I is a maximal independent set of $T(G)$ and M is any maximal matching of G containing all the e-vertices of I, then $|M| = |V(G)| - |I|$. Moreover, if I is a* maximum *independent set of $T(G)$, then M is a $MIED(G)$ and M, together with vertices of G not saturated by M, form a* maximum *independent set of $T(G)$.*

Lemma 4.3 *If M is a $MIED(G)$, then $M \cup (V - V_M)$ is a maximum independent set of $T(G)$. Thus $\alpha(T(G)) = |V(G)| - \gamma'(G)$.*

The above lemmas associate edge dominating sets in G with independent sets in $T(G)$. However a direct relationship exists between these parameters in $T(G)$ alone.

Theorem 4.3 *[32] If G is a connected graph and I is any independent set of $T(G)$, then either $H = T(G) - I$ has a perfect matching, or for any vertex v of I, $T(G) - (I - \{v\})$ has a perfect matching.*

From the above theorem and the fact that a set M of edges of a graph G is an edge dominating set if and only if the vertices not saturated by M form an independent set in G, we can conclude the following:

Lemma 4.4 *Let $T(G)$ be the total graph of a connected graph G having n vertices and m edges, and let D be a $MED(T(G))$. Then:*

- *The vertices not saturated by D form an independent set of size $\alpha(T(G))$ or $\alpha(T(G)) - 1$.*
- $\gamma'(T(G)) = |D| = \lceil \frac{n+m-\alpha(T(G))}{2} \rceil$

4.4 Hardness Results

We have already remarked that the edge domination problem remains **NP**-hard even on highly structured inputs; we now prove that the edge domination problem is **MAX SNP**-hard even when restricted to bounded-degree graphs.

Lemma 4.5 DEGREE-3 VERTEX COVER, *which stands for the vertex-cover problem restricted to graphs G with $\Delta(G) \leq 3$, is* **MAX SNP**-*hard.*

Proof : We exhibit a L-reduction from *DEGREE-4 VERTEX COVER* which is known to be **MAX SNP**-hard [41]. Let $H = (V, E)$ be a connected graph with maximum degree $\Delta(H) \leq 4$ with $V = \{v_1, v_2, \ldots, v_n\}$ and $|E| \geq n$. Construct $G = (V', E')$ as follows:

Replace each $v_i \in V$ by a path $(v_{i1}, w_i, v_{i2})$of length two. If $d_H(v_i) < 3$, then all edges of H incident upon v_i are incident upon v_{i1} in G. If $3 \leq d_H(v_i) \leq 4$, then two of the edges of H incident upon v_i are incident upon v_{i1} in G and the remaining $d_H(v_i) - 2$ are incident upon v_{i2}. Clearly $\Delta(G) \leq 3$, also it is easy to see that a minimum vertex-cover of G may be assumed to include either $\{v_{i1}, v_{i2}\}$ or $\{w_i\}$ from the set $\{v_{i1}, v_{i2}, v_{i3}\}$ for each i. Thus it follows that:

$$\gamma(G) = n + \gamma(H).$$

Since $\Delta(H) \leq 4$ and $|E(H)| \geq n$, we have $\gamma(H) \geq n/4$ and so $\gamma(G) \leq 5\gamma(H)$, and the reduction is an L–Reduction. Hence *DEGREE-3 VERTEX COVER* is **MAX SNP**-hard. □

Lemma 4.6 *The edge domination problem is* **MAX SNP***-hard even on graphs with maximum degree* $\Delta \leq 3$.

Proof : Follows from the fact that the reduction of theorem 1 in [44] is a L–Reduction that will work even if we start with a graph G with $\Delta(G) \leq 3$. □

The above lemma together with theorem 4.1 implies:

Theorem 4.4 *Unless* $\mathbf{P} = \mathbf{NP}$*, the edge domination problem, even when restricted to graphs with maximum degree at most three, does not admit a PTAS i.e has an approximation threshold strictly greater than 0.*

We now proceed to give a few approximation–preserving reductions to prove the impossibility (assuming $\mathbf{P} \neq \mathbf{NP}$) of the existence of a PTAS for the independent set problem on k-iterated total graphs, for any $k \geq 1$. The basic idea behind the proof is as follows. Let G be a graph with $\Delta(G) \leq 3$. Lemma 4.7 proves that the existence of a PTAS for independent set problem on $T_i(G)$ implies the existence of a PTAS for edge domination on $T_{i-1}(G)$, for $i \geq 1$. Lemma 4.8 proves that, for $i \geq 1$, the existence of a PTAS for edge domination on $T_i(G)$ implies the existence of a PTAS for independent set problem on $T_i(G)$. Now consider the independent set problem on $T_k(G)$, for some $k \geq 1$. Employing lemma 4.7 once, followed by $k-1$ applications of lemma 4.8 interleaved with $k-1$ applications of lemma 4.7, proves that if the independent set problem on $T_k(G)$ has a PTAS, then so does the edge domination problem on $T_0(G) = G$, which by theorem 4.4 is impossible unless $\mathbf{P}{=}\mathbf{NP}$. We therefore conclude:

Theorem 4.5 *For any fixed* $k \geq 1$*, there does not exist a PTAS for the independent set problem on* k*-iterated total graphs unless* $\mathbf{P} = \mathbf{NP}$.

We now proceed to prove lemmas 4.7 and 4.8. From now on in this chapter, let $G = (V_0, E_0)$ stand for any connected graph that is not a tree with $|V_0| = N_0$ and $|E_0| = M_0$ and $\Delta_0 = \Delta(G) \leq 3$. Also let $\alpha(G) = \alpha_0$ and $\gamma'(G) = \gamma'_0$. Consider the sequence of iterated total graphs: $T_i(G) = (V_i, E_i)$ where $|V_i| = N_i$ and $|E_i| = M_i$. Also let $\Delta_i = \Delta(T_i(G))$, $\delta_i = \delta(T_i(G))$, $\alpha_i = \alpha(T_i(G))$ and $\gamma'_i = \gamma'(T_i(G))$. Note that $\delta_i = 2\delta_{i-1}$ and $\Delta_i = 2\Delta_{i-1}$ for $i \geq 1$. Since $\Delta_0 \leq 3$, we have $\Delta_i \leq 3 \times 2^i$.

Lemma 4.7 *If there exists a PTAS for the independent set problem on* $T_i(G)$*,* $i \geq 1$*, then there exists a PTAS for the edge domination problem on* $T_{i-1}(G)$ *as well, where* G *stands for any generic connected graph (that is not a tree) with* $\Delta(G) \leq 3$.

Proof : We give an approximation–preserving L–Reduction from the edge domination problem (say A_1) on $T_{i-1}(G)$ to the independent set problem (say A_2) on $T_i(G)$. Given an instance $H = T_{i-1}(G)$ of A_1, we construct the instance $T(H) = T_i(G)$ of A_2. Let I be the independent set returned by a proposed heuristic for A_2. WLOG assume that I is a maximal independent set. One can construct an edge dominating set for the instance H

of A_1 as follows: Set $f(I)$ to be any maximal matching of H containing all e-vertices of I, by lemma 4.2, $f(I)$ is a $IED(H)$ of size $|V(H)| - |I|$, so

$$\begin{aligned} |\gamma'(H) - f(I)| &= ||V(H)| - \alpha(T(H)) - (|V(H)| - |I|)| \\ &= |\alpha(T(H)) - |I|| \end{aligned} \tag{4.1}$$

Also, $\alpha(T(H)) = |V(H)| - \gamma'(H) = N_{i-1} - \gamma'(H)$. But, clearly the minimum vertex-cover of H has size at most $2\gamma'(H)$, and hence:

$$\begin{aligned} \gamma'(H) &\geq \frac{M_{i-1}}{2\Delta_{i-1}} \\ &\geq \frac{N_{i-1}}{3 \times 2^i} \text{ (since } M_{i-1} \geq N_{i-1} \text{ and } \Delta_{i-1} \leq 3 \times 2^{i-1} \text{).} \end{aligned}$$

Hence,

$$\alpha(T(H)) \leq (3 \times 2^i - 1)\gamma'(H) \tag{4.2}$$

From equations (4.1) and (4.2), we find that the above reduction is indeed a L–Reduction. Since L–Reductions preserve approximability [41], the result follows. □

Lemma 4.8 *If there exists a PTAS for the edge domination problem on $T_i(G)$ $(i \geq 1)$, where as before G stands for any generic connected graph with $\Delta(G) \leq 3$, then there exists a PTAS for the independent set problem on $T_i(G)$ as well.*

Proof : Let $H = T_i(G)$, where $\Delta(G) \leq 3$, be a graph whose $\alpha(H)$ is desired. Suppose a PTAS $\mathcal{P}$ exists for the edge domination problem on $T_i(G)$. Let $\epsilon > 0$ be given. First, use the PTAS $\mathcal{P}$ to obtain a factor $(1 + \epsilon)$ approximation to the minimum independent dominating set of H, i.e find a minimum maximal matching M of H such that:

$$\frac{|M|}{\gamma'(H)} \leq 1 + \epsilon \tag{4.3}$$

Now,

$$\begin{aligned} |V(H)| &= |V(T_{i-1}(G))| + |E(T_{i-1}(G))| \\ &\leq |V(T_{i-1}(G))|(1 + \frac{\Delta(T_{i-1}(G))}{2}) \\ &\leq (1 + 3 \times 2^{i-2})|V(T_{i-1}(G))| \end{aligned} \tag{4.4}$$

By lemma 4.3,

$$\begin{aligned} \alpha(H) &= |V(T_{i-1}(G))| - \gamma'(T_{i-1}(G)) \\ &\geq \frac{|V(T_{i-1}(G))|}{2} \\ &\geq \frac{|V(H)|}{2 + 3 \times 2^{i-1}} \text{ (using (4.4))} \end{aligned} \tag{4.5}$$

Also, by lemma 4.4,

$$\begin{aligned}\gamma'(H) &= \lceil \frac{|V(H)| - \alpha(H)}{2} \rceil \\ &\leq \lceil \frac{3 \times 2^{i-1} + 1}{2} \alpha(H) \rceil \\ &\leq (1 + 3 \times 2^{i-2})\alpha(H) \text{ since } \alpha(H) \geq 1 \qquad (4.6)\end{aligned}$$

Having obtained a $IED(H)$ satisfying (4.3), output $I = V(H) - V_M$ as an approximation to the maximum independent set of H, clearly $|I| = |V(H)| - 2|M|$.
Since H is a total graph, by lemma 4.4,

$$\alpha(H) \leq |V(H)| - 2\gamma'(H) + 1$$

So,

$$\begin{aligned}|\alpha(H) - |I|| &\leq ||V(H)| - 2\gamma'(H) + 1 - (|V(H)| - 2|M|)| \\ &\leq 2\gamma'(H)\epsilon + 1 \text{ (using (4.3))} \\ &\leq (2 + 3 \times 2^{i-1})\epsilon\alpha(H) + 1 \text{ (using (4.6))}\end{aligned}$$

Thus, using the factor $(1+\epsilon)$ approximation algorithm for edge domination on $T_i(G)$, $(i \geq 1)$, we can get a $(2+3\times 2^{i-1})\epsilon$–asymptotic approximation algorithm for the independent set problem on $T_i(G)$. Since $\epsilon > 0$ was arbitrary, we conclude that the asymptotic approximation threshold of the independent set problem on $T_i(G)$ equals 0. Since all constant restrictions of this problem (i.e given a *constant* K, determine if $\alpha(T_i(G)) \geq K$) are polynomial-time solvable, using lemma 4.9 we conclude that the approximation threshold of this problem is also zero, i.e this problem has a PTAS. □

Lemma 4.9 *[41, Exercise 13.4.4 (b)] If all constant restrictions of a problem are polynomial-time solvable, then the asymptotic approximation threshold equals the actual approximation threshold.*

Proof : For definiteness, let us consider a maximization problem A. For any given $\beta \geq 0$ which is greater than the asymptotic approximation threshold $aat(A)$ of A, let H_β be a heuristic that satisfies, for any instance x of A,

$$|OPT(x) - c(H_\beta(x))| \leq \beta OPT(x) + \delta$$

where $\delta \geq 0$ is a constant (possibly dependent on β).

For any given $\epsilon > aat(A)$, we can get a ϵ-approximate algorithm J_ϵ for A as follows: Firstly, pick a ϵ' that satisfies $aat(A) < \epsilon' < \epsilon$. Then, given an instance x of A, J_ϵ first checks in polynomial time whether $OPT(x) \geq \frac{\delta}{\epsilon - \epsilon'}$, if so it uses the heuristic $H_{\epsilon'}$ to get a solution $H_{\epsilon'}(x)$ such that:

$$\begin{aligned}c(H_{\epsilon'}(x)) &\geq (1 - \epsilon')OPT(x) - \delta \\ &\geq (1 - \epsilon)OPT(x) \text{ (since } OPT(x) \geq \tfrac{\delta}{\epsilon - \epsilon'})\end{aligned}$$

If not, i.e $OPT(x) < \delta/(\epsilon - \epsilon')$ then J_ϵ returns the exact value of $OPT(x)$ by applying at most $\lceil \delta/(\epsilon - \epsilon') \rceil$ polynomial-time constant restriction algorithms. □

4.5 Approximation Guarantees

In this section, we give approximation algorithms with guaranteed worst-case performance bounds for the independent set problem on k-iterated total graphs (for $k \geq 1$) and the edge domination problem on k-iterated total graphs (for $k \geq 0$).

Lemma 4.10 *Let M and M' be any two maximal matchings of a graph $G = (V, E)$. Then $|M'| \leq 2|M|$.*

Proof : Since M is a maximal matching, $V - V_M$ is an independent set; so any edge in M' must have at least one end in V_M. Hence $|M'| \leq |V_M| = 2|M|$. □

Lemma 4.11 *A minimum edge dominating set of a graph G can be approximated to a factor of two in polynomial time.*

Proof : Given $G = (V, E)$, output (in linear time) any maximal matching M of G, clearly M is an edge dominating set. Since $\gamma'(G) = |MED(G)| = |MIED(G)|$, using lemma 4.10 we have $|M| \leq 2\gamma'(G)$ and the result follows. □

We shall henceforth employ the notation of section 4.4 on the sequence of i-iterated total graphs $\{G_i\}_{i \geq 0}$ where $G_{i+1} = T(G_i)$ for $i \geq 0$.

Suppose that there exist polynomial time approximation algorithms E_j (resp. S_j) to approximate edge dominating sets (resp. independent sets) on j-iterated total graphs to a factor of k_j (resp. c_j) for all $j < i$. Note that $c_j \leq 1 \leq k_j \; \forall \; j$. Initially, $i = 1$, $k_0 = 2$ and $c_0 = 0$. We now prove that there exist polynomial time approximation algorithms E_i and S_i that guarantee approximation factors of k_i and c_i (to be defined later) respectively.

Algorithm S_i : $i \geq 1$
Input : A connected graph $G = (V, E)$.
Output : An independent set I_i in $T_i(G)$.

Steps: Run the algorithm E_{i-1} to get a $IED(T_{i-1}(G))$, say M_{i-1}. Output $V(T_i(G)) - V_{M_{i-1}}$ as the independent set I_i.

Performance Guarantee: We have $|M_{i-1}| \leq \min\{k_{i-1}\gamma'_{i-1}, N_{i-1}/2\}$. Also, by lemma 4.3, $\alpha_i = N_{i-1} - \gamma'_{i-1}$. Hence,

$$\frac{|I_i|}{\alpha_i} \geq \frac{N_{i-1} - \min\{k_{i-1}\gamma'_{i-1}, \frac{N_{i-1}}{2}\}}{N_{i-1} - \gamma'_{i-1}}$$

If $\gamma'_{i-1} \geq N_{i-1}/2k_{i-1}$, then

$$\begin{aligned}\frac{|I_i|}{\alpha_i} &\geq \frac{1}{2(1-\frac{\gamma'_{i-1}}{N_{i-1}})} \\ &\geq \frac{k_{i-1}}{2k_{i-1}-1}\end{aligned} \tag{4.7}$$

On the other hand, if $\gamma'_{i-1} < N_{i-1}/2k_{i-1}$, then

$$\begin{aligned}\frac{|I_i|}{\alpha_i} &\geq \frac{N_{i-1}-k_{i-1}\gamma'_{i-1}}{N_{i-1}-\gamma'_{i-1}} = k_{i-1} - \frac{k_{i-1}-1}{1-\frac{\gamma'_{i-1}}{N_{i-1}}} \\ &\geq k_{i-1} - 2k_{i-1}\frac{(k_{i-1}-1)}{2k_{i-1}-1} = \frac{k_{i-1}}{2k_{i-1}-1}\end{aligned} \tag{4.8}$$

From (4.7) and (4.8) we conclude that the proposed algorithm has for performance guarantee a factor c_i given by:

$$c_i = \frac{k_{i-1}}{2k_{i-1}-1} \tag{4.9}$$

Algorithm F_i ($i \geq 1$) (this algorithm will be used to prove the existence of algorithm E_i):
Input : A connected graph $G = (V, E)$
Output : A maximal matching M'_i of $T_i(G)$.
Steps: Run the algorithm S_i to get an independent set I_i of $T_i(G)$. Find a maximal matching R_i of $T_i(G) - I_i$. Then, find a maximal matching M'_i of $T_i(G)$ containing R_i and output M'_i.

By theorem 4.3 either $V_{R_i} = V(T_i(G)) - I_i$ or $V_{R_i} = V(T_i(G)) - (I_i - v)$ for some $v \in I_i$. Hence $|R_i| \leq |M'_i| \leq |R_i| + 1$, in fact by lemma 4.4, $|M'_i| = \lceil (N_i - |I_i|)/2 \rceil$ where $N_i = |V(T_i(G))|$.

Performance Guarantee: By the performance guarantee of S_i, we have $|I_i| \geq c_i\alpha_i$. Hence,

$$|M'_i| \leq \lceil \frac{N_i - |I_i|}{2} \rceil \leq \frac{N_i - |I_i| + 1}{2} \leq \frac{N_i - c_i\alpha_i + 1}{2} \tag{4.10}$$

On the other hand,

$$\gamma'_i = \lceil \frac{N_i - \alpha_i}{2} \rceil \geq \frac{N_i - \alpha_i}{2} \tag{4.11}$$

Combining (4.10) and (4.11) we have,

$$\begin{aligned}\frac{|M'_i|}{\gamma'_i} &\leq \frac{N_i - c_i\alpha_i + 1}{2\gamma'_i} \\ &\leq c_i + \frac{(1-c_i)N_i}{N_i - \alpha_i} + \frac{1}{2\gamma'_i} \\ &\leq c_i + \frac{(1-c_i)N_i}{N_i - N_{i-1}} + \frac{1}{2\gamma'_i} \text{ (since } \alpha_i = N_{i-1} - \gamma'_{i-1} \leq N_{i-1}\text{)}\end{aligned} \tag{4.12}$$

If G is a tree, then α_i, γ_i' etc can be obtained exactly in polynomial time (since i is assumed to be a fixed constant). So let us assume that $M_0 \geq N_0$. Then $N_1 \geq 2N_0$, $\delta_i = 2\delta_{i-1} \geq 2^i$ for $i \geq 1$ and for $i \geq 2$, $N_i = N_{i-1} + M_{i-1} \geq N_{i-1}(1 + \delta_{i-1}/2) \geq (1 + 2^{i-2})N_{i-1}$. Thus, for $i \geq 1$,

$$\frac{N_{i-1}}{N_i} \leq \min\{\frac{1}{2}, \frac{1}{1+2^{i-2}}\} \tag{4.13}$$

Combining (4.12) and (4.13), we get:

$$\frac{|M_1'|}{\gamma_1'} \leq 2 - c_1 + \frac{1}{2\gamma_1'}$$

and for $i \geq 2$,

$$\frac{|M_i'|}{\gamma_i'} \leq \frac{1+2^{i-2}}{2^{i-2}} - \frac{c_i}{2^{i-2}} + \frac{1}{2\gamma_i'}$$

Hence the asymptotic approximation threshold guaranteed by the algorithm F_i is at most ϵ_i, where:

$$\epsilon_i = \begin{cases} 1 - \frac{1}{2-c_i} & \text{if } i = 1 \\ 1 - \frac{2^{i-2}}{1+2^{i-2}-c_i} & \text{if } i \geq 1 \end{cases}$$

Since all constant restrictions of the edge domination problem on i-iterated total graphs (call it Q_i, $i \geq 0$) are polynomial-time solvable, we can invoke lemma 4.9 to conclude that the approximation threshold of Q_i is also at most ϵ_i (set $\epsilon_0 = 1/2$), i.e $\exists$ an algorithm E_i for Q_i that guarantees an approximation factor of:

$$k_i = \frac{1}{1-\epsilon_i} = \begin{cases} 2 & \text{if } i = 0 \\ 2 - c_1 & \text{if } i = 1 \\ 1 + \frac{1-c_i}{2^{i-2}} & \text{if } i \geq 2 \end{cases}$$

Using the above in conjunction with (4.9) we get: $c_1 = 2/3$, $c_2 = 4/5$ and for $i \geq 2$,

$$c_{i+1} = \frac{k_i}{2k_i - 1} = 1 - \frac{1-c_i}{2^{i-2} + 2(1-c_i)}$$

Defining $w_i = 1/(1-c_i)$ we get: $w_1 = 3$, $w_2 = 5$ and for $i \geq 2$,

$$w_{i+1} = 2 + 2^{i-2}w_i$$

Clearly the approximation threshold β_k of the independent set problem P_k on k-iterated total graphs, $k \geq 1$, satisfies $\beta_k \leq 1 - c_k = 1/w_k$. Clearly $\lim_{k\to\infty} w_k = \infty$ and $w_i < w_{i+1}\ \forall\ i \geq 1$. It therefore follows that:

Theorem 4.6 *The sequence $\{\beta_i\}_{i\geq 1}$ of approximation thresholds is a non-increasing sequence converging to zero.*

Using theorem 4.6 along with theorem 4.5 yields theorem 4.2. Thus we have exhibited a family of problems with arbitrarily small, yet positive (assuming $\mathbf{P} \neq \mathbf{NP}$), approximation thresholds.

4.6 Conclusions

The interesting thing about the problems P_k is that if a polynomial time approximation algorithm for P_k with guarantee better that c_k is discovered for any $k \geq 1$, then there exists a polynomial time approximation algorithm that achieves a factor better than 2 for the edge domination or minimum maximal matching problem on general graphs. It is an interesting question whether such an approximation algorithm exists for the edge domination problem.

Chapter 5

Approximability of Domination Problems

Domination problems are among the most well–studied and useful graph–theoretic problems. For a graph $G = (V, E)$, a vertex subset $X \subseteq V$ is a *dominating set* if every vertex in $V - X$ is adjacent to at least one vertex in X. Domination problems have several variants: X is said to be a *total* dominating set if every vertex in V is adjacent to at least one vertex in X; X is a *connected* (*clique*) dominating set if in addition to being a dominating set it induces a connected subgraph of G (clique in G). Also, X is an *independent* dominating set if it is a dominating set as well as an independent set, or, in other words, if it is a *maximal* independent set.

In general all these variants of domination are **NP**-hard [24], moreover several of them remain **NP**-hard even for restricted families of graphs, see [17] for an excellent survey for the status of domination problems on special classes of graphs. The purpose of this chapter is to investigate the approximability of these variants of domination on general graphs.

5.1 Domination, Total and Connected Domination

We now note that all variants other than independent domination are hard to approximate to a factor of $(1 - \epsilon) \ln N$, for any constant $\epsilon > 0$, where N is the number of vertices in the graph. To prove this, we employ an approximation preserving reduction from the set cover problem to these domination problems and use a sequence of recent inapproximability results [6, 20, 36] that state that set cover cannot be approximated in polynomial time to a factor of $(1 - \epsilon) \ln n$, where n is the number of elements in the ground set, unless $\mathbf{NP} \subseteq \mathrm{DTIME}(n^{O(\log \log n)})$, even for instances where the number of sets in the family is bounded by a polynomial in n (in fact bounded by $O(n \ln n)$, since the reduction presented in [20] results in such instances).

The approximation preserving reduction

Let $\mathcal{I} = (S, \mathcal{F})$ be an instance of SET-COVER where $S = \{x_1, \ldots, x_n\}$ is an n-element set and $\mathcal{F} = \{S_1, S_2, \ldots, S_m\}$ is a family of subsets of S such that $m = O(n \ln n)$. Construct a graph $G = G_{\mathcal{I}}$ from $\mathcal{I}$ as follows. The vertex set of G is $S \cup \mathcal{F}$, $\mathcal{F}$ induces a clique in G and S induces an independent set in G, implying that G is in fact a *split* graph. The edges in G with one end in S and the other in $\mathcal{F}$ are given by

$$E_{S\mathcal{F}} = \{(x_i, S_j) : x_i \in S_j\}.$$

Clearly, any dominating set of G other than an independent dominating set, can confine itself to choice of vertices in $\mathcal{F}$. Since $\mathcal{F}$ induces a clique, the problems of clique, connected and total domination have the same complexity as domination on the graph G.

Now, $\mathcal{E} \subseteq \mathcal{F}$ will be a dominating set of G if and only if it is a set cover for the instance $\mathcal{I}$. Also, the number of vertices N in G is given by $N = n + m = O(n \ln n)$. Hence it is easy to see that if there exists an $\epsilon > 0$ such that any of these domination problems can be approximated to a factor of $(1 - \epsilon) \ln N$ in polynomial time, then set cover can be approximated in polynomial time to a factor of $(1 - \delta) \ln n$, for some $\delta > 0$ that depends on ϵ, for all sufficiently large n. Using the results of [20, 36], we can now state

Theorem 5.1 *Domination, total domination, connected domination and clique domination cannot be approximated to a factor of* $(1 - \epsilon) \ln N$ *in polynomial time, for any constant* $\epsilon > 0$*, unless* **NP** $\subseteq$ *DTIME(*$N^{O(\log \log N)}$*).*

We now state the positive results on what can be achieved in polynomial time. A straightforward set cover based approach can be used to approximate domination and total domination to a factor of $\ln \Delta + O(1)$. Guha and Khuller [26] present an algorithm to approximate connected dominating sets to a factor of $\ln \Delta + O(1)$. Thus, the approximability of these three variants of domination has been established to be $\Theta(\ln N)$. Regarding the other two variants, the next two sections indicate that these are much harder to approximate.

5.2 Independent Domination

For a graph G, let $MID(G)$ denote any minimum independent dominating set (or equivalently, minimum maximal independent set) and let $\alpha_d(G) = |MID(G)|$. The independent domination problem is **NP**-hard [24, 33], but it appears very difficult to approximate owing to its non–monotonicity, i.e no proper subset or superset of an independent dominating set is an independent dominating set. In fact, no method to approximate $\alpha_d(G)$ to within a factor other than the trivial factor $O(n)$ appears to be known. In this section, we provide strong evidence supporting this difficulty of approximating $\alpha_d(G)$.

It is first proved, using a property called *self–improvement* associated with independent domination, that $\alpha_d(G)$ cannot be approximated to a constant factor unless **P**=**NP** and to a factor of $2^{O(\log^{1-\epsilon} n)}$, for any $\epsilon > 0$, unless **NP** has a quasi–polynomial (i.e $2^{\mathrm{poly}(\log n)}$) deterministic time simulation (which is considered very unlikely [34]). These results are established using techniques that have found use in several other problems as well.

We then prove, using problem–specific techniques, that approximation to within a factor of even $n/\log^{1+\epsilon} n$, for any $\epsilon > 0$, seems unlikely (the complexity–theoretic assumption made being that no **NP**-complete problem can be decided in $2^{O(n^\delta)}$ deterministic time for any $\delta < 1$). This represents the **strongest known hardness result** for approximating an unweighted **NP**-complete problem. All these results involve the use of the recent techniques on probabilistically checkable proofs and their applications to proving the hardness of approximating MAX-SAT and other **MAX SNP**-hard problems [3, 6]. It is also proved, without resorting to PCP techniques, that approximation to a factor of $n^{1-\epsilon}$ is **NP**-hard for any $\epsilon > 0$. Our proof technique is a simple adaptation of the one presented in [33] where **NP**-hardness of approximating $\alpha_d(G)$ to any constant factor is proved.

5.2.1 Hardness Results Based on Self–improvement

We now prove a self–improvement behavior for independent domination when we take graph *products*. Such a behavior has also found use in proving the inapproximability of clique [3], coloring [23] and MAX SATISFY [2, 5].

Definition 5.1 *Let $G_i = (V_i, E_i)$, $i = 1, 2$ be two graphs. Their* product $G = G_1 \times G_2$ *is the graph whose vertex set is $V_1 \times V_2$, and edge set is*

$$\{((u_1, v_1), (u_2, v_2)) : u_1 = u_2 \text{ and } (v_1, v_2) \in E_2 \text{ or } (u_1, u_2) \in E_1\}.$$

Lemma 5.1 $\alpha_d(G_1 \times G_2) = \alpha_d(G_1) \times \alpha_d(G_2)$.

Proof: If I_i is an independent dominating set of G_i, $i = 1, 2$, then it is easy to check that $I_1 \times I_2$ is an independent dominating set of $G_1 \times G_2$, so that $\alpha_d(G_1 \times G_2) \leq \alpha_d(G_1) \times \alpha_d(G_2)$.

To prove the other inequality, let I be an independent dominating set of $G_1 \times G_2$. Define $I_1 = \{u \in V_1 : (u, v) \in I \text{ for some } v \in V_2\}$. I_1 is an independent dominating set of G_1 since I is an independent dominating set of $G_1 \times G_2$. For $u \in I_1$, let $V_u = \{(u, v) : v \in V_2\}$. Since I_1 is an independent set in G_1, in order for I to dominate all vertices in V_u, $I \cap V_u$ must be an independent dominating set of G_2. Thus, $|I \cap V_u| \geq \alpha_d(G_2)$ $\forall u \in I_1$. Since $|I_1| \geq \alpha_d(G_1)$, we have $|I| \geq \alpha_d(G_1) \times \alpha_d(G_2)$. $\square$

For $k \geq 1$, define the *power* graph G^k of G as follows:

$$\begin{aligned} G^k &= G \text{ for } k = 1 \\ &= G \times G^{k-1} \text{ for } k \geq 1 \end{aligned}$$

Corollary 5.1 $\alpha_d(G^k) = \alpha_d(G)^k$ *for all $k \geq 1$.*

Remark: Note that lemma 5.1 also implies the existence of an efficient method to recover an independent dominating set of size at most $l^{1/k}$ in G from one of size l in G^k.

Theorem 5.2 *If independent domination has a polynomial time algorithm that achieves a factor p approximation, for some* constant $p > 1$, *then it in fact admits a PTAS.*

Proof: Let A_p be a polynomial time approximation algorithm with performance ratio of $p > 1$. Let $\epsilon > 0$ be any real. Let k be an integer exceeding

$$\frac{\log p}{\log(1+\epsilon)}.$$

Given an input graph, suppose we run algorithm A_p on the power graph G^k, this yields an independent dominating set of size at most $p\alpha_d(G^k) = p\alpha_d(G)^k$ in G^k. By the remark following corollary 5.1, we can compute, in polynomial time, an independent set of size at most $p^{1/k}\alpha_d(G)$ in G. Since $(1+\epsilon)^k \geq p$ by our choice of k, we have $p^{1/k} \leq 1+\epsilon$. This implies a polynomial time factor $(1+\epsilon)$ approximation algorithm for independent domination. Since $\epsilon > 0$ was arbitrary, the result follows. □

Theorem 5.3 *The independent dominating set problem is* **MAX SNP**-*hard.*

Proof: We give an L-reduction from DEGREE-4 NODE COVER, i.e the vertex cover problem on graphs with maximum degree at most 4, which is **MAX SNP**-complete [41, 42].

Let $H = (V, E)$ be a *connected* graph with $\Delta(H) \leq 4$, $V = \{1, 2, \ldots, n\}$ and $E = \{(e_j = (f_j, g_j) : 1 \leq j \leq m\}$. Let $\gamma(H)$ stand for the size of the minimum vertex-cover of H. We construct a graph $G = (V', E')$ as follows:

$$\begin{aligned} V' &= V \cup E \cup \{u_i, w_i : 1 \leq i \leq n\} \\ E' &= \{(e_j, f_j), (e_j, g_j) : 1 \leq j \leq m\} \cup \{(i, u_i), (u_i, w_i) : 1 \leq i \leq n\} \end{aligned}$$

Note that G is actually a bipartite graph. Owing to the nature of the graph, it is easy to see that there exists a minimum independent dominating set $\mathcal{I}$ such that $\mathcal{I} \cap E = \phi$. Also for $1 \leq i \leq n$, we must have $|\mathcal{I} \cap \{u_i, w_i\}| = 1$. It follows for $\mathcal{I}$ to be a dominating set, $\mathcal{I} \cap V$ must be a vertex-cover of H. It follows that $\alpha_d(G) \geq n + \gamma(H)$. Conversely, let $S \subseteq V$ is a minimum vertex-cover of G. Consider $T \subseteq V'$ defined by: $T = S \cup \{w_i : i \in S\} \cup \{u_i : i \in V - S\}$. It is easy to see that T is an independent dominating set of cardinality $n + |S|$, so that $\alpha_d(G) \leq n + \gamma(H)$. Thus, we have

$$\alpha_d(G) = n + \gamma(H).$$

That the reduction is an L-reduction follows easily from the fact $\Delta(H) \leq 4$ and that $\gamma(H) \geq |E|/\Delta(H) \geq (n-1)/4$. □

The above theorem, along with theorem 4.1, implies that independent domination does not admit a PTAS unless **P**=**NP**. Combining this with theorem 5.2, we get

Theorem 5.4 *Approximating independent dominating set to any constant factor is* **NP**-*hard.*

Since it is believed that **NP** has no quasi–polynomial deterministic time simulation [34], theorem 5.5 below states that independent domination is unlikely to admit polynomial time approximations even to *large* factors. The proof is similar to that of theorem 5.2 – a similar proof, for the problem of longest paths, can be found in [34].

Theorem 5.5 *For any $\epsilon > 0$, if there is a polynomial time algorithm for approximating independent dominating set to a factor of $2^{O(\log^{1-\epsilon} n)}$, then* **NP** $\subseteq$ *DTIME($2^{O(\log^{1/\epsilon} n)}$).*

A similar self–improvement property for clique is used to actually prove **NP**-hardness of finding a clique of size at least $n^{-\delta}$ times the largest clique, for some $\delta > 0$ [1, 3]. This is achieved by avoiding the rapid increase in problem size when using self–improvement, through the use of a combinatorial object called a *booster* and the notion of an appropriate booster product of a graph. No such booster product, however, is known for proving the hardness of independent domination, and it is interesting to investigate whether such an approach can be used to prove the hardness of approximating independent domination within n^{δ}, for some constant $\delta > 0$, though the next section actually proves much stronger negative results using problem–specific techniques.

5.2.2 Stronger Hardness Results

The reductions employed in this section are from the 3SAT problem. We use n to denote both the number of clauses and the number of variables (3SAT remains **NP**-hard under this trivial restriction as well).

Theorem 5.6 *For any $\epsilon > 0$, approximating independent domination to a factor of $N^{1-\epsilon}$ is* **NP***-hard, where N denotes the number of vertices in the input graph.*

Proof: Let ϕ be a 3SAT instance with n variables $x_1, x_2, \ldots, x_n$ and n clauses $C_1, C_2, \ldots, C_n$. Construct a graph G_ϕ as follows:

$$
\begin{aligned}
V(G_\phi) &= \{x_i, \bar{x}_i : 1 \le i \le n\} \cup \{C_{ij} : 1 \le i \le n, 1 \le j \le n^{k-1}\} \\
E(G_\phi) &= \{(x_i, \bar{x}_i) : 1 \le i \le n\} \\
&\quad \cup \{(x_i, C_{jp}) : x_i \in C_j \text{ and } 1 \le p \le n^{k-1}\} \\
&\quad \cup \{(\bar{x}_i, C_{jp}) : \bar{x}_i \in C_j \text{ and } 1 \le p \le n^{k-1}\}
\end{aligned}
$$

where $k \ge 3$ is a fixed integer that depends on ϵ. The exact choice of k will be mentioned later in the proof.

Observe that the only way to dominate the vertices $\mathcal{C}_j = \{C_{jp} : 1 \le p \le n^{k-1}\}$ is to either choose a x_i (or $\bar{x}_i$) such that $x_i \in C_j$ (or $\bar{x}_i \in C_j$) or to choose all members in $\mathcal{C}_j$. A minimum maximal independent set of G_ϕ will choose exactly one vertex from $\{x_i, \bar{x}_i\}$ for $1 \le i \le n$; thus each such set $\mathcal{I}$ will constitute a truth assignment for $\{x_i : 1 \le i \le n\}$ in an obvious way as: $x_i =$ true iff $x_i \in \mathcal{I}$. Clearly, $|\mathcal{I}| = n$ iff there exists a satisfying truth assignment for ϕ, for otherwise $\mathcal{C}_j \subset \mathcal{I}$ for some j, $1 \leq j \leq n$, which implies that $|\mathcal{I}| \geq n^{k-1}$. Hence

$$\begin{aligned} \alpha_d(G_\phi) &= n \text{ if } \phi \text{ is satisfiable} \\ &\geq n^{k-1} \text{ if } \phi \text{ is not satisfiable} \end{aligned}$$

Thus approximating $\alpha_d(G_\phi)$ to a factor of n^{k-2} is **NP**-hard. Let us now estimate n^{k-2} as a function of $N = |V(G_\phi)| = n^k + 2n$. Now, $n^{k-2} = (N-2n)/n^2 \geq (N-2n)/N^{2/k}$ since $N \geq n^k$. Thus,

$$\begin{aligned} n^{k-2} &\geq N^{1-2/k} - \frac{2n}{N^{2/k}} \\ &\geq N^{1-3/k} \end{aligned}$$

The last step can be justified as follows. Since $n \geq 2$, we have $2^{k-1} \leq n^k$ or $(2n^k)^{k-1} \leq n^{k^2}$. Also $n^k + 2n \leq 2n^k$ so that we have

$$\begin{aligned} (n^k + 2n)^{k-1} &\leq n^{k^2} \\ \Rightarrow N^{1-1/k} &\leq n^k = N - 2n \\ \Rightarrow N - N^{1-1/k} &\geq 2n \\ \Rightarrow N^{1-2/k} - \frac{2n}{N^{2/k}} &\geq N^{1-3/k} \end{aligned}$$

Hence approximating $\alpha_d(G_\phi)$ to a factor of $N^{1-3/k}$ is **NP**-hard. Choosing $k \geq \lceil 3/\epsilon \rceil$ proves the result. $\square$

Theorem 5.7 *For any $\epsilon > 0$, if the minimum independent dominating set can be approximated to within $N/(\log N)^{1+\epsilon}$ of the optimum even in $O(N^{\log^{\epsilon/2} N})$ time, then any 3SAT instance with n variables and clauses can be tested for satisfiability in $O(2^{(n/\alpha)^\delta})$ time where $\alpha > 0$ is a constant and $\delta < 1$ is a constant depending on ϵ.*

Proof: The same reduction of theorem 5.6 is used, however we assume that the 3SAT instance ϕ is such that either all the n clauses are satisfiable or at most $(1-\alpha)n$ clauses are simultaneously satisfiable for some *constant* $\alpha > 0$. The results on probabilistically checkable proofs [3] imply that there exists a constant $\alpha > 0$ such that testing such 3SAT instances ϕ for satisfiability is also **NP**-complete.

Instead of employing n^{k-1} vertices for each clause C_j (as in theorem 5.6), we now use

$$t = \frac{2^{\left(\frac{n}{\alpha}\right)^{1/(1+\epsilon)}}}{n}$$

vertices C_{jp}, $1 \leq p \leq t$, for each clause C_j. The reduction therefore is no longer a polynomial time one, but it is sub–exponential. The number N of vertices in the graph

G_ϕ thus constructed equals $nt = 2^{(n/\alpha)^{1/(1+\epsilon)}}$ if the lower order term is ignored (which can be done since we are dealing with asymptotic estimates).

Since, if ϕ is not satisfiable, at least αn clauses are not satisfied by any truth assignment, we have, paralleling the proof of theorem 5.6, that

$$\begin{aligned} \alpha_d(G_\phi) &= n \text{ if } \phi \text{ is satisfiable} \\ &\geq \alpha nt \text{ if } \phi \text{ is not satisfiable} \end{aligned}$$

Thus approximating $\alpha_d(G_\phi)$ to a factor of αt in $O(N^{\log^{\epsilon/2} N})$ time will yield a method to solve 3SAT also within the same asymptotic time bounds. Now,

$$\begin{aligned} N^{\log^{\epsilon/2} N} &= 2^{(n/\alpha)^{\frac{1}{1+\epsilon}}(n/\alpha)^{\frac{\epsilon}{2(1+\epsilon)}}} \\ &= 2^{(n/\alpha)^{\frac{1+\epsilon/2}{1+\epsilon}}}, \end{aligned}$$

thus $\delta = \frac{1+\epsilon/2}{1+\epsilon}$ satisfies the requirements of the theorem. Also, the approximation factor αt is equal to

$$\begin{aligned} \alpha t &= \frac{nt}{n/\alpha} \\ &= \frac{N}{(\log nt)^{1+\epsilon}} \\ &= \frac{N}{\log^{1+\epsilon} N}, \end{aligned}$$

and the bounds stated in the theorem follow. □

It is believed that no **NP**-complete problem can be decided in sub–exponential ($2^{O(n^\delta)}$ for some $\delta < 1$) deterministic time. The above theorem represents the strongest known negative result regarding the approximability of an unweighted **NP**-hard optimization problem. Even the clique problem, which cannot be approximated to within $1/N^{1/2-\epsilon}$ unless **P**=**NP** and within $1/N^{1-\epsilon}$ unless **NP**=co-**RP**, for any constant $\epsilon > 0$ [28], can be approximated in polynomial time to a factor of $\Omega(\log^2 N/N)$ [10] (note the factors mentioned are less than one since clique is a maximization problem). Thus the above result indicates that independent domination is indeed a very difficult problem.

5.3 Clique Dominating Set

The clique domination problem seeks to find a clique dominating set of minimum cardinality. A graph need not necessarily possess a clique dominating set , in fact a standard reduction from SAT can be used to prove that it is **NP**-complete to decide whether or not a graph has a clique dominating set.

Since any approximation algorithm must output a feasible solution, in light of this **NP**-completeness result, we assume that the approximation problem is to find an approximate clique dominating set given a graph G and some clique dominating set in G. This is because, unless we allow the input itself to contain some feasible solution, it is meaningless to talk of approximation algorithms for problems where deciding the existence of a feasible solution (in this case a clique dominating set) is itself **NP**-complete.

Basic Reduction Strategy

Our results on (in)approximability of clique domination parallel those of independent domination and use similar reductions. We now outline the basic reduction strategy.

Given a 3SAT instance ϕ with n clauses $C = \{C_1, C_2, \ldots, C_n\}$ and n variables $\mathcal{X} = \{x_1, x_2, \ldots, x_n\}$, construct a graph G_ϕ as follows. Initially form a *split* graph $H_\phi = (V_\phi, E_\phi)$ where $V_\phi = \mathcal{C} \cup \mathcal{X} \cup \mathcal{D}$ where

$$\begin{aligned} \mathcal{C} &= \{C_{ij} : 1 \le i \le n, 1 \le j \le t\} \\ \mathcal{D} &= \{D_{ij} : 1 \le i \le n, 1 \le j \le t\}, \end{aligned}$$

for some suitable parameter t which depends on n.

In the split graph H_ϕ, $\mathcal{X} \cup \mathcal{D}$ induces a clique while $\mathcal{C}$ induces an independent set. Edges between $\mathcal{C}$ and $\mathcal{X} \cup \mathcal{D}$ are given by

$$\begin{aligned} E'_\phi = \; & \{(x_i, C_{jp}) : x_i \in C_j \text{ and } 1 \le p \le t\} \cup \\ & \{(\bar{x}_i, C_{jp}) : \bar{x}_i \in C_j \text{ and } 1 \le p \le t\} \cup \\ & \{(C_{ij}, D_{ij}) : 1 \le i \le n, 1 \le j \le t\} \end{aligned}$$

Form G_ϕ from H_ϕ by removing the n edges $\{(x_i, \bar{x}_i) : 1 \le i \le n\}$. It is clear that $\mathcal{D}$ forms a clique dominating set of G_ϕ, we assume that $\mathcal{D}$ is also part of the input, along with G_ϕ, to the clique domination problem.

As in the proof of theorem 5.6, it can be shown that if ϕ is satisfiable, then G_ϕ has a clique dominating set of size n, else if at least $k \ge 1$ clauses of ϕ are left unsatisfied by *any* truth assignment to the variables in ϕ, then *any* clique dominating set of G_ϕ has size at least kt.

By choosing appropriate values of t, we can state the following result paralleling those of theorems 5.6 and 5.7:

Theorem 5.8 *For any $\epsilon > 0$, clique domination has no polynomial time factor $N^{1-\epsilon}$ approximation algorithm unless* **P**=**NP**, *and cannot be approximated to within $O(N/\log^{1+\epsilon} N)$ even in $O(N^{\log^{\epsilon/2} N})$ time unless the* **NP**-*complete problem 3SAT can be decided in sub–exponential deterministic time.*

5.4 Conclusions

This chapter investigated the approximability of the several variants of domination problems. While domination, total and connected domination admit polynomial time approximations with performance ratio $O(\log N)$, it is proved that the variants independent and clique domination are *unlikely* to admit – in polynomial time – even factor $O(N/\log^{1+\epsilon} N)$ approximations for any $\epsilon > 0$. But we believe that, in fact, neither of these problems can be approximated to within anything other than the trivial $O(N)$ factor. The reductions used here **cannot** be used to prove any stronger inapproximability results, so it is extremely challenging to improve the $O(N/\log^{1+\epsilon} N)$ bound proved in this chapter.

Chapter 6

Concluding Remarks and Open Problems

6.1 Contributions of the Thesis

An important method of overcoming the intractable nature of several graph–theoretic problems is to focus attention on special classes of graphs and/or to settle for suboptimal approximate solutions. The purpose of our work has been to prove the difficulty of tackling certain graph–theoretic problems even under these two situations. The main tools used for proving these results are polynomial time reductions and L–Reductions from suitably chosen canonical problems that are already known to be hard. We list below some of the results presented in this thesis:

- **NP**-hardness of finding minimum graph spanners in bipartite, chordal, split and bounded–degree graphs. These results settle an open problem raised in [12] and also greatly simplify some of the proofs presented there.
- **NP**-hardness of packing maximum number of vertex-disjoint triangles in chordal, planar, line and total graphs.
- **NP**-completeness of deciding the existence of a P_3-factor in a bipartite graph and a K_4-factor in a comparability graph.
- A polynomial time algorithm for the K_3-packing problem on split graphs and **NP**-hardness of K_4-packing on split graphs.
- Given any $\epsilon > 0$, one can find an integer $k \geq 1$ such that the approximation threshold δ_k of the independent set problem on k–iterated total graphs satisfies $0 < \delta_k \leq \epsilon$ (assuming $\mathbf{P} \neq \mathbf{NP}$).
- **NP**-hardness of approximating independent domination to any constant factor and *Quasi*-**NP**-hardness of approximating it within large $(2^{O(\log^{1-\epsilon} n)})$ factors using a property known as *self–improvement*.
- **NP**-hardness of approximating independent and clique dominating sets to within $N^{1-\epsilon}$ for any $\epsilon > 0$.
- If independent or clique dominating sets can be approximated within $N/\log^{1+\epsilon} N$ even in $O(N^{\log^{\epsilon/2} N})$ time, for any $\epsilon > 0$, then the **NP**-complete problem 3SAT has a sub–exponential deterministic time solution (which is believed to be unlikely). This result represents the strongest known negative result regarding the approximability of an unweighted graph–theoretic problem.

6.2 Open Problems

We have mentioned the open problems pertaining to the various chapters at the end of each chapter itself, we present them here in a consolidated fashion for ease of reference.

- What is the complexity of MINIMUM t-SPANNER, for $t \geq 2$, on planar graphs?
- Determine the complexity of MINIMUM 2-SPANNER on graphs with maximum degree Δ for $5 \leq \Delta \leq 8$, permutation graphs, and interval graphs.
- Can the P_3-packing problem be solved in polynomial time on split graphs? An answer to this question will enable determining the packing–hardness number of the class of split graphs. Note that a factor $1/2$ approximation is possible in polynomial time using an idea similar to that of theorem 3.8.
- Prove or disprove that the K_3-packing problem is **NP**-complete on comparability graphs. It would be interesting to see whether an approach similar to that of theorem 3.11 can be used to settle this question.

We next list a few open problems that arise from the study relating to approximability carried out in this thesis.

- Determine the best approximation ratio that can be achieved for the edge domination problem. We conjecture that the best ratio achievable in polynomial time is two. As remarked in section 4.6, a proof of this conjecture will imply that the approximation thresholds of the family of problems exhibited in chapter 4 is given exactly by the sequence $\{c_k\}_{k\geq 0}$.
- Prove or disprove that the fact that independent domination cannot be approximated to anything other than the trivial $O(N)$ factor in polynomial time. This problem appears to be pretty hard, and the ultimate negative result in this direction would be the **NP**-hardness (or even *Quasi*-**NP**-hardness) of approximating independent dominating set to within $o(N)$ (or even $O(N/\log N)$) of the optimal. A proof of such a result (if it holds that is!) will probably require an as yet undiscovered approach.

Bibliography

[1] N. Alon, U. Feige, A. Wigderson and D. Zuckerman, Derandomized graph products, *manuscript*, 1993.

[2] S. Arora, L. Babai, J. Stern and Z. Sweedyk, The hardness of approximate optima in lattices, codes and linear equations, *Proc. of the 34th IEEE Symposium on Foundations of Computer Science*, 1993, pp 724-733.

[3] S. Arora, C. Lund, R. Motwani, M. Sudan and M. Szegedy, Proof verification and intractability of approximation problems, *Proc.* 33^{rd} *IEEE Symposium on Foundations of Computer Science*, (1992), pp 13-22.

[4] S. Arora and S. Safra, Approximating Clique is **NP**-complete, *Proceedings of the 33rd IEEE Symposium on Foundations of Computer Science*, 1992, pp. 2-13.

[5] E. Amaldi and V. Kann, The complexity and approximability of finding maximum feasible subsystems of linear relations, *Theoretical Computer Science*, 147 (1995), pp 187-210.

[6] M. Bellare, S. Goldwasser, C. Lund and A. Russell, Efficient probabilistically checkable proofs and applications to approximation, *Proc. of the 25th Annual ACM Symposium on Theory of Computing*, 1993, pp 294-304.

[7] F. Berman, D. Johnson, T. Leighton and P. W. Shor, Generalized Planar Matching, *Journal of Algorithms*, 11(1990), pp 153-184.

[8] H. Bodlaender and K. Jansen, Restrictions of graph partition problems - Part I, manuscript(1994).

[9] F. T. Boesch and J. F. Gimpel, Covering the points of a digraph with point-disjoint paths and its application to code optimization, *Journal of the ACM*, 24(1977), pp 192-198.

[10] R. Boppana and M. Haldórsson, Approximating maximum independent sets by excluding subgraphs, *BIT*, 32 (1992), pp 180-196.

[11] L. Cai, Tree Spanners: Spanning Trees that approximate distances. Tech. Report 260/92, University of Toronto, (1992).

[12] L. Cai, **NP**-completeness of minimum spanner problems, *Disc. Appl. Math.*, 48 (1994), 187-194.

[13] L. Cai and M. Keil, Spanners in Graphs of Bounded Degree, *Networks*, 24 (1994), 233-249.

[14] G. J. Chang, The k-path packing and the k-covering problems, *Ars Combinatoria*, 25A (1988), pp 11-20.

[15] D. G. Corneil, H. Lerchs and L. S. Burlingham, Complement Reducible Graphs, *Discrete Applied Math.*, 3(1981), pp 163-174.

[16] D. G. Corneil, S. Masuyama and S. L. Hakimi, Edge-disjoint packings of graphs, *Discrete Applied Math.*, 50(1994), pp 135-148.

[17] D. G. Corneil and L. K. Stewart, Dominating Sets in Perfect Graphs, *Discrete Mathematics*, 86 (1990), pp 179-189.

[18] G. Cornuejols, D. Hartvigsen and W. Pulleyblank, Packing Subgraphs in a Graph, *Operations Research Letters*, 1(1982), pp 139-143.

[19] J. Edmonds, Paths, trees and flowers, *Canadian J. Math.*, 17(1965), pp 449-469.

[20] U. Feige, A threshold of $\ln n$ for approximating set cover, *Proceedings of the 28th Annual ACM Symposium on Theory of Computing*, 1996, pp 314-318.

[21] U. Feige, S. Goldwasser, L. Lovász, S. Safra and M. Szegedy, Approximating clique is almost **NP**-complete, *Proceedings of the 32nd IEEE Symposium on Foundations of Computer Science*, 1991, pp 2-12.

[22] M. Fürer and B. Raghavachari, Approximating the minimum degree spanning tree to within one from the optimal degree, *Proc. 3rd Annual ACM–SIAM Symposium on Discrete Algorithms*, (1992), pp 317-324.

[23] M. R. Garey and D. S. Johnson, The complexity of near-optimal graph coloring, *Journal of the ACM*, 23 (1976), pp 43-49.

[24] M. R. Garey and D. S. Johnson, Computers and Intractability: A guide to the theory of **NP**-completeness, *Freeman, San Francisco*, 1979.

[25] M. C. Golumbic, Algorithmic graph theory and Perfect graphs, *Academic Press, New York*, 1980.

[26] S. Guha and S. Khuller, Approximation Algorithms for Connected Dominating Sets, *Proc. of the 4th European Symposium on Algorithms*, 1996.

[27] F. Harary, Graph Theory, *Addison-Wesley, Reading, MA*, 1969.

[28] J. Hastad, Clique is hard to approximate within $n^{1-\epsilon}$, *Proceedings of the 37th IEEE Symposium on Foundations of Computer Science*, 1996, pp 627-636.

[29] P. Hell and D. G. Kirkpatrick, On generalized matching problems, *Info. Proc. Letters*, 12(1981), pp 33-35.

[30] P. Hell and D. G. Kirkpatrick, Packings by cliques and by finite families of graphs, *Discrete Mathematics*, 49(1984), pp 45-59.

[31] P. Hell and D. G. Kirkpatrick, Packings by complete bipartite subgraphs, *SIAM J. Algebraic Discrete Methods*, 7(1986), pp 199-209.

[32] J. D. Horton and K. Kilakos, Minimum Edge Dominating Sets, *SIAM Journal on Discrete Mathematics*, 6 (1993), pp 375-387.

[33] R. W. Irving, On approximating the minimum independent dominating set, *Information Processing Letters*, 37 (1991), pp 197-200.

[34] D. Karger, R. Motwani and G. D, S. Ramkumar, On approximating the longest path in a graph, *Proceedings of Workshop on Algorithms and Data Structures*, 1993, LNCS v. 709, pp 421-430.

[35] D. G. Kirkpatrick and P. Hell, On the complexity of general graph factor problems, *SIAM Journal on Computing*, 12(1983), pp 601-609.

[36] C. Lund and M. Yannakakis, On the hardness of approximating minimization problems, *Journal of the ACM*, 41(1994), pp 960-981.

[37] M. S. Madanlal, G. Venkatesan and C. Pandu Rangan, Tree 3-spanners on Interval, Permutation and Regular Bipartite Graphs, *Information Processing Letters*, 59 (1996), 97-102.

[38] S. Masuyama, On the tree packing problem, *Disc. Appl. Math.*, 35(1992), pp 163-166.

[39] S. Masuyama and T. Ibaraki, Chain Packing in Graphs, *Algorithmica*, 6(1991), pp 826-839.

[40] S. Micali and V. V. Vazirani, An $O(\sqrt{|V|}|E|)$ algorithm for finding maximum matching in general graphs, *Proc.* 21^{st} *Annual Symposium on the foundation of Comp. Sci.*, (1980), pp 17-27.

[41] C. H. Papadimitriou, Computational Complexity, *Addison-Wesley Publishing Company, Inc.*, (1994).

[42] C. H. Papadimitriou and M. Yannakakis, Optimization, approximation and complexity classes, *Journal of Computer and System Sciences*, 43 (1991), pp 425-440.

[43] D. Peleg and A. A. Schäffer, Graph Spanners, *Journal of Graph Theory*, 13 (1989), pp 99-116.

[44] M. Yannakakis and F. Gavril, Edge Dominating sets in graphs, *SIAM Jl. of Appl. Math.*, 38 (1980), pp 364-372.