CNAM • ENS Rennes • University of Liverpool

Mathieu MARI

Study of greedy algorithm for solving Maximum Independent Set problem

• Internship report for the Master MPRO •

Supervisor : Piotr Krysta

University of Liverpool

Victoria gallery.

Abstract.

The minimum degree greedy algorithm is a very natural algorithm to get a large independent set on a graph. It consists in iteratively selecting a vertex with minimum degree, and removing its neighbours until there are no remaining vertices. Bodlaender *et al.* showed that deciding whether this greedy algorithm outputs an optimal solution or not is hard to decide. In this paper, we give a new simpler proof of their result, and extend this result to the more rectrictive classes of bipartite graphs and cubic planar graphs. We also prove that the problem of finding the biggest set produced by this algorithm can not be approximated within a ratio $8/7 - \epsilon$.

April - July 2017

1 Introduction

Greedy algorithms are a general method to get a solution for some optimisation problems. It consists in a sequence of optimal local moves, and builds the solution step by step. These algorithms are usually fast, do not have a huge space complexity, and often produce very good solutions. For some problems, such as finding a minimum proper colouring on interval graphs this is even optimal [12]. In 1935, Hassler Whitney introduced the combinatorial notion of *matroid*, which describes a wide class of problems for which greedy algorithms work optimally. Unfortunately, the matroid structure is quite restrictive and a lot of optimisation problem can't be studied with this theory. Even when the greedy is not optimal, it can still be efficient in terms of approximation algorithms. Our general goal in this internship was to understand how the greedy algorithm behaves for Maximum Independent Set (MIS) which is one of the most important NP-complete problems in graph theory, and has a wide range of applications in several areas, such as map labeling, scheduling, or even molecular biology.

The basic idea to get a good independent set in a graph is to use a greedy algorithm. At each step, we choose an available vertex, we put it in the solution, and we remove this vertex and all of its neighbours from the graph, until there are no remaining vertices in the graph. At each step, one would like to find "the best vertex" to pick in order to get a good independent set at the end. The less we delete neighbours of the picked vertex, the best our solution will iteratively be, so it seems natural to pick at each step the vertex with minimum degree among all vertices.

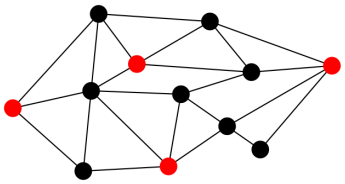

Figure 1: *Red nodes form an independent set.*

In terms of worst case approximation, this very simple method (called **Greedy**) is actually one of the best algorithms possible because it achieves a n-approximation ratio, while MIS can not be approximated within a factor $n^{1-\epsilon}$, where n denotes the number of vertices and $\epsilon > 0$, under widely believed assumptions (see [9]). This algorithm has also a very good performance on random graphs, because the size of the solution output by **Greedy** is the half of the optimal solution in expectation [4].

Our first goals were to understand when this greedy algorithm is actually efficient and when it isn't, what is the relation between the performance and differents parameters of the graphs (degrees, connectivity, density, etc.), and try to find a wide class of graphs where this algorithm is efficient.

This is very interesting to see that this algorithm is non-deterministic. Indeed, at each step one may find several vertices with minimum degree. Then, in order to mesure the performance of this greedy algorithm, we have to find the proper parameter to use, the worst or the best execution, or even the average size of a solution. For instance, the value of a smallest solution gives a bound on the approximation ratio of any greedy algorithm for Maximum Independent Set. For a given k, we can also ask the following question : Can this greedy algorithm produce a solution larger than k ? Since independent sets produced by this algorithm (*greedy sets*) form a subset of all independent set, this problem is closely related maximum independent set problem, but some interesting situations occur. On some classes of graphs, **Greedy** can produce an optimal solution but it remains hard to find this execution (see [3]). Interestingly enough, finding the best greedy set can be difficult even if finding the best independent set is easy (*e.g.* bipartite graphs, Theorem 4).

Studying the structure formed by greedy sets of a graph or a class of graphs has its own interest and could be useful in practice. For instance, in the area of combinatorial auctions, maximizing the social welfare in the case of single-minded bidders is actually equivalent to MIS. A very interesting approach when we want to get a incentive compatible (truthful) mechanism for Single-Minded bidders is to use a greedy algorithm because it achieves a good approximation of the optimal social welfare (for details read Algorithmic Game Theory [11], Chapter 11). In this greedy mechanism, the order on which bidders can be selected is fixed at the beginning, and this order can not change even if the order of preference on bidders becomes different because of the choices made by the mechanism. This fixed order is important to guarantee the truthfulness property but at the same time if the distribution of

degrees of the graph is quite homogeneous, the algorithm can be as bad as random choices. This is why we study in this paper the *adaptative* version the greedy algorithm, where nodes' degrees change during the execution. More generally, studying the efficiency of an algorithm relatively to its best output instead of the optimal value of the basic problems can be meaningful in some settings. We sometimes have few more assumptions about the instances of a problem which prevent us to use some algorithms. In the area of mechanism designs we consider the case where instances depend on the algorithm used (bidders may lie if it will increase their benefit). We can also cite the case of online problems,where the data of an instance comes part by part so that the algorithm can not consider the instance as a unique block.

1.1 Notations

Let $G = (V, E)$ be a simple undirected graph. When V and E are not explicit, we denote by $V(G)$ and $E(G)$ respectively the set of vertices and the set of edges of a graph G. The degree of a vertex $v \in V$ in G is denoted by $d(v)$. If u and v are two adjacent vertices in G, the corresponding edge in E is written (u, v) or just uv. The minimum degree in graph G is $\delta(G) := \min_{v \in V} d(v)$, and its maximum degree is $\Delta(G) := \max_{v \in V} d(v)$. The *neighbourhood* of a vertex v is the subset of vertices adjacent to v in G and it is denoted by $N_G(v)$. The *closed* neighbourhood contains the vertex : $N_G[v] := \{v\} \cup N_G(v)$. The subgraph of G induced by a subset $W \subseteq V$ is denoted by $G[W]$. A complete graph on n vertices is denoted by K_n and a complete bipartite graph on n and m vertices is denoted by $K_{n,m}$.

For a given $r \geq 1$, an algorithm $\mathcal{A}$ is an *r-approximation algorithm* for the maximisation (*resp.* minimization) problem Π, if for any instance I of Π, any solution output by $\mathcal{A}$ with value $\mathcal{A}(I)$ respects $r \cdot \mathcal{A}(I) \geq OPT_\Pi(I)$ (*resp.* $\frac{\mathcal{A}(I)}{r} \leq OPT_\Pi(I)$) where $OPT_\Pi(I)$ denotes the value of the optimal solution of Π on instance I.

An *independent set* in G is a subset of pairwise non-adjacent vertices. An independent set is *maximal* if it is not included in a strictly larger independent set and *maximum* if it has maximum cardinality among all independent sets. We denote by $\alpha(G)$ the cardinality of a maximum independent set, and we call Maximum Independent Set, or simply MIS, the following decision problem :

Name : MIS
Instance : A graph G and an integer k
Question : Is there an independent set S in G with cardinality $|S| \geq k$?

The algorithm called **Greedy** is defined by the following procedure.

Algorithm 1 Greedy

Require: a graph $G = (V, E)$
  $W \leftarrow V$
  $S \leftarrow \emptyset$
  while $W \neq \emptyset$ **do**
    Find a vertex $v \in W$ with minimum degree in $G[W]$
    $W \leftarrow W \setminus N_G[v]$
    $S \leftarrow S \cup \{v\}$
  end while
  return S

An output S of **Greedy** is a *greedy set* and its elements are called *picked* or *selected* vertices. It is easy to check that a greedy set is a maximal independent set.

We consider the following two decision problems that will later be associated with their corresponding maximization versions.

Name : MaxGreedy *Instance :* A graph G and an integer k *Question :* Is there a greedy set S in G with cardinality $\|S\| \geq k$?	*Name :* MinGreedy *Instance :* A graph G and an integer k *Question :* Is there a greedy set S in G with cardinality $\|S\| \leq k$?

We denote by $\alpha^+(G)$ (*resp.* $\alpha^-(G)$) the size of a maximum (*resp.* minimum) greedy set in G.

If a graph G has several connected components $G_1, \ldots, G_k$, then $\alpha(G) = \alpha(G_1) + \cdots + \alpha(G_k)$. **Greedy** also behaves like that. This algorithm proceeds with some local modifications, hence running the greedy algorithm on the whole graph or consecutively on all its connected components leads to the same result. Therfore, without loss of generality, graphs will always be assumed to be connected.

2 Counter-examples

If one tries to draw some small random connected graphs to see what happens when we run the greedy algorithm, then one will probably find that this algorithm returns each time a optimal solution. But we can wisely suspect some counter-examples, because Maximum Independent Set is NP-hard, while **Greedy** runs in polynomial time. The smallest graph for which **Greedy** can fail, *i.e.* there exists a sequence of choices which lead to a non-optimal solution, has six vertices and is shown in Figure 2. The first real counter-example has seven vertices (see Figure 3) and any sequence of choices leads **Greedy** to a non-optimal solution.

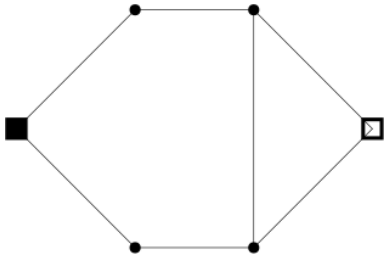

Figure 2: *Smallest graph where* ***Greedy*** *can fail. Picking* ■ *leads to a solution of size* 2 *while* □ *gives a solution of size* 3.

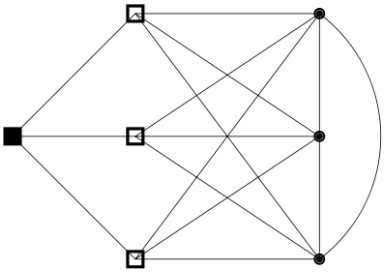

Figure 3: *Smallest graph where* ***Greedy*** *fails. Any solution contains* ■ *and has size* 2 *while a optimal solution has contain the* 3 *vertices* □.

This is instructive to look at some well-known results about Maximum Independent Set problem, because we can deduce some propeties about such greedy algorithms. One important result about maximum independent set problem is due to Håstad [9], who proved that there is no polynomial-time $(n^{1-\epsilon})$-approximation for MIS, for any $\epsilon > 0$, where n denotes the number of vertices, unless $P = NP$. In particular, this result holds for **Greedy**, so one should be able to find a family of graphs $(G_n)_{n\geq 0}$ so that each graph G_n has $O(n)$ vertices and $\frac{\alpha^+(G_n)}{\alpha(G_n)} = O(1/n)$. In the following section we describe such a class.

Håstad's theorem has another interesting consequences : **Greedy** is the best (among other) approximation algorithm for MIS ! Indeed, this algorithm will picked at least one vertex, so it is an n-approximation algorithm.

2.1 Two important families of counter-examples

We now present two families of graphs on which **Greedy** is as bad as we want. These examples are very useful because they will help later to design some gadgets used to prove some hardness results. The first is composed of general graphs and the second is a subclass of the more restrictive bipatite graphs.

General graphs. For $n \geq 3$, consider $G_n = (V, E)$ such that $V = \{x, v_1, \ldots, v_n\} \cup V'$. V' is a complete graph on n vertices, x is adjacent to every v_i and v_i is adjacent to all vertices of V' (see Figure 4). This graph has $2n + 1$ vertices, vertex x has degree n and vertex v_i degree $n + 1$, hence x is picked first by the greedy algorithm, and the size of the solution produced is 2. But the optimal solution is $\{v_1, \ldots, v_n\}$ with n vertices. On these graphs, **Greedy** becomes deterministic and we have $\frac{\alpha(G_n)}{\alpha^+(G_n)} = \frac{n}{2}$. Note that the graph drawn in Figure 3 is actually the graph G_3.

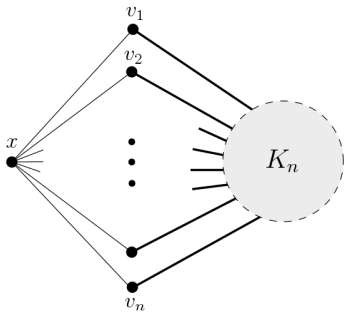

Figure 4: *Hard graphs for* ***Greedy***

Remark. Interestingly enough, we could avoid this bad result by randomly choosing vertices during the execution, eventually relatively to their respective degree. For example, if we set that the probability that v is picked by the greedy algorithm is $d(v) \cdot \left(\sum_{u \in V} d(u)\right)^{-1}$, thus the expected size of a solution is $\frac{2n^2+n^3}{3n^2+n} \sim_\infty \frac{n}{3}$ which gives an expected efficiency ratio of $1/3$.

Bipartite graphs. For bipartite graphs, the greedy algorithm has also an unbounded approximation ratio (see Figure 5). For $n \geq 3$, let $B_n = (U, V, E)$ be a bipartite graph. Suppose that $U = U_1 \cup \cdots \cup U_n$ and $V = \{x_1, \ldots, x_n\} \cup V'$, with $|U_1| = \cdots = |U_n| = |V'| = n$ and

$$E = \bigcup_{i=1}^{n} \{uv, u \in U_i, v \in V' \cup \{x_i, \ldots, x_n\}\}$$

It's not very difficult to see that **Greedy** will pick all vertices in column V which gives an greedy set of size $2 \cdot n$, while the maximum independent set in this graph has size at least n^2 by choosing column U. Then the gap is $n/2$.

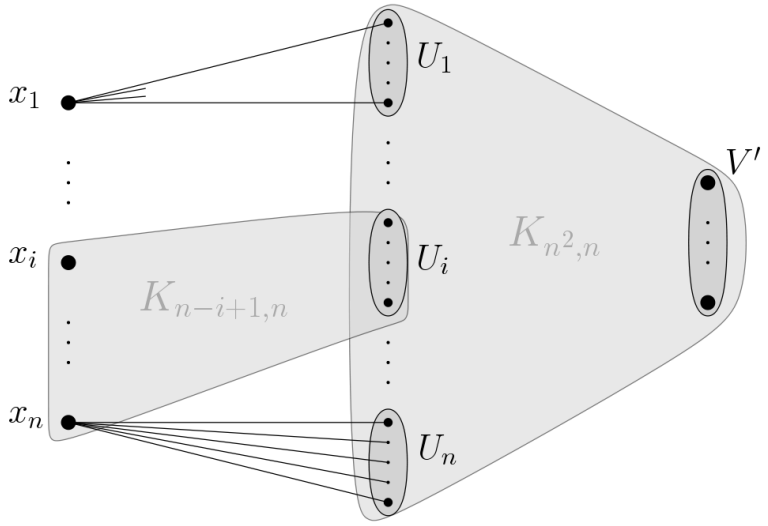

Figure 5: *Hard bipartite graphs for* ***Greedy****.*

3 Positive results

3.1 Some graphs where Greedy is optimal

We now focus on finding some classes of graphs for which **Greedy** is optimal. For what type of graphs picking the smallest degree vertex first always leads to an optimal solution ? Some properties about independent set could help to identify such classes.

PROPOSITION. *Let $G = (V, E)$ be a graph and $x \in V$ such that $G[N_G[x]]$ is a clique. Then x belongs to some maximum independent set.*

Indeed, if $G[N_G[x]]$ is a clique, its intersection with any independent set contains at most one vertex, and x can be added to any independent set in $G[V \setminus N_G[x]]$.

In particular any vertex of degree one belongs to a maximum independent set. For this reason Greedy will always produce an optimal solution on *trees*. We can prove using the same proposition with $|N_G[x]| = 3$ that **Greedy** is optimal for *maximal outerplanar* graphs, which is the class of graphs which can drawn on a sphere with all vertices adjacent to the same face, and which are maximal for graph inclusion.

There are other classes on which **Greedy** is optimal such as complete graphs, cycles, series-parallel graphs, cographs, split graphs, k-regular bipartite graphs [7, 3].

Another important but non-explicit class of graph for which **Greedy** is optimal is the class of well-covered graphs, introduced by Plummer [13], and widely studied (see [5] for a survey). A graph is *well-covered* if all its maximal independent sets have the same size. In particular since any greedy set is maximal, the greedy algorithm is optimal.

3.2 Greedy as an approximation algorithm for MIS

The first approach to understand the greedy algorithm is to consider it as an approximation algorithm for Maximum Independent Set. Our previous counter-examples show that for general graphs, **Greedy** is an n-approximation algorithm for MIS but this ratio can be improved for some subclasses.

One interesting remark about the two families of counter-examples in the previous section is that the maximum degree goes to infinity when n grows. This observation leads to the following result stated in [7].

Proposition 1. *Let S be a greedy set on the graph G with n vertices. We have*

$$|S| \geq \frac{n}{\Delta(G)+1}$$

Proof. Every time a vertex v is picked by **Greedy**, at most $\Delta(v)$ are removed, so at the end at most $|S|\Delta(G)$ vertices have been removed. Then G has at most $|S|+\Delta(G)|S|$ vertices which can be rewritten as $n \leq (\Delta(G)+1)|S|$. ♣

A direct consequence of this result is that **Greedy** becomes a $(\Delta+1)$-approximation algorithm for graphs with degree at most Δ, and later this ratio has been improved to $\frac{\Delta+2}{3}$ ([7]).

One can adapt this proof to see that this algorithm is a 6-approximation in *planar* graphs, using the fact that one can always find a node with degree at most five in a planar graphs. This result can also be seen as a consequence of the following theorem.

Theorem ([7]). *Let S be a greedy set on a graph G with n vertices and average degree $\bar{d}$. We have*

$$|S| \geq \frac{n}{\bar{d}+1}$$

This gives the slightly better results that **Greedy** is an $(\bar{d}+1)$-approximation algorithm for MIS, a result which was later improved to $\frac{2\bar{d}+3}{5}$ ([7]). For lower degree graphs, some more precise bounds are known such as 5/3 for graphs with degree at most three [7] and 3/2 for cubic graphs [8].

4 Negative results

The two following decision problems capture the question of the efficiency of the greedy algorithm *i.e.* is **Greedy** good relatively to a ratio $r \geq 1$ to compute a maximum independent set ? The first of these problems is about the existence of *at least one* sequence of choices which leads to a good solution and the other asks if *all* sequences of choices are good.

Name : $\textsc{GreedyOpt}_r^{\exists}$	*Name :* $\textsc{GreedyOpt}_r^{\forall}$
Instance : A graph G	*Instance :* A graph G
Question : $r \cdot \alpha^+(G) \geq \alpha(G)$?	*Question :* $r \cdot \alpha^-(G) \geq \alpha(G)$?

Note that r is a parameter of the problem and should be a rational number otherwise these two problems become undecidable [3]. GreedyOpt$_r^\forall$ belongs to class co-NP because an example of a polynomial certificate of answer *no* would be a greedy set S and an independent set S' such that $r \cdot |S| < |S'|$. Bodlaender *et al.* proved that this problem is difficult [3].

Theorem 1. GreedyOpt$_r^\forall$ *is co-NP-complete.*

Their proof is slighly difficult and we will give later a simple proof. This proof can be used exactly in the same way to show that GreedyOpt$_r^\exists$ is co-NP-hard but this problem does not belong to co-NP nor NP. They proved that this problem is actually DP-hard where the complexity class DP gathers all decision problems which can be expressed as a conjonction of a NP and a co-NP problem. One example of a DP problem is Exact Independent Set where given a graph and an integer k the goal is to decide if the maximum independent set in G has size exactly k.

Let $\mathcal{S}_1$ be the class of all graphs G such that $\alpha^+(G) = \alpha(G)$. One useful result proved by Bodlaender *et al.* states that MIS remains NP-hard on this class, *i.e.* assuming that **Greedy** can produce an maximum independent set does not help to determine such a sequence of choices to lead the greedy algorithm. For us it follows one important corollary.

Theorem 2. MaxGreedy *is NP-complete.*

We will show that this result remains true for the more restricted classes of bipartite graph and cubic planar graphs.

4.1 Our results

We can now summerize our main results :

- We found a easier proof of Theorem 1 than the one given by Bodlaender *et al.* [3]
- We show that MaxGreedy is hard to approximate within a ratio $8/7 - \epsilon$. (Corollary 1)
- We show that MaxGreedy and MinGreedy are NP-complete on bipartite graphs. (Theorem 4)
- We show that MaxGreedy is NP-complete on cubic planar graphs. (Theorem 5)

5 Hardness results for general graphs

In this section we design a new proof, easy to understand, for Theorem 1. We will reduce the complementary problem of MIS which is co-NP-complete to GreedyOpt$_r^\forall$. More precisely, given a rational number r, a connected graph G and an integer k, we build a graph G' in polynomial-time such that $\alpha(G) \leq k$ if and only if $r \cdot \alpha^-(G') \geq \alpha(G')$. The idea for this construction is to keep control of the size of the maximum indepedent set and at the same time force the greedy algorithm to be deterministic.

Let $r = \frac{p}{q} \geq 1$ be a rational where p and q are positives. Let $G = (V, E)$ be a graph and k an integer : we build the graph $G' = (V', E')$ shown in Figure 6, in the following way. First create p copies $G^1, \ldots, G^p$ of graph G *i.e.* $|V(G^i)| = |V(G)|$ and $(u^i, v^i) \in E(G^i) \Leftrightarrow (u, v) \in E$, for all i in $\{1, \ldots, p\}$. Create a complete graph K_{np} on $n \cdot p$ vertices. Then create l new vertices $x_1, \ldots, x_l$ where $l = qk - 1$.

$$V' := \{x_1, \ldots, x_l\} \cup V(K_{np}) \cup \bigcup_{i=1}^{p} V(G^i)$$

Now, create new edges from all graphs G^i to K_{np} and vertices $x_1, \ldots, x_l$, so that :

$$E' = \big\{(u, v), u \in V(G^i), 1 \leq i \leq p, v \in \{x_1, \ldots, x_l\} \cup V(K_{np})\big\} \cup E(K_{np}) \cup \bigcup_{i=1}^{p} E(G^i)$$

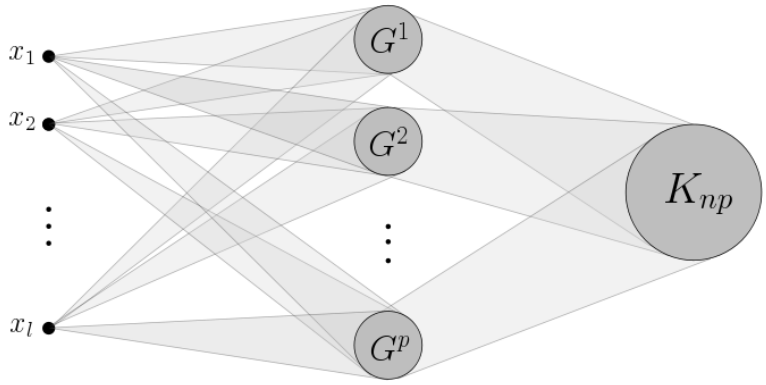

Figure 6: *The graph G' used in the proof of Theorem 1*

Claim. *Using the same notations we have $\alpha(G) \leq k$ if and only if $r \cdot \alpha^-(G') \geq \alpha(G')$*

Proof. Since G is connected, all its vertices have at least degree one, so the vertices with smallest degree in G' are $x_1, \ldots, x_l$ and thus the solution output by **Greedy** is $\{x_1, \ldots, x_l, y\}$ with size $l+1$ where y is any vertex in K_{np}. Then we have $\alpha^-(G') = \alpha^+(G') = l+1$. A maximum independent set in G' does not contain at the same time a vertex in G^i and a vertex in K_{np} or $\{x_1, \ldots, x_l\}$. Thus, we have $\alpha(G') = \max(p \cdot \alpha(G), l+1)$.

- Suppose $\alpha(G) > k$. We have $p \cdot \alpha(G) > pk \geq qk = l+1$, so $\alpha(G') = p\alpha(G)$. It follows that $r \cdot \alpha^-(G') = \frac{p}{q}(l+1) = pk < p \cdot \alpha(G) = \alpha(G')$
- Suppose $\alpha(G) \leq k$. We have $r \cdot \alpha^-(G') = r(l+1) \geq l+1$ and $r \cdot \alpha^-(G') = pk \geq p \cdot \alpha(G)$. Then $r \cdot \alpha^-(G') \geq \max(p \cdot \alpha(G), l+1) = \alpha(G')$. ♣

Note that since **Greedy** behaves deterministically on G', this reduction also proves that GreedyOpt$_r^{\exists}$ is co-NP-hard. Moreover, one can easily adapt this reduction for Exact Independence Number, to show that this problem is also DP-hard.

6 Inapproximability

In this section, we show that MaxGreedy is hard to approximate for general graphs. To reach this goal, we will present a way to simulate a 3-SAT formula by a graph, so that deciding the sequence of choices on this graph corresponds exactly to choose the valuation of the variables of the formula. At the end the size of the greedy set is almost proportional to the number of satisfied clauses on the given formula. The hardness properties of Max-3-Sat (*resp.* Min-3-Sat) will imediately be translated to MaxGreedy (*resp.* MinGreedy).

Theorem 3. *If* MaxGreedy *admits a r-approximation algorithm, with $r > 1$ then,* Max-3-Sat *admits a $(r+\epsilon)$-approximation algorithm, for any $\epsilon > 0$.*

Max-3-Sat is an NP-complete problem but also APX-hard. It is known that for any $\epsilon > 0$ there is no $(8/7 - \epsilon)$-approximation algorithm for this problem unless P=NP [10]. It directly follows an inapproximability result for our problem.

Corollary 1. *There's no $(8/7-\epsilon)$-approximation algorithms for* MaxGreedy*, unless P=NP.*

Proof of the Theorem. We suppose that there exists a polynomial-time algorithm $\mathcal{A}$ such that, for any graph G, the greedy set output by $\mathcal{A}$ has size $\mathcal{A}(G) \geq \frac{\alpha^+(G)}{r}$.

Let $\epsilon > 0$ and let ϕ be a 3-SAT formula with n variables $x_1, \ldots, x_n$ and m clauses $C_1, \ldots, C_m$.

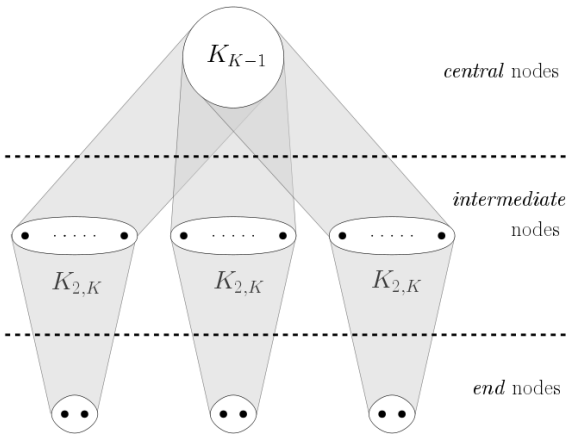

Figure 7: *Gadget used in the proof of Theorem 3*

We call the *frequency* of a literal l_i, denoted by $f(l_i)$, the number of clauses in which it belongs to. By changing the index of variables and switching literal l_i and $\bar{l_i}$, we can suppose *w.l.o.g.* that $f(x_i) \geq f(\bar{x_i})$ for all $1 \leq i \leq n$, and $f(x_1) \leq \cdots \leq f(x_n)$. Then we define $f_{\max} := f(x_n) = \max_{1 \leq i \leq n} f(x_i)$ and K an integer greater than $2f_{\max} + 3$. We build a graph G_ϕ as follows :

1. For each variable x_i, create two adjacent *literal nodes* x_i and $\bar{x_i}$ corresponding to each of its literals.

2. For each clause C_j create a *gadget* as shown in Figure 7. Each gadget contains $K - 1$ *central* nodes forming a clique K_{K-1} and $3K$ *intermediate* nodes divided into three groups of K vertices. Any intermediate node is adjacent in this gadget to all central nodes. Finally, for each group of intermediate nodes there are two *end* nodes which form with their corresponding intermediate nodes a complete bipartite graph on 2 and K vertices. Each pair of end nodes is associated to one of the three literals of the corresponding clause.

3. If a clause contains a literal x_i (*resp.* $\bar{x_i}$), create two edges between an available pair of end nodes of the corresponding gadget and the literal node x_i (*resp.* $\bar{x_i}$).

4. For each variable x_i such that $\Delta_i := f(x_i) - f(\bar{x_i}) > 0$, create $2\Delta_i$ new vertices $p^i_1, \ldots, p^i_{2\Delta_i}$, all adjacent to the literal node $\bar{x_i}$. This step ensures that x_i and $\bar{x_i}$ have exactly the same degree $2f(x_i) + 1$ in G_ϕ, then **Greedy** will have to choose between x_i and $\bar{x_i}$.

5. Finally add some edges and $2f_{\max} + 3$ new vertices such that the vertices created at step 4. form a clique with more than $2f_{\max} + 3$ vertices. We call this clique K_*.

We now study the behaviour of the greedy algorithm on the graph G_ϕ which can be decomposed into two phases. The first one lasts n turns and consists of choosing a valuation for the formula. During the second phase **Greedy** has to pick vertices on the remaining connected components. This phase is deterministic *i.e.* the size of the greedy set produced at the end only depends on the choices made during the first phase. We give more details below.

First phase : choice of a valuation. In this graph, for a given variable x_i, its corresponding literal nodes have same degree $2f(x_i) + 1$ and any other nodes have degree at least $2f_{\max} + 2$, even when some end nodes have been removed. In particular, **Greedy** will first pick x_1 or $\bar{x_1}$. For instance if x_1 is chosen, then $\bar{x_1}$ is removed and vice versa, some end nodes of some gadgets are removed and also eventually some nodes in K_*. One can easily check that the degree of other literal nodes have not changed, any gadget has minimun degree at least $2f_{\max} + 2$ and K_* still contains more than $2f_{\max} + 2$ vertices, and hence, x_2 and $\bar{x_2}$ are minimum degree vertices of the graph, *etc.* **Greedy** will consecutively choose non-deterministically between all x_i and $\bar{x_i}$ for i from 1 to n. We associate a valuation $\nu : \{x_1, \ldots, x_n\} \to \{0, 1\}$ to this choice such that $\nu(x_i) = 1$ if and only if literal node x_i is picked by the greedy algorithm.

Second phase : build the solution. After the first phase the remaining graph has $m + 1$ connected components corresponding to the m clauses and the remaining nodes of K_*. since the graph is disconnected, we can look at every connected components separately.

We claim that **Greedy** will pick exactly seven nodes on a given gadget if the corresponding clause $C_j = l_1 \vee l_2 \vee l_3$ is not satified by ν, and $3K$ nodes if at least one literal is satisfied. Indeed,

- If C_j is not satisfied, $\nu(l_1) = \nu(l_2) = \nu(l_3) = 0$, which means that $\bar{l_1}, \bar{l_2}$ and $\bar{l_3}$ have been picked during the first phase, and then l_1, l_2 and l_3 have been removed. Then the minimum degree vertices in the gadget are the end nodes with degree K. Then, **Greedy** will pick each pair of end nodes plus one central node, and remove all intermediate nodes. So exactly seven nodes are picked in this gadget.

- If C_j is satified, we can suppose *w.l.o.g.* that $\nu(l_1) = 1$, so the literal node l_1 has been picked and all of its neighbours removed, including the corresponding end nodes of the gadget. Then there are some intermediate nodes with degree $K-1$, which are now the smallest degree nodes of this gadget. In this situation, **Greedy** will pick all intermediate nodes and remove all others so at the end $3K$ nodes will be picked.

At the very end, **Greedy** picks one vertex in K_* and removes all other vertices.

Let c_ν be the number of clauses satisfied by ν. The size of the solution output is

$$n + c_\nu \cdot (3K) + (m - c_\nu) \cdot 7 + 1 = (n + 1 + 7m) + (3K - 7)c_\nu$$

and reciprocally, if there exists a greedy set $\mathcal{C}$ with size M on this graph then the valuation ν defined by

$$\nu(x_i) = 1 \text{ if and only if } x_i \in \mathcal{C} \tag{1}$$

then exacly $\frac{M-n-1-7m}{3K-7}$ clauses are satisfied by ν. In particular, if c^* denotes the maximum number of clause of ϕ simultaneously satisfied by a valuation, we have

$$\alpha^+(G_\phi) = (n + 1 + 7m) + (3K - 7)c^*$$

If there exists a polynomial time r-approximation algorithm $\mathcal{A}$ for MaxGreedy, then we should find on G_ϕ a greedy set $\mathcal{C}$ with size $M \geq \frac{\alpha^+(G_\phi)}{r}$ and the corresponding valuation ν defined as in (1) should satisfy c_ν clauses such that

$$\begin{aligned} c^*/c_\nu &= \frac{\alpha^+(G_\phi) - n - 1 - 7m}{M - n - 1 - 7m} \\ &\leq \frac{\alpha^+(G_\phi)}{M - (n + 1 + 7m)} \\ &\leq \frac{\alpha^+(G_\phi)}{\frac{\alpha^+(G_\phi)}{r} - (n + 1 + 7m)} \\ &= r \cdot \frac{1}{1 - \frac{r(n+1+7m)}{\alpha^+(G_\phi)}} \\ &\leq r + \epsilon \end{aligned}$$

for K big enouph. Indeed for any formula, at least one clause is satisfiable so $c^* \geq 1$ and it follows that $\alpha^+(G_\phi) \geq 3K$. Therefore, when K goes to infinity, the ratio $\frac{r(n+1+7m)}{\alpha^+(G_\phi)}$ goes to zero. This graph G_ϕ has size $O(m(n + K))$, and a suitable value of K depends polynomialy on n and m, so the graph G_ϕ can be computed in polynomial time.

♣

Remark. This reduction from Max-3-Sat is an other proof that MaxGreedy is NP-Hard. Moreover, since Min-3-Sat (find the valuation which satifies the minimum of clauses) is co-NP-Hard, we can easily prove that finding the minimum greedy sequence (we remind that the independent set should be maximal) is co-NP-hard and even APX-hard based on the fact that Min-3-Sat can not be approximate within a factor $7/6 - \epsilon$ [2].

7 Bipartite graphs

In this section we adapt the reduction from Max-3-Sat used in section 6 to prove that MaxGreedy remains a hard problem for bipartite graphs. This is an interesting result, because a maximum independent set can be computed in polynomial time in bipartite graphs using for example linear programming. For this reason, restricted to bipartite graphs, GreedyOpt$_r^\exists$ now belongs to class NP. A polynomial length certificate is a sequence of choices for the greedy algorithm. Then we compare the size of the greedy set with the size of a maximum independent set that we can compute in polynomial time.

THEOREM 4. MAXGREEDY *is NP-complete on bipartite graphs*

The idea for this proof is almost the same as the reduction used to show the inapproximability for general graphs, but we should garantee that the graph G_ϕ built during this reduction is bipartite.

The difficulty comes from the connection between the literal nodes and the gadgets. If we look at a bipartite graph as a 2-colourable graph, then all end nodes should have the same colour and positive literal nodes should have a different colour than negative ones. If a clause contains a positive and a negative literal at the same time, then it becomes impossible to colour this graph with only two colours. Therefore, we will make sure that clauses contains only positive or only negative literals.

The second difficulty comes from the clique K_*. To avoid this clique, we will ensure that both literals associated to any variable have the same frenquency. We now define a variant of the satisfaction problem called VAR-SAT, satisfying our criteria.

DEFINITION. *A logic formula in CNF form is a* VAR-SAT *formula if*

Bounded clauses. *Each clause contains two or three literals.*

Monotony. *All clauses are* monotone*, i.e. contains only positive literals or only negative literals.*

Same frequency. *For each variable, its positive and negative literals both appear exactly one or two times,* i.e. $f(x) = f(\bar{x}) \in \{1, 2\}$ *for all variables* x*.*

EXAMPLE. $xyz \wedge \bar{y}\bar{z} \wedge zw \wedge \bar{x}\bar{z}\bar{w}$ *is an example of* VAR-SAT *formula.*

The decision problem VAR-SAT is : given a VAR-SAT formula, is there a valuation which satisfies simultaneously all of its clauses ?

LEMMA 1. VAR-SAT *is NP-complete.*

A formula of VAR-SAT can be got from a formula of 3-SAT using some of the equisatisfiable-preserving reductions described below.

Frequency at most three. If a variable x appears in more than three clauses, say k clauses, then add k new variables denoted by $x_1, \ldots, x_k$, replace x in the i-th clause by x_i, add the new clauses $\bar{x_i} \vee x_{i+1}$ for $1 \leq i \leq k-1$ and $\bar{x_k} \vee x_1$. The k new clauses are simultaneously satisfied if and only if all x_i get the same value. Therefore, these formula are equisatisfiable, and each variable x_i appears now at most three times.

Monotonicity. If a clause C is not monotone, then denote C_+ and C_- respectively the positive and negative literals of $C = C_+ \vee C_-$. Then, add a new variable u and transform C into $(C_+ \vee u) \wedge (C_- \vee \bar{u})$. These two clauses are both monotone and equivalent to C. Note that, if C has size two or three, then these new clauses have still size two or three.

Equal frequency. If there exists a variable x such that its positive and negative literals do not have the same frequency, for instance if x appears once and $\bar{x}$ appears twice, then add two new variables u and v and add two new monotone clauses, $x \vee u \vee v$ and $\bar{u} \vee \bar{v}$.

The technique used to reduce the frequency is inspired from [15].

Proof. To show that VAR-SAT remains NP-hard, we prove that, given any formula ϕ of 3-SAT, we can compute in polynomial time a formula ϕ' such that these two formula are equisatisfiable.

We can construct ϕ' from ϕ using the following algorithm.

1. If there exists a variable x such that only its positive (or only negative) literals appear, then remove all the clauses in which x (or $\bar{x}$) appears.

2. For all variables which appears (both positive and negative literals) more that three time, apply the *frequency at most three* reduction.

3. Apply the *monotonicity* reduction to any clause which is not monotone.

4. For any variable x, for which x and $\bar{x}$ do not appear the same number of times, apply the *equal frequency* reduction.

One can check that at the end, the formula contains only clauses of size two or three and that the last step preserves monotonicity. Since after Step 2, variables have frequency three, if there exists a variable x such that x and $\bar{x}$ do not appear the same number of times, one should appear once and the other twice.

♣

Let ϕ be a VAR-SAT formula with n variables m_2 clauses of size two and m_3 clauses of size three, and k an integer. The construction of G_ϕ is the same than the one described in Section 6 except that the gadgets are slightly different :

- Since the maximum frequency of any variable is two, it is sufficient to fix $K = 7$ for all gadgets.
- We replace the clique formed by the central nodes by 6 isolated vertices to get the gadgets bipartite.
- We remove a group of intermediate nodes and end nodes for gadgets associated with a clause containing only two literals.

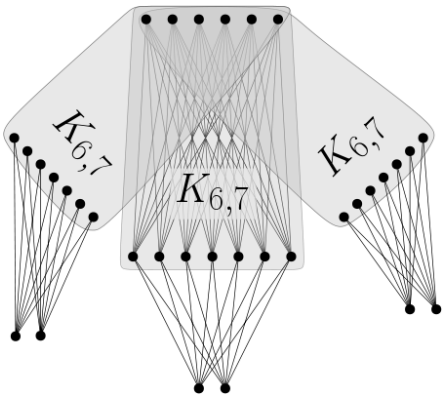

Figure 8: *Bipartite gadget associated with a clause with three literals*

To summarize, the gadget corresponding to a clause of size three (*resp.* two) has 6 *central nodes*, three (*resp.* two) groups of seven *intermediate* nodes which form with the central a complete bipartite graph $K_{6,7}$ and three (*resp.* two) pairs of *end* nodes forming with their corresponding intermediate nodes a complete bipartite graph $K_{2,7}$. See Figure 8.

Then, graph G_ϕ is bipartite. Indeed, all its components are bipartite and the clauses are monotone so that the edges created between the literal nodes and the gadgets have not created some odd cycles. This graph has $2n + 24m_2 + 33m_3$ vertices so it can be made in polynomial time.

CLAIM. *ϕ is satisfiable if and only if $\alpha^+(G_\phi) \geq n + 14m_2 + 21m_3$*

Proof. The behaviour of **Greedy** can be decomposed exactly in the same way than in Section 6. Then ϕ is satisfiable if and only if there exists a greedy set which contains all the intermediate nodes plus n literal nodes, *i.e.*, at least $n + 14m_2 + 21m_3$ vertices. ♣

Remark. The graph G_ϕ has a maximum degree bounded by 21 (central nodes), and this proof was designed is order to be easy checkable so one can easily prove that the problem is still hard for bipartite graphs with degree bounded by d, for some $3 \leq d \leq 21$.

8 Cubic planar graphs

A cubic graph is a graph where any vertex has degree exactly three. In this section we prove that finding a maximum greedy set is NP-complete even on the very restricted class of cubic planar graphs. We know that MIS is NP-hard in this class [6], and even APX-hard for cubic graphs [1].

An easy way to prove that MAXGREEDY is hard would be to prove that $\alpha(G) = \alpha^+(G)$ when G is planar and cubic. But Halldorson *et al.* [8] proved that it is not always the case (see Figure 9). One reason for that difference is that the only vertex of degree three picked by **Greedy** is the first one. After that, there always exists a vertex with degree one or two which is a 2-neighbour of an other vertex previously picked. Then sometimes a neighbour of a removed vertex is forced to be picked even if it is not supposed to belong to the maximum solution.

To reach our goal, we will design a reduction from MIS in planar cubic graphs and for that, we first introduce some gadget on the edges which will help to avoid these forced moves.

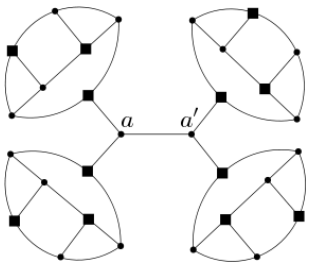

Figure 9: *The only maximum independent set (square nodes) has size* 12 *but any greedy set contains* a *or* a'.

THEOREM 5. MAXGREEDY *is NP-complete for planar cubic graphs.*

Proof. Let $G = (V, E)$ be a cubic planar graph with m edges. Let us construct a graph G' by replacing each edge $uv \in E$ by the structure $\mathcal{H}_{uv}$ described in Figure 10. We call

$$V' := V(G') = V \cup \bigcup_{e \in E} V(\mathcal{H}_e)$$

and

$$E' := E(G') = \bigcup_{uv \in E} (E(\mathcal{H}_e) \cup \{au, gv\})$$

where au and gv correspond to the edges connecting u and v to the graph $\mathcal{H}_{uv}$.

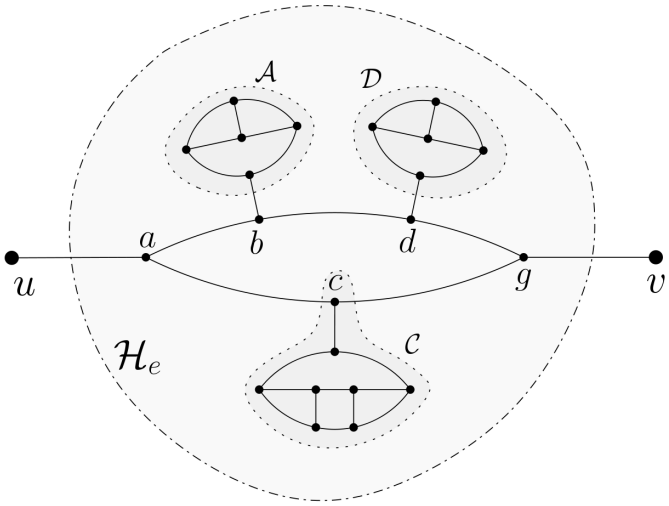

Figure 10: *Each edge* $e = uv$ *is replaced by this gadget* $\mathcal{H}_e$

G' has order $|V| + 22m$ and can be computed in polynomial time.

CLAIM. *Let* $S' \subseteq V'$ *be an independent set in* G' *and* $e = uv \in E$. *Then,* $|V(\mathcal{H}_e) \bigcap S'| \leq 9$. *Moreover, if both* u *and* v *belong to* S' *then* $|V(\mathcal{H}_e) \bigcap S'| \leq 8$.

We can easily check that $|\mathcal{A} \cap S'| \leq 2$, $|\mathcal{D} \cap S'| \leq 2$, $|\mathcal{C} \cap S'| \leq 3$ and $|\{a, b, d, g\} \cap S'| \leq 2$. Thus, $|V(\mathcal{H}_e) \bigcap S'| \leq 9$. Moreover, if both u and v belong to S' then $\{a, g\} \cap S' = \emptyset$ and then $|\{a, b, d, g\} \cap S'| \leq 1$ which gives $|V(\mathcal{H}_e) \bigcap S'| \leq 8$.

CLAIM. $\alpha(G') \leq \alpha(G) + 9m$

Let $S' \subseteq V'$ be an independent set in G'. Denote by F the set of edges of G which have both end nodes in S'. We have

$$|S' \cap V| - |F| \leq \alpha(G)$$

Indeed, $(S' \cap V) \setminus \{x_e, e \in F\}$ is a independent set in G where u_e is one of the two vertices incident to an edge $e \in F$. Then,

$$|S' \cap V| - |F| \leq |(S' \cap V) \setminus \{x_e, e \in F\}| \leq \alpha(G)$$

It follows that

$$\begin{aligned}
|S'| &= |S' \cap V| + \sum_{e \in E} |\mathcal{H}_e \cap S'| \\
&\leq (\alpha(G) + |F|) + (\sum_{e \in F} |\mathcal{H}_e \cap S'| + \sum_{e \notin F} |\mathcal{H}_e \cap S'|) \\
&\leq (\alpha(G) + |F|) + (\sum_{e \in F} 8 + \sum_{e \notin F} 9) \\
&= (\alpha(G) + |F|) + (\sum_{e \in E} 9 - \sum_{e \in F} 1) \\
&= \alpha(G) + 9m
\end{aligned}$$

Because this inequality is true for any independent set S', we have $\alpha(G') \leq \alpha(G) + 9m$.

Claim. *There exists a greedy set S' in G' of size $\alpha(G) + 9m$.*

Let S be a maximum independent set in G. Construct the set S' as follows

- While there exists some unpicked nodes in G' do
  1. If there exists an unpicked vertex $u \in S$ with minimum degree, add u to S' and nodes b and g in all adjacent gadgets $\mathcal{H}_{uv}$ (see Figure 10)
  2. Otherwise, there exists a vertex of type a with minimum degree in some gadget $\mathcal{H}_{uv}$.
     - If $v \in S$, add a and d to S'
     - If $v \notin S$, add a and g to S'
- Run **Greedy** on the remaining connected components $\mathcal{A}, \mathcal{D}$ and $\mathcal{C}$ of graphs $\mathcal{H}_e$ which have not been picked yet.

At the end, we have $S' \cap V = S$ and $|S' \cap V(\mathcal{H}_e)| = 9$ for all e in E. Then the greedy set S' has the desired size.

Therefore, $\alpha^+(G') = \alpha(G') = \alpha(G) + 9m$ and then for any integer k we have

$$\alpha^+(G') \geq k + 9m \quad \text{if and only if} \quad \alpha(G) \geq k$$

.

♣

8.1 Approximation algorithms in cubic graphs

Halldorsson *et al.* proved that the performance ratio of **Greedy** in cubic graphs is 3/2 [8]. In a more precised greedy algorithm called **MoreEdges**, **Greedy** should always select a vertex which has a neighbour with degree three if he has to choose among several degree two vertices.

The performance ratio of **MoreEdges** is improved to 17/12. It imediately follows that **MoreEdges** is a $\frac{17}{12}$-approximation algorithm for MaxGreedy in cubic graphs, because this algorithm outputs a greedy set. Even if Halldorsson *et al.* proved that this ratio is tight for MIS, it might be possible that this ratio is even better for MaxGreedy.

9 Further work

We give a list of questions and thoughts which could be some interesting directions for future researchs.

- Vertex Cover and more generally Set Cover are minimization problems for which **Greedy** is an interesting approach. For instance, in the general setting it produces a $O(\log n)$-approximation [16]. Is it hard to decide if **Greedy** can produce an optimal solution for these problems ?

- The proof of inapproximability in the general case and the proof of hardness for bipartite graphs are very similar in the structure. Unfortunately, we do not know if the special case Var-Sat is hard to approximate or not that's why we only get a NP-hardness result. We think that MaxGreedy is hard to approximate even in bipartite graphs.

- An interesting property of degree at most three graphs and more generally bounded degree graphs is that there is a finite number of k-neighbouroods (nodes at distance at most k from a given root). In [8], Halldorsson and Yoshihara look at the 2-neighbourhood to help the greedy algorithm to choose when two nodes have minimum degree (see **MoreEdges**). This leads to a stricly better approximation ratio. More generally, is it possible to explore the k-neighbourhood to help **Greedy** in its choices to get a better ratio depending on k ?

- What about the *non-adaptative* version of **Greedy** where the order of choosing vertices is fixed from the beginning ?

10 Acknowledgments

I was very glad to have Piotr Krysta as a supervisor during this internship, and I am very grateful for his kindness and friendship. I really appreciate to be considered as another researcher and not just as a student. I thank Nan Zhi for the pleasant moments of research we had together (with Piotr too). I thank the people of the department of computer science of the university of Liverpool for welcoming me and helping me during this internship. I also thank Sang-Ki, Tom and Reino for nice coffees and football games.

References

[1] Paola Alimonti and Viggo Kann. Some apx-completeness results for cubic graphs. *Theoretical Computer Science*, 237(1):123 – 134, 2000. 11

[2] Adi Avidor and Uri Zwick. Approximating min 2-sat and min 3-sat. *Theory of Computing Systems*, 38(3):329–345, May 2005. 9

[3] Hans L Bodlaender, Dimitrios M Thilikos, and Koichi Yamazaki. It is hard to know when greedy is good for finding independent sets. *Information Processing Letters*, 61(2):101–106, 1997. 1, 5, 6

[4] Amin Coja-Oghlan and Charilaos Efthymiou. On independent sets in random graphs. *Random Struct. Algorithms*, 47(3):436–486, 2015. 1

[5] Michael D. Plummer. Well-covered graphs: A survey. 16:253–287, 07 1993. 5

[6] M.R. Garey, D.S. Johnson, and L. Stockmeyer. Some simplified np-complete graph problems. *Theoretical Computer Science*, 1(3):237 – 267, 1976. 11

[7] M. M. Halldórsson and J. Radhakrishnan. Greed is good: Approximating independent sets in sparse and bounded-degree graphs. *Algorithmica*, 18(1):145–163, May 1997. 5

[8] Magnús M. Halldórsson and Kiyohito Yoshihara. *Greedy approximations of independent sets in low degree graphs*, pages 152–161. Springer Berlin Heidelberg, Berlin, Heidelberg, 1995. 5, 11, 13, 14

[9] Johan Hastad. Clique is hard to approximate within n1 – e. 0:627, 01 1996. 1, 3

[10] Johan Håstad. Some optimal inapproximability results. *J. ACM*, 48(4):798–859, July 2001. 7

[11] Noam Nisan, Tim Roughgarden, Eva Tardos, and Vijay V. Vazirani. *Algorithmic Game Theory*. Cambridge University Press, New York, NY, USA, 2007. 1

[12] Stephan Olariu. An optimal greedy heuristic to color interval graphs. *Information Processing Letters*, 37(1):21 – 25, 1991. 1

[13] Michael D. Plummer. Some covering concepts in graphs. *Journal of Combinatorial Theory*, 8(1):91 – 98, 1970. 5

[14] Brenda S. Baker. Approximation algorithms for np-complete problems on planar graphs. 41:265–273, 12 1983.

[15] Craig A Tovey. A simplified np-complete satisfiability problem. *Discrete Applied Mathematics*, 8(1):85–89, 1984. 10

[16] Vijay V. Vazirani. *Approximation Algorithms*. Springer-Verlag New York, Inc., New York, NY, USA, 2001. 13