# FRUGAL METHODS FOR THE INDEPENDENT SET AND GRAPH COLORING PROBLEMS

BY MAGNÚS MÁR HALLDÓRSSON

A dissertation submitted to the

Graduate School—New Brunswick

Rutgers, The State University of New Jersey

in partial fulfillment of the requirements

for the degree of

Doctor of Philosophy

Graduate Program in Computer Science

Written under the direction of

Ravi Boppana

and approved by

______________________________

______________________________

______________________________

______________________________

New Brunswick, New Jersey

Oct, 1991

## ABSTRACT OF THE DISSERTATION

# Frugal Methods for the Independent Set and Graph Coloring Problems


by Magnús Már Halldórsson

Dissertation Director: Ravi Boppana



We consider two classical hard graph problems: finding the maximum *independent set* of vertices, and *coloring* the vertices with fewest colors possible.

We are interested in *frugal* algorithms for these problems, ones that use limited amount of computing capabilities. Such algorithms yield approximate, rather than exact, solutions, and are valued according to their *performance guarantee*, which is the maximum factor that approximate solutions can differ from optimal ones. We are primarily interested in two classes of frugal algorithms: polynomial-time, and on-line algorithms.

We present several polynomial-time algorithms for obtaining large approximations in graphs containing large independent sets, and use them to improve the best performance guarantees known for both problems. We show how nearly all effective algorithms for these problems fall into yet another category of frugal algorithms, based on removing subgraphs, and obtain tight lower bounds on their performance.

For on-line graph coloring, we give strong lower bounds for both deterministic and randomized algorithms, that hold even if the models are weakened in several ways, and improve the best performance guarantee known for randomized on-line coloring. Finally, we prove simple tight bounds on two interpretations of the on-line independent

set problem and some of its variations.

# Preface

This thesis deals with two classical, hard problems on graphs: coloring the vertices, and finding the largest independent set in the graph. More specifically, it is about ways and means for solving these problems approximately – attempting to solve them with limited amounts of resources. It contains algorithms, and analyses of their performance, that show that certain things can be done, and it also contains lower bounds, or proofs that other things can not be done.

As any doctoral dissertation, this one is not written for the layperson in mind. The contents and exposition are those appropriate for the academic journals in theoretical computer science. However, I subscribe to Halmos's rule of writing in English, as opposed to formalism, and have therefore tried to keep new notation, acronyms, and pedantry to an absolute minimum. As for other writing principles, I try to follow the Strunkian minimalist wisdom, albeit not always successfully.

The organization of the thesis is approximately as follows:

The first chapter introduces the problems, their relationship, the terminology and notation, the models of computation, and alternative approaches.

Chapter 2 is meant to be a comprehensive review of previous work along these lines, with occasional additional observations of our own.

Chapter 3 contains results on polynomial-time approximation algorithms. We give algorithms that, among other, improve the best performance guarantees known, and prove lower bounds for a certain category of algorithms introduced.

Chapter 4 contains results on on-line algorithms. In particular, we show that various extensions of on-line algorithms are inherently ineffective for these problems. We also give an improved randomized on-line coloring algorithm.

The epigraphs starting each chapter are from Hávamál (The Words of Odin the

High One) from Elder (Sæmundar) Edda. We hope the reader will enjoy the few drops of classical wisdom. The originals are given in modern Icelandic spelling, taken from [33], while the translations are from [15] and [45].

## Acknowledgements

*Ungur var eg forðum,*
*fór eg einn saman,*
*þá varð eg villur vega;*
*auðigur þóttumk,*
*er eg annan fann,*
*maður er manns gaman.*

*Young and alone*
*on a long road,*
*once I lost my way;*
*Rich I felt*
*when I found another;*
*Man rejoices in man.*

As the pinnacle of my education, this thesis is distilled from the input and advice of great many teachers. I can name only a few, perhaps the most pronounced.

From Barna og Grunnskóla Stykkishólms: Lúðvíg Halldórsson, Ólafur H. Torfason, og Þórður Jóhannesson.

From Menntaskólanum við Hamrahlíð: Valdimar Valdimarsson, Vilhjálmur Kjartansson (Villi radíó), Halldór Halldórsson, Őrnólfur Thorlacius, Stefán Briem.

From University of Oregon: William M. Kantor, Richard J. Higgins, and my six writing instructors, but especially David Harrison, Dick Koch, and Chris Wilson.

From Rutgers: Ann Yasuhara, for bearing with my "checkered career"; Barbara Ryder, for motherly discipline; Ryan Hayward, for the concepts of "key llama" and "P. O. C."; Chris Tong, for a vivid introduction to AI; Eric Allender, for a careful exposition of complexity theory; Ken Kaplan, for effectiveness; Endre Szemerédi, for once asking me for a lecture; Gábor Tardos, for quick insights into my questions; and, Haym Hirsh, for interesting chats (at unusual occasions) about learning algorithms and combinatorics.

I thank all Rutgers computer science graduate students for being enjoyable comrades. Specifically, I thank, in somewhat of a chronological order, Boyce Byerly for taking the heat for the `dcs.general` flames as well as hosting Château Kajimoto so charmingly, Martin Caroll, for sharing LaTeX wizardry, Ed Segall for co-running the colloquia, Srimat Chakradhar, Ashok Kumar Arora, and Sizheng Wei for accepting my monopoly of `charybdis` and making Hill 523 a homey place, Chak for the energetic "let's-kick-some-butt attitude towards research, Shiva Chaudhuri for the "we're-in-this-together"-ness, and Jaikumar Radhakrishnan for the combinatorial insight and interest in my research problems.

The members of my thesis committee get my kudos for accepting that task. So do Valentine Rolfe and Priscilla Rasmussen for taking care of the "little" things.

I thank the Department of Computer Science (DCS) and Laboratory for Computer Science Research (LCSR) at Rutgers, the National Science Foundation (NSF), and the Center for Discrete Mathematics and Theoretical Computer Science (DIMACS) for their financial support. Without their support I would still be toiling. The NSF deserves a commendation for the vision shown by setting up the DIMACS center.

The research reported here is in collaboration with a few people. Sections 3.2,3.3, and 3.5 are joint with Ravi Boppana. Some of the results of sections 4.2.1 and 4.2.3 were independently, and earlier, discovered by Márió Szegedy, and will be reported in a joint paper. I thank both of them for permitting me to include their work here. Also, thanks to Sundar Viswanathan for corresponding about his and Márió's results. Finally, the results of section 4.2.2 were strongly influenced by discussions with Jaikumar Radhakrishnan and Gabór Tardos, to whom I am grateful.

I am much indebted to my adviser, Ravi Boppana. It is hard to imagine this research being done without his guidance. Attention was ample, and direction non-intrusive but subtly crucial. Finding him was definitely my lucky break.

My parents deserve more appreciation than I'll ever manage to utter in their company; thanks for patiently standing behind us. Thanks to Halldóra Miyoko-chan for the I ♡ U's.

And last but not the least, the work behind this document was an on-line exercise in the execution of RealLife$^{\text{TM}}$, performed joint with my wife, Yayoi. Gokurousama deshita.

## Dedication

Ritgerð þessa tileinka ég móðurafa mínum heitnum, Bjarna Vilhjálmssyni íslenskufræðingi.

# Table of Contents

# List of Figures

# List of Abbreviations

$\alpha(G)$ Independence number of the graph $G$, or the size of a maximum independent set in $G$.

$cl(G)$ Clique number of the graph $G$.

$\chi(G)$ Chromatic number of the graph $G$, or the minimum number of colors needed to color $G$.

$i(G)$ Independence ratio of $G$, or the independence number divided by the number of vertices.

$r(s, t)$ The Ramsey number: the smallest integer $n$ such that every graph on $n$ vertices has a $s$-clique and $t$-independent set.

$R(G, H)$ The graph Ramsey number: the smallest integer $n$ such that every graph on $n$ vertices contains a subgraph isomorphic to $G$ or its complement contains a subgraph isomorphic to $H$.

$\mathsf{A}(G)$ The solution found by algorithm $\mathcal{A}$ on input $G$.

$n$ The number of vertices of the graph in context.

$m$ The number of edges of the graph in context.

$N(v)$ The subgraph induced by vertices adjacent to node $v$.

# Chapter 1

# Introduction

| | |
|---|---|
| *Ár skal rísa,* | *He must rise betimes* |
| *er annars vill* | *who fain of another* |
| *fé eða fjör hafa.* | *or life or wealth would win;* |
| *Sjaldan liggjandi úlfur* | *scarce falls the prey* |
| *lær um getur* | *to sleeping wolves,* |
| *né sofandi maður sigur.* | *or to slumberers victory in strife.* |

## 1.1 Graphs, Colorings, and Independent Sets

This thesis deals with classical optimization problems on undirected, unweighted, simple graphs. Let us first review the properties of such graphs.

A graph $G$ is a pair $(V, E)$ of a set of vertices (or nodes) $V$, and a set of edges, or unordered pairs of vertices, $E$. Two vertices are *adjacent* iff the pair is contained in the edge set, and *non-adjacent* otherwise.

An *independent set* in a graph is a set of vertices that are mutually non-adjacent, and a *clique* is a set of mutually adjacent vertices. The *complement* $\overline{G}$ of a graph $G$ is a graph with the same vertex set, but an edge between two vertices iff there was no edge between them in $G$. A clique in $G$ is an independent set in $\overline{G}$, and vice versa.

A *vertex coloring* of a graph, or simply *coloring* for short, is an assignment of values to the vertices such that no two adjacent vertices are assigned the same value. Alternatively, a coloring is a partition of the vertex sets into a collection of vertex-disjoint independent sets. The dual of a coloring is a *clique cover*, which is a partition of the vertices into disjoint cliques; a clique cover of the complement of a graph implies a coloring of the original graph.

A *vertex cover* of a graph is a set of vertices with the property that every edge in the graph has one of the vertices in the vertex cover as an endpoint. Since two vertices not in a vertex cover can not be adjacent, the vertices not in a given vertex cover form an independent set and vice versa.

### 1.1.1 Optimization Problems

The abovementioned concepts lead to some of the most basic, yet still poorly understood, properties of graphs: Given a graph $G$, what is the size of the largest independent set (clique) in $G$, the size of the smallest vertex cover of $G$, and the fewest number of colors needed to partition $G$? And we may want to ask for not only a number but a representative solution as well. We define below the problems we shall encounter.

INDEPENDENCE NUMBER : What is the size of the maximum independent set in the graph?

CHROMATIC NUMBER : What is the smallest number of colors needed to color the graph?

INDEPENDENT SET : Find an independent vertex set of maximum cardinality.

GRAPH COLORING : Color the graph, with fewest number of colors possible.

VERTEX COVER : Find a vertex set covering the edges, of minimum cardinality.

All of these problems are known to be as hard to solve as any problem in $\mathcal{NP}$, the class of problems solvable in non-deterministic polynomial time. Given the amount of effort spent in attempting to find efficient solutions for those problems, it is generally considered most unlikely that those problems will have tractable solution methods, i.e. solvable in time polynomial in the length of the input.

A problem related to INDEPENDENT SET is finding a *maximal* independent set in a graph. It is defined to be an independent vertex set that cannot be extended further – the addition of any of the remaining vertices will also add an edge. This is commonly referred to as finding a *local optima*, while INDEPENDENT SET means finding a *global* optima. It is computationally much easier to solve; the trivial greedy algorithm, that we shall describe later, suffices.

### 1.1.2 Graph Notation

For an undirected graph $G = (V, E)$, $\alpha(G)$ is the *independence number* of the graph or the size of the largest independent set, $i(G)$ is the *independence ratio* or the independence number divided by the order of the graph, $cl(G)$ is the *clique number*, and $\chi(G)$ is the *chromatic number* (the number of colors needed to vertex color $G$). The size of a set $S$ is denoted by $|S|$, in particular, $|V(G)|$ is the *order* of $G$ or the number of vertices (sometimes sloppily denoted by $|G|$), $|E(G)|$ the number of edges, and $|C|$ the number of nodes in a clique $C$. For a vertex $v$, $N(v)$ refers to the subgraph induced by the neighbors of $v$, and $\overline{N}(v)$, similarly, the subgraph induced by the non-neighbors of $v$. A graph is $H$*-free* if it contains no subgraph (edge subset) isomorphic to $H$. Unless otherwise stated, $G$ is the input graph, $n$ is the order of $G$, $m$ is the number of edges in $G$, and $H$ is a fixed forbidden (not necessarily induced) subgraph.

We shall use "log" to denote base 2 logarithm, and "ln" for the natural logarithm.

## 1.2 Dealing with Intractability

Given that the graph coloring problem is $\mathcal{NP}$-complete, and has no polynomial (or even sub-exponential) time algorithms in sight, what is there to do if one must nevertheless color a graph? Or find a large independent set? There are five major types of approaches.

Perhaps the most immediate approach is JUST DO IT. If exponential time is what's needed, use exponential time, but try to minimize the constant in the exponent or minimize the average time spent for a given distribution of inputs. There are numerous candidate methods for both problems, all of which, unfortunately, run in exponential time in the worst case.

Obtaining exact solutions is not always necessary. Often, we are quite content with finding "good" approximate solutions, especially if we can find them "fast". The other three approaches all find approximate solutions, and differ in their measure of the quality of the approximations and, to a lesser extent, their emphasis on speed.

The second main strategy can be called KEEP IT FAST AND SIMPLE. There are numerous simple strategies that can find reasonably good solutions very quickly. These methods are "greedy", or semi-greedy, in that they are free of any backtracking and proceed until they find a local optima that, depending on the problem, is either a maximal independent set or a coloring of all the nodes. We shall consider some of them later, in order to evaluate their worst-case performance.

The next approach is DO IT BETTER AND BETTER. Two solutions are *neighbors* if a "simple" operation on one yields the other. Such an operation could be adding/removing a node to an independent set approximation, or swapping a group of nodes from distinct color classes. The hope is that a sequence of the neighborhood operations brings us to successively better solutions. This suggests the classical problem of finding a global optima in a space with multiple local optima. One general approach to that problem is *simulated annealing* and its variant *tabu search*. So-called Kempe-chains seem to provide the best neighborhood operations for coloring in this framework. Some variations of this successive-improvements paradigm are methods that incrementally execute an exact algorithm, maintaining the best solution at any point.

The last alternative approach that we shall mention is PROVE IT'S USUALLY GOOD, or the analytical average case behavior of approximate algorithms. Our graph problems have been analyzed extensively on *random graphs*. Such results can be of both theoretical and practical interest. The standard model of random graphs is to place an edge between each of the $\binom{n}{2}$ pairs of vertices independently with probability $p$. Whether an edge is placed between two given vertices is thus a Bernoulli trial with probability $p$, and the number of edges in the graph is a binomial distribution. $\mathcal{G}(n,p)$ denotes the space of all such graphs. A perhaps more natural method is to randomly pick one of the graphs with $M$ edges, but it can be shown to be equivalent for all practical purposes when $np = \omega(1)$ (i.e. grows larger than any constant), simply by letting $M = np$.

The main disadvantage of such results, shared with most of average case analysis, is that they hold only for certain fixed distributions. While the assumption is that those distribution are somehow the natural ones, the theory itself has nevertheless established that these graphs have certain strong properties and hence results on such graphs can be more of a statement on graphs with those properties. Another disadvantage is that the analysis is complicated and only the simplest algorithms have been analyzed successfully.

For a good overview of the above approaches to the coloring problem, see a survey of Kant and vanLeeuwen [29].

The final option, the topic of this dissertation, concerns algorithms with provably good bounds on their performance.

## 1.3 Approximation Algorithms: Inexact Methods with Bounded Performance

We are interested in tractable algorithms, namely those that run in polynomial time, that yield inexact yet "good" solutions. Our quality standard for approximate algorithms is how large a factor from optimal the solutions found can stray. The *approximation ratio* of an algorithm on a given input is defined as the ratio of the approximation size to the optimal solution size, for minimization problems, and the inverse of that, for maximization problems.

For an algorithm $\mathsf{A}$ let $\chi_A(G)$ $(\alpha_A)$ be the number of colors (size of the independent set) it uses on (finds in) input graph $G$. The approximation ratio of $\mathsf{A}$ on $G$ is thus $\chi_A(G)/\chi(G)$ $(\alpha(G)/\alpha_A(G))$, respectively.

The *performance guarantee* of an algorithm $A$, is defined as the maximum approximation ratio of A over all inputs of graphs on $n$ vertices.

We are also interested in the best performance guarantee possible for a given problem and within a given model of computation, i.e. the optimal performance guarantee of a class of algorithms. We say that a problem is $f(n)$-*approximable* within a model $M$ if

$$f(n) = \min_{A \in M} \text{Performance guarantee of } \mathsf{A}$$

The approximability of a problem is the least performance guarantee of any algorithm for the problem (within the given class of algorithms).[1] We also allow ourselves sometimes to speak of the approximability of an *algorithm*, instead of the more verbose

[1] The term "approximability" can be used identically to "complexity".

"performance guarantee".

In this thesis, we primarily focus on two classes of algorithms: $\mathcal{P}$ – polynomial-time algorithms, and $\mathcal{OL}$ – on-line algorithms. [2] We also consider some submodels and randomized variants.

## 1.4 The Relationship between Approximating Independent Set and Graph Coloring

There is a natural method of transforming any independent set method, into a coloring heuristic that we call *coloring by excavating independent sets*. It goes as follows: Find an independent set, color it with a new color, and repeat this process with the remaining uncolored vertices.

```
ExcavationColoring (G)
begin
  i ← 0
  while G ≠ ∅ do
    i ← i + 1
    I_i ← Independent Set Algorithm (G)
    G ← G − I_i
  od
  return ({I_1, I_2, ..., I_i})
end
```

Algorithm 1.1: Algorithm for approximating independent sets

This excavation procedure yields a good coloring algorithm when given an independent set algorithm with good guaranteed performance.

**Lemma 1.1** *Let* A *be an algorithm that guarantees finding independent sets of size $f(n)$ in $k$-colorable graphs of order $n$, where $f$ is a positive, non-decreasing function. Then an iterative application of* A *on a $k$-colorable graph $G$ produces a coloring of $G$ with no more than $\sum_{i=1}^{n} 1/f(i)$ colors.*

Notice in particular that if $f(n)$ grows with some power of $n$, then the number of colors used by the excavation procedure is only $\mathcal{O}(n/f(n))$.

---

[2] We hope the reader will forgive this abuse of notation; while $\mathcal{P}$ usually refers to a class of *problems*, in particular decision problems, we shall overload the term here since we are primarily interested in two problems, both of which are optimization problems.

## 1.5 On-line Algorithms

In real life one must make decision based only upon past and current events, and although one can try to extrapolate future events that can only be an educated guess rather than covering any worst-case scenario. Moreover, one can not take back the past – what happened, happened, and in general can not be erased.

This idea has been formalized as *on-line computation.* It can be thought of as a two-person game between the algorithm and the adversary. The adversary presents a sequence of requests one by one, and the algorithm must serve each request before moving on to the next and is not allowed to change his mind later on. Moreover, the algorithm does not know *a priori* the length of the request sequence.

In *on-line graph coloring* the vertices are presented one by one along with their edges to the previous nodes. The algorithm must irrevocably assign each vertex a color before seeing any further vertices or edges. Since both the algorithm and the adversary incrementally construct a coloring of the graph, we shall say that the adversary assigns *colors* to the vertices while the algorithm assigns them *bins.*

We discuss alternative formulations of the on-line independent set problem in section 4.4.

### 1.5.1 Models of On-Line Computation

Let us define the on-line graph coloring problem more formally. An adversary selects a graph and an ordering of the graph. She feeds us one node at a time along with its edges to the nodes already received. We are to irrevocably assign that node a color consistent with the colors of the previous nodes. We are allowed arbitrary amount of time but are not given any further information about the graph, neither its size nor its chromatic number.

An on-line algorithm may also use randomization. Because the adversary can often use knowledge of the workings of a deterministic algorithm to construct hard inputs, using random bits may be a way to foil the adversary. Against randomization there are several different adversaries. We follow [6] but focus on graph coloring.

**Oblivious adversary:** One who must construct the graph in advance (based only on the description of the on-line algorithm but before any nodes are colored), but pays for it optimally.

**Adaptive on-line adversary:** One who presents the next vertex based on the algorithm's bin assignment of the previous ones, but colors it immediately.[3]

**Adaptive off-line adversary:** One who presents the next vertex based on the algorithm's bin assignments to previous ones, but colors them optimally at the end.

It was proved in [6] that this adversary is so strong that randomization is of no use against it. As a result, we shall refer to it as the *deterministic adversary*. For conciseness[4], we shall refer to the adaptive on-line adversary as simply the *adaptive adversary*.

We shall be interested in a weakening in the adversary's power that is orthogonal to the classification presented so far. A *transparent* adversary is an on-line adversary that reveals her own coloring of the previous nodes. This revelation makes bin assignments easier, because it restricts the future construction of the remaining graph significantly.

**Transparent oblivious adversary:** One who must construct the graph and color it in advance, and incrementally reveal that coloring. Hence each request consists of a vertex, its edges to previous nodes, and the color assignment to the previous nodes, all constructed in advance.

**Transparent adaptive adversary:** One who presents next vertex based on the algorithm's bin assignment to date, who colors that vertex on-line based on the algorithm's bin assignment for that node, and then reveals that coloring. Hence each request consists of a vertex, its edges to previous nodes, and the color assignment to the previous nodes, all constructed adaptively on-line.

The relationships between these various adversaries can be seen in figure 1.1.

---

[3]Hence, an "on-line" adversary.

[4]Or perhaps, more accurately, terseness.

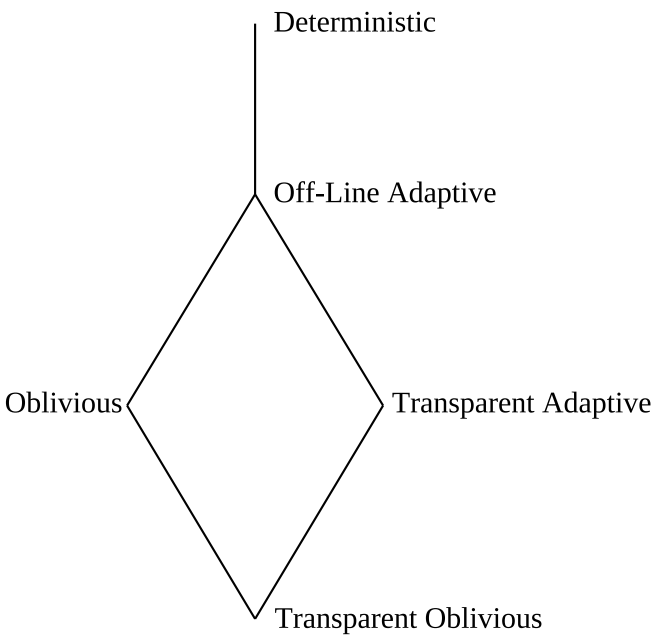


Figure 1.1: Models of on-line computation

# Chapter 2

# Related Work

| | |
|---|---|
| *Inn vari gestur,* | *Let the wary stranger* |
| *er til verðar kemur,* | *who seeks refreshment* |
| *þunnu hljóði þegir,* | *keep silent with sharpened hearing;* |
| *eyrum hlýðir,* | *with his ears let him listen,* |
| *en augum skoðar;* | *and look with his eyes;* |
| *svo nýsist fróðra hver fyrir.* | *thus each wise man spies out the way.* |

## 2.1 Analysis of Heuristics

### 2.1.1 Greedy Independent Set

The most natural heuristic for finding a maximal independent set in a graph is the simple Greedy Independent Set algorithm. Suppose we decide to place a node $v$ into a given independent set. It then suffices to search only in the non-neighborhood of $v$, $\overline{N}(v)$, for the remaining nodes in the set. This suggests a natural, greedy heuristic, whose result we can specify recursively as

$$\text{Choose } v \in V(G)$$
$$I(G) \leftarrow \{v\} \cup I(\overline{N}(v))$$

It is easy to see that this greedy method performs badly in worst case; given the star graph $K_{1,n-1}$, consisting of a single vertex adjacent to all $n-1$ independent vertices that follow, the method finds an approximation of size one from a graph with an optimal solution of size $n-1$.

On the average, however, the greedy method performs reasonably well. The following result appears widely [13, 25].

**Theorem 2.1** *With high probability, the independence number of a random graph (with edge probability 1/2) is* $2(1 + o(1)) \log n$*, while the size of the independent set found by Greedy is* $(1 + o(1)) \log n$*.*

*Proof.* The expected number of independent sets of size $r$ is $\binom{n}{r}2^{-\binom{r}{2}}$. This value is much greater than one when $r < (2 - \epsilon) \log n$ and much less than one when $r > (2 + \epsilon) \log n$, for any $\epsilon > 0$, hence the first claim.

The greedy method starts by picking a random vertex. Its degree is a binomial distribution, and hence with high probability differs only slightly from $n/2$. Hence, with high probability each step will approximately halve the number of candidate vertices, for a total of $\log n$ steps. ∎

Greedy is also useful on graphs with low maximum degree. If the degree of each vertex is at most $\Delta$, then each step in the recursion will decrease the graph by no more than $\Delta + 1$ vertices. Hence, Greedy will find an independent set of size at least $\lceil \frac{n}{\Delta+1} \rceil$.

### 2.1.2 First Fit

When the excavation scheme is applied to the Greedy Independent Set algorithm, we obtain the *Greedy Coloring* method. *First Fit* is another even greedier coloring method: sequentially assign each node the lowest available color, the lowest numbered color whose vertices are all non-adjacent to the current node. It is not too hard to see that the coloring output by Greedy also has this First Fit property, hence the two methods produce the same result. First Fit, however, has the advantage of being faster and being "single-pass", or *on-line*: the assignment to vertex $i$ depends only on the first $i - 1$ vertices, not the remaining vertices. We shall consider that property in more detail later in this chapter.

First Fit does not perform well in worst-case. On the bipartite complement of a perfect matching on $2k$ vertices it uses $k$ colors, a new color on each pair $(v_i, w_i)$ of vertices, for an approximation ratio of $n/4$. Again, it nevertheless performs reasonably well on the average. If we switch back to the Greedy Coloring method, we see that each independent set found leaves the remaining graph unexplored and thus "still random".

By counting carefully we find that the number of colors used will be no more than $n/(2\log n)(1+o(1))$ with high probability.

### 2.1.3 Minimum Degree Greedy

The greedy schema is general method that allows for many variations and possible optimizations. One parameter is the *pivoting* strategy used, or the choice of the next node. Instead of picking an arbitrary node (or the next node in some ordering of the nodes), it intuitively makes sense to choose a node of a low degree since that means that it will have more non-neighbors and thus more likely to continue for long. We can write this as.

```
Min-Degree-Greedy-IS(G)
begin
  Choose v ∈ V(G) of minimum degree
  return {v} ∪ Min-Degree-Greedy-IS(N̄(v))
end
```

This independent set algorithm appears to perform better than the basic greedy method, but has been found hard to evaluate analytically because the choice of the pivot "destroys" the randomness of the graph. Experts have claimed that it almost certainly doesn't beat the basic method asymptotically.

More importantly, it performs most poorly on the graph consisting of a clique of size $t$ and a singleton node both adjacent to an independent set of size $t$. On this graph, $G_x$, Min-Degree-Greedy-IS will choose the singleton vertex first, since it is the only vertex of degree $n/2$ or less, and for the second and the last vertex choose an arbitrary clique vertex, for an approximation ratio of $n/4$.

Similar to the basic greedy, Min-Degree-Greedy does obtain a good bound on graphs with low *average* degree. If $\overline{d}$ is the average degree of the graph, then it will find an independent set of size at least $\lceil \frac{n}{\overline{d}+1} \rceil$.

The situation is somewhat brighter for Min-Degree-Greedy-Coloring, as analyzed by Johnson in 1974 [28]. In any $k$-colorable graph there is a vertex with degree no more than $n - \lceil n/k \rceil$, since at least $n/k$ vertices must belong to the largest color class in an optimal coloring of the graph. It follows that the minimum degree vertex has a degree no higher and since any subgraph of a $k$-colorable graph is also $k$-colorable it must hold for any subsequent steps in the recursion. Each step will leave at least one out of every $k$ vertices, hence approximately $\log_k n$ [1], steps will be needed before the graph is exhausted, even when accounting for the diminishing number of candidate vertices. Hence, in any $k$-colorable graph, Min-Degree-Greedy-IS finds a $\mathcal{O}(\log_k n)$ independent set, and, by excavation, a coloring with $\mathcal{O}(n/\log_k n)$ colors. This yields a $\mathcal{O}(\min_k(n/\log_k n)) = \mathcal{O}(n/\log n)$ performance ratio.

### 2.1.4 Wigderson

Johnson's bound for Min-Degree-Greedy-Coloring remained the best performance guarantee for coloring for nearly a decade. Finally in 1983, Wigderson [49] improved the bound considerably, in particular for graphs with bounded chromatic number.

Wigderson made two crucial observations, that have remained important for further developments.

**Observation 2.1** *The neighborhood of any node in a three-colorable graph must be two-colorable.*

Since two-colorable, or bipartite, graphs can easily be colored exactly, this implies suggests a means for reducing three-coloring approximately to the tractable two-coloring. If any vertex in the three-colorable graph has degree $\sqrt{n}$ or more, then we can find an independent set of size $\sqrt{n}/2$. If no vertex has degree exceeding $\sqrt{n}-1$, the greedy method will yield an independent set of size $n/\sqrt{n} = \sqrt{n}$. By the excavation lemma, this implies a coloring of three-colorable graphs using $\mathcal{O}(n)$ colors.

This observation can be generalized to any chromatic number. It also holds that the neighborhood of any node in a $k$-colorable graph must be $(k-1)$-colorable. The hypothesis, however, turns out to be stronger (or the antecedent weaker) than necessary; the neighborhoods do not need to be $(k-1)$-chromatic even if the graph is $k$-chromatic. We can rewrite this in terms of forbidden subgraphs.

---

[1] Logarithm base $k$; $\log_k n = \log n/\log k$. Base 2 is default.

Notice that bipartite graphs are precisely the graphs free of odd cycles. The first observation thus states that the neighborhood of any node in a three-colorable graph contains no odd cycles. Alternatively, the observation states that a three colorable graph contains no subgraph consisting of a single node adjacent to all vertices of an odd cycle. This subgraph is known as an odd *wheel*. More generally, an $s$-wheel $W_{s,t}$ is a graph consisting of an odd cycle on $t$ vertices and a clique on $s$ vertices (known as *spokes*) adjacent to each cycle vertex.

**Observation 2.2** *The neighborhood of any node in a $s$-wheel-free graph is $(s-1)$-wheel free.*

In particular, a $k$-colorable graph is $(k-2)$-wheel-free.

This provides us with the means of coloring $k$-colorable graphs (more precisely $(k-2)$-wheel-free graphs) by a reduction to the problem of coloring $(k-1)$-colorable graphs. If maximum degree exceeds $n^{(k-1)/(k-2)}$ we recursively color that $(k-1)$-colorable neighborhood, otherwise we apply GreedyIndependentSet. By the inductive assumption, this yields an $\Omega(n^{1/(k-1)})$ independent set, for an $\mathcal{O}(n^{(k-1)/(k-2)})$ coloring using excavation.

Note that we can always obtain the same result as if we knew the chromatic number of the graph in advance, by simply executing the algorithm for all $n$ possible value of $k$.

Finally, we apply this algorithm for low chromatic graphs and Min-Degree-Greedy on high chromatic graphs. The performance guarantee will then be of the order

$$\max_k \min(\frac{n^{(k-2)/(k-1)}}{k}, \frac{n}{\log_k n}) = n(\log\log n/\log n)^2$$

Wigderson's method also approximates cliques [48] in somewhat roundabout fashion. If a graph has no $k+1$ clique it also has no $k$ wheel in which case Wigderson's algorithm will find a coloring of $G$ with $\mathcal{O}(kn^{(k-2)/(k-1)})$ colors. But since the clique number of a graph cannot exceed the chromatic number, the graph cannot have a clique larger than this. The approximation ratio is thus $n^{(k-2)/(k-1)}/k$, and the performance guarantee $\max_k n^{(k-2)/(k-1)}/k = \mathcal{O}(n/\log n)$. This applies immediately to the independent set problem, since the two are identical for our purposes,

### 2.1.5 Berger and Rompel

Berger and Rompel [7] recently gave an algorithm that finds independent sets of size $(\log_k n)^2$ in $k$-colorable graphs. When combined with the approximate graph coloring method of [49] (or alternatively that of section 3.2), it produces a performance guarantee of $\mathcal{O}(n(\log\log n/\log n)^3)$.

Assume we are given a $k$-colorable graph $G$. Let $A$ be a largest independent set in $G$ – thus of size at least $\lceil n/k \rceil$. For a subset $I$ of $G$, let $\overline{N}(I)$ denote the subgraph induced by vertices neither in $I$ nor adjacent to any vertex in $I$. The following three observations are a key to the argument of [7].

1. If $I$ is a subset of $A$, then $I$ must be independent of the rest of $A$, hence the order of $\overline{N}(I)$ is at least the order of the rest of $A$. More formally, $I \subseteq A \Rightarrow |\overline{N}(I)| \geq |A - I|$.

2. Consider the probability that a random set $I$ of $t$ vertices is contained in $A$. There are $\binom{n}{t}$ such sets in $G$ but at least $\binom{n/k}{t}$ such sets in $A$. Hence the probability that $I$ is a subset of $A$ is approximately at least $(1/k)^t$, which is $n^{-1}$ when $t$ is $\log_k n$. Thus in expected time $\mathcal{O}(n)$ we can find a subset of $A$ of size $\log_k n$. Since we can't easily check whether such a set actually is a subset of $A$, we shall be content with finding a set of size $\log_k n$ with the same properties as a set contained in $A$. Namely, it is independent, and has at least $|A-I| \geq \lceil n/k \rceil - \log_k n$ non-neighbors.

3. Finally, notice that once we have found an independent set $I$ with not too many neighbors, we can apply the method recursively on $\overline{N}(I)$, the non-neighbors. At each step the graph shrinks by no more than a factor of $k$, hence the recursion will be of depth at least $\log_k n$. The combined independent set will be of size at least $(\log_k n)^2$.

This process can be made deterministic, as presented in [7]. Partition the vertices of the graph into bins of size $k \log_k n$. By the pigeonhole principle, at least one of the $n/(k\log_k n)$ bins must contain a subset of $A$ of size $\frac{n/k}{n/(k\log_k n)} = \log_k n$, and we can find this subset by exhaustively examining the $\binom{k\log_k n}{\log_k n} \approx n$ such subsets in each bin. We have retained this randomized framework for simplicity of presentation.

The first step of the method of Berger and Rompel finds an independent set of size $\log_k n$ in any graph with an independent set of size $n/k$. We can thus take that result from the first iteration as an approximation of the maximum independent set. The approximation ratio is $\frac{n/k}{\log_k n}$ which is monotone decreasing with $k$, hence the performance guarantee is $n/\log n$.

We can apply this method in parallel with the one of Wigderson, and use whichever result is larger. The performance guarantee of this combined method will then be

$$\max_k \min(\frac{n^{(k-2)/(k-1)}}{k}, \frac{n}{\log_k n})$$

We find that the two functions cross at approximately $k = \frac{\log n}{2 \log\log n}$, yielding a performance guarantee of $\mathcal{O}(n(\log\log n/\log n)^2)$ for Independent Set.

### 2.1.6 Vishwanathan

Vishwanathan [47] recently published a randomized algorithm for coloring graphs on-line, using ideas of the previous work of Wigderson [49], and of Lovász, Saks, and Trotter [35]. We shall discuss on-line algorithms later in this chapter, and Vishwanathan's algorithm in a later chapter, but mention this method here because it immediately suggests a deterministic off-line algorithm. His method is a restructuring of Wigderson's algorithm that allows us to color the graph all at once, rather than excavating independent sets. We shall also be able to utilize that to create a parallel coloring algorithm.

A *partial coloring* of the graph using fixed set of colors, means assign these colors to the vertices until the remaining uncolored vertices are no longer compatible to any color. The crucial property is that for every color class, each uncolored vertex is adjacent to some vertex in the class.

The method proceeds as follows: Partially color the graph using $f(n,k)$ colors; find the color class $C$ with the fewest number of vertices; partition the uncolored vertices into subproblems according to the first node in $C$ they are adjacent to; and finally, recursively solve each subproblem, each of which is of $(k-1)$-colorable with fewer than $n/f(n,k)$ nodes.

The optimal choice of $f(n, k)$ is approximately $n^{(k-2)/(k-1)}$, which leads to a $\mathcal{O}(n^{(k-2)/(k-1)})$ coloring. We cover the details in chapter 4.

### 2.1.7 Methods for Vertex Cover

Folklore (see [24, pp. 134] attributed to Gavril) tells us that any maximal matching approximates the minimum vertex cover by a factor of two.

This was slightly improved independently by Bar-Yehuda and Even [4], and Monien and Speckenmeyer [36], to a factor of $2 - \Omega(\frac{\log\log n}{\log n})$. We rely heavily on their work in section 3.3 to graphs with very high independence number.

### 2.1.8 Blum

Blum has improved approximate coloring of graphs with fixed chromatic number, in particular for 3-coloring from the $\mathcal{O}(\sqrt{n})$ of Wigderson and the $\mathcal{O}(\sqrt{n}/\log n)$ of Berger and Rompel, to $n^{0.4+o(1)}$ [9] and later to $n^{0.375+o(1)}$ [10].

The involved method of [9] can be summarized in the following three steps ([10] adds another level of complexity to the method):

1. Destroy all copies of the subgraphs $K_4 - e$ and 1-2-3 graphs by collapsing certain pairs of nodes.

2. Classify vertices according to degree, producing a polynomial number of subgraphs, one of which has an independence ratio close to half.

3. Apply algorithm 3.4 on each of these subgraphs.

The graph $K_4 - e$ is the clique on 4 vertices with one edge removed. "1-2-3 graphs" is our terminology for the graphs with three specific parts: A, consisting of two disconnected nodes; B, an independent set of at least 3 nodes; and C, an odd cycle, where parts A and C are completely disconnected, A and B are completely connected, and the connections between B and C are such that each node in C is connected to some node in B. Since C requires three colors, B needs two, and thus the two nodes in A must have the same color under any legal 3-coloring of the subgraph, hence the name 1-2-3. Similarly, the two disjoint nodes in $K_4 - e$ must share the same color.

Steps one and three of his approach follow the subgraph-removal paradigm described in section 3.5, whereas step two uses the size of the independent sets promised by the $k$-colorability property. As a result, this method is the only one to which the lower bounds of chapter 3 do not apply.

## 2.2 Lower Bounds on Approximations

The lower bound argument that has produced the largest payoff involves *graph products.* The idea is to produce an expanded version of the input graph, one whose independence or chromatic number relates to the original graph and the expansion ratio in some favorable way. If the independence/coloring problem can be solved fairly accurately on the expanded graph, it then automatically yields a still more accurate solution on the original graph. Hence, the graph product serves to amplify the approximation, showing that certain performance guarantees can always be improved upon.

The first result, dates back to the early days of NP-completeness.

**Theorem 2.2 (Garey, Johnson [24])** *If* Chromatic Number *is approximable within less than* $4/3$ *approximable (asymptotically), then* $P = NP$.

*Proof.* Let the *composition* of two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ be the graph $G = (V, E)$ given by $V = V_1 \times V_2$, and

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

Assume that Chromatic Number is approximable asymptotically within less than $4/3$, namely there is some integer $k$ and an algorithm $\mathsf{A}$ that obtains a approximation ratio of better than $4/3$ on all graphs with chromatic number at least $k$.

Consider a graph $G$, an arbitrary instance of the 3-Chromaticity problem, and let $G^* = K_k[G]$, where $K_k$ is the complete graph on $k$ vertices. Observe that $G^*$ is just $k$ isomorphic copies of $G$, with every two vertices in different copies being joined by an edge, so $\chi(G^*) = k \cdot \chi(G) \geq k^2$.

However, if $G$ is 3-colorable, then $\mathsf{A}(G) < (4/3) \cdot \chi(G^*) \leq (4/3) \cdot 3K = 4K$, while if $G$ is 4-chromatic, then $\mathsf{A}(G) \geq \chi(G^*) \geq 4K$. Hence, we have obtained a decision procedure if an arbitrary graph is 3-colorable, that runs in polynomial time since $k$ is independent of $G$ and thus the size of $G^*$ are polynomial in $G$. ∎

A similar proof technique, but this time using hypergraphs or multicolorings, was used later by Garey and Johnson to show that the constant in the theorem could be improved to 2. That remains the best lower bound today.

Also, the composition operator can be used to show that the best performance guarantee for the independent set problem is not a fixed constant.

**Theorem 2.3 ([24])** *If* Independent Set *can be approximated within $a$, then it can be approximated within $b$, for any reals $a, b > 1$.*

*Proof.* Let $N$ be the least integer such that $a^{1/N} < b$. Repeatedly apply the composition operator on $G$ to obtain a new graph $G_N$, defined inductively by $G_1 = G$ and $G_i = G_{i-1}[G]$. Since $N$ depends only on $a$ and $b$, $G_N$ can be constructed in polynomial time.

Notice that $\alpha(G_N) = \alpha(G)^N$, and that given an independent set $S_N$ in $G_N$ we can find an independent set $S$ in $G$ satisfying $|S| \geq \lceil |S_N|^{1/N} \rceil$ in polynomial time.

Hence, if we can approximate $G_N$ within a factor of $a$, i.e. find an independent set $S_N$ in $G_N$ satisfying $|S_N| \geq \alpha(G_N)/a = \alpha(G)^N/a$, we have obtained an independent set $S$ in $G$ of size $|S| \geq (\alpha(G)^N/a)^{1/N} \geq \alpha(G)/a^{1/N} \geq \alpha(G)/b$. ∎

Notice, that the theorem holds also for Independence Number.

Finally, using a slightly different version of graph product, Linial and Vazirani [34] showed that if the chromatic number can be approximated within $2^{\log^{1-\epsilon} n}$, then it can be approximated within $2^{\log^{\delta} n}$, for any $\epsilon, \delta > 0$. The result does not extend to actually finding a coloring.

### 2.2.1 Classes of Approximability

One of the main strengths of $\mathcal{NP}$-completeness results is that if one $\mathcal{NP}$-hard problem has a polynomial time solution, then all of them do. Unfortunately, these reduction do not carry over in terms of the approximabilities of the problems. In fact, $\mathcal{NP}$-complete problems vary greatly in their ease of approximation. Knowing that a problem is $\mathcal{NP}$-hard/easy tells us therefore nothing about its approximability.

One means of countering this problem is restricting the reductions used in order to preserve approximability. Several researchers have given varying definitions of *approximation-preserving reductions* [38, 18, 40, 42, 16], but the common thread in most of them is the restriction that a solution for one problem that differs by $\epsilon$ from optimal that is reduced to a solution to a second problem, the latter will be no more than a constant times $\epsilon$ factor from optimal. [2]

Papadimitriou and Yannakakis [40] used these reduction techniques to introduce a class of $\mathcal{NP}$ optimization problems that all have a constant approximation and have a polynomial-time approximation scheme iff some problem in the class has it. The definition of the class is based on the logical definition of $\mathcal{NP}$ by Fagin [21]. He showed that $\mathcal{NP}$ consists of all all *existential second-order formulas* which are expressions of the form $\exists S \phi(S)$, where $S$ is a finite structure and $\phi(S)$ is a first-order formula. It turns out that every such formula is equivalent to one of the form $\exists S \forall \overline{x} \exists \overline{y} \psi(\overline{x}, \overline{y}, S)$, with $\psi$ quantifier-free.

They defined the class MAX NP to consist of all $\mathcal{NP}$ maximization problems whose optimum can be expressed as

$$\max_S |\{\overline{x} : \exists \overline{y} \psi(\overline{x}, \overline{y}, S)\}|$$

where $\psi$ is quantifier free. They also defined a subclass, named MAX SNP, consisting of those problems in MAX NP that do not need the existential quantifier in the expression. The key property of both classes is that all problems in both classes must have have constant approximations. They gave many example of problems complete for MAX SNP under an approximation preserving reduction they named *L-reduction*, in particular, MAX 2SAT, INDEPENDENT SET on bounded-degree graphs, and VERTEX COVER on bounded-degree graphs.

---

[2]We are, on the other hand, often equally content with reductions that add a factor that grows moderately with input size.

This was built on further by Panconesi and Ranjan [39] introduced the class MAX $\Pi_1$, defined like MAX NP but with a universal instead of existential quantifier. They showed that problems complete for that class (under a approximation preserving reduction) can not be approximated within a constant factor (unless $\mathcal{P} = \mathcal{NP}$). They investigated a subclass, where the occurrences of the structure $S$ in the formula $\psi$ expressed in CNF all occur negatively. The class RMAX(k) contains problems where $S$ occurs no more than $k$ times, all negatively. In particular, INDEPENDENT SET is in RMAX(2).

Kolaitis and Thakur [32] have analyzed this hierarchy of problems even further, shown that there is exactly one higher level, MAX $\Pi_2$, and that the four levels are strict. They give a similar classification and hierarchy for minimization problems. In particular, they define the subclass MIN F$\Pi_1$(k), corresponding to RMAX(k), whose complete problems are all constant approximable and are considered unlikely to have a polynomial-time approximation scheme. Surprisingly, they show that, except for MAX NP and MAX SNP, all classes cut arbitrarily through approximative properties, in that the classes low in the hierarchy contain some hard-to-approximate problems, while classes high in the hierarchy do not contain even problems in $\mathcal{P}$.

### 2.2.2 Randomized Amplification

We follow the excellent analysis and enhancements of Avrim Blum [11] of results of Berman and Schnitger [8].

The graph products mentioned in the beginning allow us to amplify the independent set in the graph to obtain stronger approximations. The limitation of the approach is that the amplification process can be repeated no more than constant number of times, in order for the size of resulting graph product to remain polynomial bounded.

One means of getting around that limitation is to avoid creating the full resulting graph product, but only a randomly chosen representative sample.

Given a graph $G$ and three parameters $r$,$p$, and $k$, we produce a new graph $H$ as follows. The vertices of $H$ are $m = n^{rp+2}$ randomly chosen subsets of $V(G)$, each of size $r \log_k n$. Denote these subsets in $V(G)$ as $s_1, s_2, \ldots, s_m$, and the corresponding vertices in $V(H)$ as $w_1, w_2, \ldots, w_m$. Two vertices in $H$ are adjacent iff the union of the subsets of vertices in $G$ they represent is not independent. That is, $(w_i, w_j) \in E(H)$ iff $(s_i \cup s_j) \cap E(G) = \emptyset$.

We can map sets of vertices from one graph to the other in a natural manner. Let $\varphi(s_i) = w_i$, and $\varphi^{-1}(w_i) = s_i$. Also, for $S \subseteq V(G)$, define $\varphi(S) = \{\varphi(s_i) | s_i \subseteq S\}$, and for $T \subseteq V(H)$, $\varphi^{-1}(T) = \{v | v \in s_i \text{ for some } \varphi(s_i) \in T\}$. The important fact is that the functions map independent sets to independent sets.

**Fact 2.1 (Blum)** *If $S$ is an independent set in $G$, then $\varphi(S)$ is an independent set in $H$. If $T$ is an independent set in $H$ of size at least 2, then $\varphi^{-1}(T)$ is independent in $G$.*

*Proof.* If $w_i, w_j \in \varphi(S)$, then $s_i, s_j \subseteq S$. So if the edge $(w_i, w_j)$ is in $H$, then $s_i \cup s_j$ is a non-independent subset of $S$. If $T$ is independent in $H$ of size greater than 1, then for each $w_i \in T$, the set $s_i$ must be independent in $G$. So, $\varphi^{-1}(T)$ is a union of independent sets that are pairwise independent of each other, and thus is independent itself. ∎

The key property about this graph product is that it amplifies the difference in size of independent sets. If we have an independent set $I$ of size $n/k$ in graph $G$, then since each $s_i$ has probability about $(\frac{1}{k})^{r \log_k n}$ of being chosen inside $I$, the size of $\varphi(I)$ in $H$ is $\theta(n^{rp+2}(\frac{1}{k})^{r \log_k n}) = c_1 n^{r(p-1)+2}$, with high probability. However, the expected size of $\varphi(I')$ for $I'$ an independent set of size $n/k^p$ is only $c_2 n^2$. In fact, with high probability, $\varphi(I')$ will be small for *all* such $I'$.

Hence, the problem of distinguishing between graphs with an $n/k$ independent set and only $n/k^p$ independent sets can be reduced to distinguishing between $N/n^r$ and $N/(n^r)^p$ independent sets. Allowing ourselves a minor sloppiness, we can say that $H$, the graph product result, is the result of $\varphi$ on the original graph $G$, and vice versa.

**Theorem 2.4 (Blum)** *Suppose there exists a (randomized) algorithm* $\mathsf{A}$ *for* Independent Set *on $N$-vertex graphs that runs in time $t(N)$ and has performance guarantee at most $\frac{1}{8}N^{(1-\frac{r+2}{pr+2})}$. Then, on graphs $G$ containing an $n/k$ independent set, $\varphi^{-1}(\mathsf{A}(\varphi(G)))$ is an independent set of size $n/k^p$ with high probability, obtained in time $t(n^{pr+2}) + \mathcal{O}(n^{rp+O(1)})$. (Where $r \log_k n \ll n/k^p$).*

*Proof.* $\mathsf{A}(\varphi(G))$ is an independent set in $H$ of size at least

$$\frac{c_1 n^{rp-r+2}}{c_2[n^{rp+2}]^{(1-\frac{r+2}{rp+2})}} = c_3\, n^2$$

It follows that $\varphi^{-1}(\mathsf{A}(\varphi(G)))$ must be of size at least $n/k^p$, with high probability. ∎

If we view this theorem in the contrapositive form and letting $p = 1+\epsilon$ and $\frac{r}{r+2} = 1-\delta$, we get the following corollary.

**Corollary 2.1 (Blum)** *If for some $k$ and some $\epsilon > 0$ there is no randomized polynomial-time algorithm which finds an independent set of size $n/k^{1+\epsilon}$ on $n$-vertex graphs containing an independent set of size $n/k$, then: for any $\delta > 0$, the general* Independent Set *problem is not approximable within $o(n^{\frac{\epsilon}{1+\epsilon}(1-\delta)})$.*

Now we can use these results to relate the hardness of approximating Independent Set to the approximability of a number of related problems: $k$-Coloring, $\mathcal{O}(\log n)$-Coloring, Vertex Cover, and $n/k$-Independent Set.

**Theorem 2.5** *Suppose there exists a (randomized) polynomial-time algorithm $\mathsf{A}$ for* Independent Set *with performance guarantee $n^{1-\epsilon}$ for some $\epsilon > 0$. Then,*

1. *$n/k$-*Independent Set *is $k^c = \mathcal{O}(1)$ approximable (where $c \leq 3/\epsilon - 3$)*
2. *([11]) $k$-*Coloring *is $\mathcal{O}(\log n)$ approximable.*
3. *([11]) $\mathcal{O}(\log n)$-*Coloring *is $\mathcal{O}(\log^c n)$ approximable (where $c \leq 1 + 3/\epsilon$)*

*Proof.* For $p > \frac{3-2\epsilon}{\epsilon}$, algorithm $\mathsf{A}$ has performance guarantee $o(n^{1-\frac{3}{p+2}})$. So by theorem 2.4, we can find an independent set of size $n/k^p$ in graphs with $n/k$ independent sets, for an approximation of $k^{p-1} = k^{\frac{3}{\epsilon}-3}$.

The latter two claims follow automatically when coloring by excavation. ∎

**Theorem 2.6** *If there does not exist a randomized polynomial-time approximation scheme for* Vertex Cover, *then* Independent Set *is not approximable within $n^c$, for some $c > 0$. In particular, if* Vertex Cover *can not be approximated (randomized) within $(1+\epsilon)$ then* Independent Set *can not be approximated within $o(n^{\frac{\epsilon}{1+\epsilon}}(1-\delta))$, for all $\delta > 0$.*

*Proof.* If Vertex Cover is not $(1+\epsilon)$-approximable, then for some $k$, no (randomized) polynomial-time algorithm can guarantee a vertex cover of size $\frac{(k-1)n}{k}(1+\epsilon)$ in graphs with minimum vertex cover of $\frac{(k-1)n}{k}$ [3]. It follows that no algorithm can find an independent set of size $n - \frac{(k-1)n}{k}(1+\epsilon) = \frac{n}{k}(1-\epsilon(k-1))$ in graphs with maximum independent set of size $\frac{n}{k}$. We claim that the size of such a set is at most $\frac{n}{k^{1+\epsilon}}$, hence, by the previous corollary, the claim follows.

To see why this inequality holds, note first that the vertex cover can always be approximated within $\frac{k}{k-1}$ using the whole vertex set as the approximation, hence it suffices to consider only values of $k$ for which $1+\epsilon \leq 1 + \frac{1}{k-1}$, or $k \leq 1 + \frac{1}{\epsilon}$. If we now denote $k$ by $1+t$, we have that $\epsilon t \leq 1$ (and clearly $t > 0$). Also, recall that $\log(1+x) \leq x$ for all $x \geq -1$. Hence, we have that $\log(1-\epsilon t) \leq -\epsilon t$.

Since $1-\epsilon(k-1) = 1/k^{\log_k 1/(1-\epsilon(k-1))}$, and $\log_k \frac{1}{1-\epsilon(k-1)} = \frac{-\log(1-\epsilon t)}{\log(1+t)} \geq \frac{-(-\epsilon)t}{t} = \epsilon$, we have that $\frac{n}{k}(1-\epsilon(k-1)) \leq \frac{n}{k^{1+\epsilon}}$. ∎

### 2.2.3 Cliques and Interactive Proofs

Feige, Goldwasser, Lovász, Safra, and Szegedy [22] have recently shown an intimate connection between accepting *interactive proofs* and approximating Independent Set. We shall attempt to give a rough outline of those results.

Interactive proofs are languages that all-powerful "verifier" can check by interacting with a number of polynomial-time "provers". An alternate definition of interactive proofs, and one more applicable to the results we consider, is the class of languages accepted by a probabilistic machine equipped a memoryless oracle, with bounded, one-way-erring probability.

Let the *transcript* of a probabilistic machine with an oracle, given input $x$ of length $n$, denote the random bits used, along with the sequence of query-answer pairs, possibly involving different oracles. Consider all accepting transcripts computation of such a machine on a given input $x$, and call two transcripts *consistent* if same queries are

---

[3] This does not immediately follow. It is plausible that each value of $k$ could be covered by some algorithm while no algorithm covers all values. However, it suffices to look at only a constant (in terms of $\epsilon$) number of values of $k$, with only a small roundoff error.

answered the same way.

Construct a graph $G_x$, where the nodes are all accepting transcripts, and two nodes are adjacent iff they are not consistent. The number of such nodes, and the time for generating the graph, is exponential in the size of the transcript, which is $r_M(n)+c_M(n)$, where $r_M(n) = r(n)$ denotes the number of random bits used by the probabilistic machine and $c_M(n) = c(n)$ denotes the number of communication bits used.

A key result of the authors is an improvement of the original protocol of [3], showing that $r_M(n) + c_M(n) \leq \log(T(n)) \cdot \log\log T(n)$ is sufficient transcript length to accept all language in $NTIME(T(n))$ (for $T(n) \geq n$).

It can be shown that the independence number of the graph is equivalent to $2^{r(n)}$ times the maximum probability, over all oracles, of $M$ accepting. Since the acceptance probability is bounded away, by a constant factor, say 4, it follows that if $x \in L$ then $\alpha(G_x) = 2^{r(n)}$ and if $x \notin L$ then $\alpha(G_x) < 2^{r(n)}/4$.

Hence, if there is an algorithm that can distinguish between graphs of size $2^{r(n)+c(n)}$ with $\alpha(G_x)$ being $2^{r(n)}$ as opposed to $2^{r(n)}/4$, then such an algorithm would be powerful enough to accept all interactive proofs, and hence all languages in $NEXPTIME$, by a result of Babai, Fortnow, and Lund [3].

The graph products approach of Garey and Johnson then tells us that approximating within any constant factor is sufficient.

**Theorem 2.7** *If* Independence Number *is constant factor approximable, then for any $T(n) \geq n$, $NTIME(T(n)) \subset DTIME(T(n)^{O(\log\log T(n))})$.*

The approximation factor can be pumped up by running the probabilistic machine many times on the input, bounding the error probability by a decreasing function. In particular, run $M$ $l = \log^{(1-\epsilon)/\epsilon} T(n)$ times on input $x$ and accept $x$ if $M$ accepts $x$ on all $l$ iterations. By this choice of $l$, the number of random and communication bits used by $M'$ is still poly-logarithmic in $T(n)$. For $x \notin L$, the probability of accepting has been decreased to $1/2^l$. Thus, the difference between the size of the maximum independent set in $G'_x$ when $x \in L$ and when $x \notin L$ is a factor of $2^{\log^{1-\epsilon} T(n)}$ of $|G'_x|$.

Hence,

**Theorem 2.8** *If* INDEPENDENCE NUMBER *is* $2^{\log^{1-\epsilon} n}$ *approximable in quasi-polynomial time, for some* $\epsilon > 0$*, then for any* $T(n) \geq n$*,* $NTIME(T(n)) \subset DTIME(2^{(\log T(n))^k})$.

In particular, the conclusion would imply that $\mathcal{NP}$ is contained in quasi-polynomial (i.e. $2^{\log^k n}$) time and non-deterministic exponential time is equivalent to deterministic exponential time.

## 2.3 On-line Algorithms

The on-line requirement is a restrictive constraint for an algorithm (or an algorithm designer). The method must be "greedy", in the sense of not being allowed to change any decisions, it has no visibility, no information about the remainder of the graph, and it has no choice about which node is looked at next.

Of the heuristics covered so far, only the simplest one, First Fit, is on-line. Although it performs badly in worst case on general graphs, it turns out to be the highly versatile and has good performance on many special classes of graphs.

First Fit obtains constant approximability on interval graphs [30], split graphs, complements of bipartite graphs, and complements of chordal graphs [26]. It also approximates degenerate (also called inductive) graphs, which include planar and chordal graphs, within $\mathcal{O}(\log n)$ [27], which is within a constant of best possible for on-line algorithms. That lower bound holds also for on-line algorithms that are allowed significant *lookahead*, seeing into the near future. Also, First Fit performs as well as any on-line algorithm on trees.

In the case of interval graphs, Kierstead and Trotter [31] gave a more involved algorithm that obtained 3-approximability, and showed that was the best possible.

It has been shown by several authors [26, 5] that for any algorithm there are graphs that require unbounded number of colors. What is more striking is that there is a fixed tree on $n$ vertices such that any on-line algorithm will use at least $1 + \lfloor \log n \rfloor$ colors given some ordering of the tree. This lower bound was cited in [35], while a weaker argument was given in [26]. We shall describe the construction here and show that First-Fit attains this bound.

**Theorem 2.9** *For every deterministic on-line coloring algorithm, there is a tree on $2^{k-1}$ nodes that requires $k$ colors.*

*Proof.* Fix a deterministic algorithm $A$. For every $k$ we construct a tree $T_k$ of size $2^{k-1}$ for which the $A$ uses $k$ colors. $T_1$ is just a single node, and for $k > 1$, take $T_1, T_2, \ldots, T_{k-1}$ and root them so that the colors of the $k-1$ roots are distinct. Now introduce a new node that is adjacent to the roots of the $k-1$ trees and forces the $k$-th color to be used. The size of $T_k$ is thus $1 + \sum_{i=1}^{k-1} T_i = 2^{k-1}$. ∎

Notice that in the above argument, the adversary used a fixed graph $T_k$ varying only the ordering.

**Theorem 2.10** *If First-Fit uses $k$ colors on a tree, its size must be at least $2^{k-1}$.*

*Proof.* Proof by induction. The claim immediately holds for $k = 1$ and $k = 2$. If a vertex is colored with the $k$-th color, it must have been adjacent to nodes assigned all the lower valued colors. Moreover, the nodes must have been in disjoint subtrees, or the resulting graph would no longer by acyclic. By induction, the sizes of those subtrees must be at least $2^0, 2^1, \ldots, 2^{k-2}$, hence the combined tree is of size one larger, or at least $2^{k-1}$. ∎

We had seen before that First Fit performed very badly on the more general class of bipartite graphs. There is, however, a simple variation of First Fit that uses no more than $2 \log n$ colors, matching the lower bound for trees within a constant. This was suggested in [35] as an "easy exercise", and also by Viswanathan [47].

Let us first see why First-Fit doesn't perform on bipartite graphs. Consider the complete bipartite graph $K_{t,t}$ and subtract from it a perfect matching. Feed the algorithm the graph one matching at a time. First-Fit will color both of those nodes with the same color, which it will then not be able to use ever again on the graph.

Consider the following algorithm: Use the first available color, unless it would duplicate a color that was only in the opposite partition, in which case, use a new color. A key property of this method is that new colors are added only if components of equal number of colors are merged together.

**Theorem 2.11** *The above algorithm uses at most $2k$ colors on every bipartite graphs on at most $2^{k+1} - 2$ nodes.*

*Proof.* The important observation is that connectivity makes an optimal coloring unique. Consider a component of the subgraph seen so far. Each partition has at most color bin that doesn't appear in the other partition.

Assume the current node is adjacent to two previous components, making it into one. The colors of one component must to be a subset of the other's, hence if one has two more colors than the other, no new colors are required. Only if the components have the same set of colors, the previously "free" colors may now appear in both partitions, and two new colors will be needed. A component for which the algorithm uses $2k$ colors must therefore be built from two components requiring $2(k-1)$ colors and two additional vertices. The bound follows by induction on $k$. ∎

We note without proof that this bound is the tightest possible, unless $n$ or $k$ are fixed in advance.

**Theorem 2.12** *For all $k$, there is a graph on $2^{k+1} - 3$ nodes for which every algorithm uses at least $2k - 1$ colors. Moreover, for every algorithm, if for some $k$ and a graph on $2^{k+1} - 2$ nodes the algorithm uses fewer than $2k$ colors, then for some $k'$ the algorithm uses $2k'$ colors on some supergraph on fewer than $2^{k'+1} - 2$ nodes.*

### 2.3.1 Performance Guarantees

It was an interesting open question for some time whether any on-line algorithm could achieve less than linear performance guarantee. It was answered affirmatively by Lovász, Saks, and Trotter [35]. Their algorithm has a performance guarantee of $\mathcal{O}(n/\log^* n)$. For a fixed $k$, their method uses $\mathcal{O}(n \log^{(2k-3)} n / \log^{(2k-4)} n)$ colors, and, in particular, $n \log\log\log n / \log\log n$ colors on 3-colorable graphs. [4]

The algorithm partitions the vertices into greedy bins and subproblems, similar to Vishwanathan's sequential algorithm given earlier, except that it cannot choose the

[4] Mistakenly stated as $n/\log\log n$ in their paper, as pointed out by Viswanathan [46].

smallest bin after the fact but must work with the nodes as they arrive. A node incompatible with the greedy bins is placed in a residue bins if it has a large intersection with the pre-neighborhoods of the vertices in the bin. That insures that the number of residue bins doesn't grow out of bounds. Using $n/\log\log n$ greedy bins, they show that no more than $n/4\log\log n$ residue sets are created. A coloring algorithm follows by recursively coloring the residue sets, which by Wigderson's observation are of a lesser chromaticity.

The above argument assumed that $n$, the number of vertices, was known in advance. That does not hold in general for on-line algorithm, but can easily be sidestepped using the *doubling trick*. We assume the graph is of size $2^t$, with $t$ initially being 1. As soon as that assumption is exceeded, we increment $t$, allocate new set of greedy bins, and continue as if we were starting from scratch. Because of the convexity of the function that specifies the number of greedy bins, this doubling method adds no more than a constant factor of overhead.

Vishwanathan [47] recently gave a randomized on-line algorithm, using similar ideas, that colors $k$-colorable graphs with $\mathcal{O}(k2^k n^{(k-2)/(k-1)}(\log n)^{1/(k-1)})$ expected number of colors, yielding a $\mathcal{O}(n/\sqrt{\log n})$ performance guarantee. We give improvements of that method in section 4.3, that removes the $k2^k$ factor and reduces the performance guarantee to $\mathcal{O}(n/\log n)$.

As for lower bounds, for graphs with bounded chromatic number, Vishwanathan [47] improved previous bounds of Alon [2], and Szegedy and Vishwanathan to $\Omega(\frac{1}{\chi-1}(\frac{\log n}{12(\chi+1)})^{\chi-1})$. Vishwanathan's result holds also for randomized algorithms against oblivious adversary. This is still very far from the upper bounds cited.

# Chapter 3

# Polynomial-time Approximations

| | |
|---|---|
| *Gráðugur halur,* | *A gluttonous man* |
| *nema geðs viti,* | *who guzzles away* |
| *etur sér aldurtrega;* | *brings sorrow on himself;* |
| *oft fær hlægis,* | *At the table of the wise* |
| *er með horskum kemur,* | *he is taunted often,* |
| *manni heimskum magi.* | *mocked for his bloated belly.* |

## 3.1 Our Contributions

We present several algorithms that improve the best bounds known for approximating the independent set and graph coloring problems and some of their variants, and prove that the approaches used to date to obtain such results can not be improved much further.

We start with the *Ramsey* algorithm for finding independent sets in graphs, which is actually a schema of algorithms parameterized by the pivoting function applied, similar to the greedy algorithm. It has the property that its solution on any graph is at least as large as the one the corresponding greedy algorithm finds. More importantly, for our purposes, we show that given a graph containing no clique of size $k$, it will find an independent set of size at least $kn^{1/(k-1)}$. We use that to show that given graphs with independence number exceeding $n/k$, we can find an independent set approximation of size $\Omega(kn^{1/(k-1)})$ by removing cliques from the graph. This implies a $\mathcal{O}(n/\log^2 n)$ performance guarantee for INDEPENDENT SET, the first such bound published. This also improves slightly on Wigderson's algorithm [49].

We also give some algorithms for graph with very high independence number, and show how known results on approximating VERTEX COVER follow from it.

We show how to use the above INDEPENDENT SET performance guarantee to improve a result of Berger and Rompel [7], obtaining the best performance guarantee known for GRAPH COLORING of $\mathcal{O}(n(\log\log n)^2/\log^3 n)$.

We finally consider the question how much further these approaches can be taken. We present lower bounds for a class of algorithms for GRAPH COLORING, INDEPENDENT SET, and VERTEX COVER that match or almost match the upper bounds known. The class of algorithm considered includes all algorithms known today with non-trivial performance guarantees.

## 3.2 The Ramsey algorithm

The greedy scheme is a generally useful and practical means of finding an independent set of approximate size. However, as the worst case examples show, it has the inconvenient property that it completely ignores nodes neighboring the pivot nodes, even if that neighborhood was large and fully independent.

We are led to a new rule for searching for an independent set. As in the greedy method, choose a vertex and search in the non-neighborhood of that node. But this time also search in the neighborhood of the pivot node, and use whichever result is bigger. Again, a dual rule applies to the cliques. More formally,

$$\begin{aligned}
&\text{Choose } v \in V(G)\\
&I(G) \leftarrow \max(\{v\} \cup I(\overline{N}(v)), I(N(v)))\\
&C(G) \leftarrow \max(\{v\} \cup C(N(v)), C(\overline{N}(v)))
\end{aligned}$$

The resulting algorithm is shown in figure 3.1.

```
Ramsey (G)
begin
  if G = ∅ then return (∅, ∅)
  choose some v ∈ G
  (C1, I1) ← Ramsey(N(v))
  (C2, I2) ← Ramsey(N̄(v))
  return (larger of (C1 ∪ {v}, C2), larger of (I1, I2 ∪ {v}))
end
```

Algorithm 3.1: The Ramsey algorithm

If we look at the behavior of the algorithm, we see that it breaks the problem into a tree-like structure of subproblems. In one sense, the algorithm transforms the graph into a binary tree where each internal node is adjacent to all of its left descendants and non-adjacent to all of its right descendants. Under this interpretation, the independent set found by the algorithm is intimately related to a path in the tree with the largest number of right edges. Specifically, it consists of the leaf, and the parents of the right edges in that path. Hence, the size of the independent set found is exactly the maximum number of right edges in any path in the tree, plus one. Similarly, the size of the clique found is the maximum number of left edges in any path, plus one.

As an example, assume the input graph $G$ contains no triangles. Clearly, the algorithm cannot find any cliques of size more than 2, hence no path in the tree can have more than a single left edge. It follows that either the rightmost path has $\sqrt{n}$ nodes, or there are fewer than $\sqrt{n}$ paths in the tree, in which case one of them has more than $\sqrt{n}$ nodes. Either way, the algorithm finds an independent set of size no less than $\sqrt{n}$.

This formulation gives us an effective way of expressing the sizes of the approximations. A computation of the algorithm that produces a clique of size $s$ and an independent set of size $t$ corresponds to a binary tree where the largest number of left edges in a path is $s-1$ and the largest number of right edges is $t-1$. Let $r(s,t)$ denote the smallest integer $n$ such that all trees of size $n$ have paths with at least that many left or right edges. This value is one larger than the size of the largest tree with no path having $s-1$ left edges or $t-1$ right edges, which again is one less than the number of external nodes in that tree. Since each external node has an associated unique path, there can be no more than $\binom{(s-1)+(t-1)}{(t-1)}$ such nodes. Hence, $r(s,t) \leq \binom{s+t-2}{t-1}$.

The next theorem follows from the preceding discussion.

**Theorem 3.1** *The algorithm* Ramsey *finds an independent set $I$ and a clique $C$ such that $r(|I|,|C|) \geq n$.*

The algorithm Ramsey is related to a classical problem in extremal graph theory. Let $R(s,t)$ denote the smallest integer $n$ such that all graphs of order $n$ either contain an independent set of size $t$ or a clique of size $s$. This function was named after the

English mathematician Frank P. Ramsey who first showed that it was well-defined. Our algorithm in his name, and the associated analysis, provides another proof to an upper bound for the Ramsey function, first proved by Erdős and Szekeres in 1934 [20].

**Theorem 3.2** $R(s,t) \le r(s,t) \le \binom{s+t-2}{s-1}$

Define $t_s(n) = \min\{t \mid r(s,t) \ge n\} = \min\{t \mid \binom{s+t-2}{t-1} \ge n\}$. Note that if the graph contains no clique of size $k+1$, the independent set found must be of size at least $t_k(n)$. As $k$-colorable graphs are a subset of $(k+1)$-clique free graphs, the same bound holds for them. We can approximate $t_k(n)$ fairly accurately by $kn^{1/(k-1)}$, for $k \le 2\log n$, and $\log n / \log \frac{k}{\log n}$, for $k \ge 2\log n$.

Notice also, that the product of $|C|$ and $t_{|C|}(n)$ is minimized when they are equal, at which point each exceeds $2\log n$. Hence,

**Corollary 3.1** Ramsey *finds an independent set* $I$ *and a clique* $C$ *such that* $|I| \cdot |C| \ge \frac{1}{4}(\log n)^2$.

When carefully implemented, algorithm 3.1 involves $\mathcal{O}(n+m)$ work. Select pivot nodes according to a lexicographic-first rule. Maintain a list or array to index the vertices in the current subgraph, and divide the list into two, representing the neighborhood and non-neighborhood of the pivot node, before making the recursive call. If care is taken to conquer the smaller subproblem first, only linear (in $n$) extra space is required.

For small values of $k$, we are able to improve on Ramsey slightly. Boppana [14] has shown that a technique by Ajtai, Komlós, and Szemerédi [1] treated by Shearer [41] as a *randomized* greedy algorithm, can be made deterministic to find an independent set in $k$-clique-free graphs of size $\Omega(n^{1/(k-1)}(\log n)^{(k-2)/(k-1)})$ in polynomial time.

## Wheels

A *wheel*, denoted by $W_{p,q}$, is a graph that consists of an odd cycle of $q \ge 3$ nodes, and $p \ge 0$ *spokes*, which are nodes that connect to all other nodes in the graph. A wheel with $p$ spokes is referred to as a $p$-wheel. The clique number of an $p$-wheel is $p+2$ (except when $q=3$), whereas the chromatic number is $p+3$.

Note that if a graph does not contain a $p$-wheel, then its neighborhood graph cannot contain a $(p-1)$-wheel nor can its non-neighborhood graph contain a $p$-wheel. Hence we obtain the same recursive relationship as in the Ramsey algorithm. Only the base case is different; we capitalize on the fact that coloring a bipartite graph is easily solvable in linear time.

**WheelFreeRamsey** $(G)$
begin
  if ($G$ is bipartite) then return ($\emptyset$, the larger color set)
  choose some $v \in G$
  $(W_1, I_1) \leftarrow$ WheelFreeRamsey($N(v)$)
  $(W_2, I_2) \leftarrow$ WheelFreeRamsey($\overline{N}(v)$)
  return (larger of ($W_1 \cup \{v\}, W_2$), larger of ($I_1, I_2 \cup \{v\}$))
end

Algorithm 3.2: Ramsey algorithm for wheels

Define $R(W_p, K_t)$ to be the minimal $n$ such that all graphs of order $n$ contain some $p$-wheel or an independent set of size $t$. We find that $R(W_p, K_t) \leq R(W_{p-1}, K_t) + R(W_p, K_{t-1})$ and $R(W_0, K_t) = 2t-1$ and $R(W_p, K_2) = p+3$. An inductive argument shows that $R(W_p, K_t) \leq 2\binom{p+t}{t-1}$, only a factor of two from the upper bound of the regular Ramsey function.

Given a graph with no $(k-2)$-wheels, WheelFreeRamsey finds an independent set of size at least $\Omega(kn^{1/(k-1)})$. By applying a version of algorithm 3.3 that utilizes WheelFreeRamsey, we can color a graph without $(k-2)$-wheels using $\mathcal{O}(n^{(k-2)/(k-1)}/k)$ colors.

Algorithm 3.2 is strongly related to Wigderson's [49] coloring algorithm. By considering the whole uncolored portion of the graph in each iteration, instead of fully coloring the pivot nodes' neighborhoods before coloring their non-neighbors, WheelFreeRamsey improves the approximation by a factor of $k$. Also, by focusing alternately on neighborhoods and non-neighborhoods, another factor of $k$ is gained. Wigderson's method, however, has the advantage of $\mathcal{O}(\chi(G)\ (n+m))$ time complexity, compared to the $\mathcal{O}(\text{COLORS}\ (n+m))$ complexity of our method. Compared with the graph coloring algorithm deduced from the Ramsey algorithm for clique-free graphs, this algorithm improves the exponent from $\frac{k-1}{k}$ to $\frac{k-2}{k-1}$.

### Performance guarantee

We have seen that if the graph contains no large cliques, then Ramsey performs quite well. Unfortunately, if that precondition does not hold, we cannot make any statement about its performance. Nevertheless, if we could somehow get rid of these large cliques, we could do well on the remaining graph.

We are led to a simple method:

> Remove a maximal set of disjoint $k$-cliques from $G$, for some constant $k$.
>
> Apply Ramsey to the remaining graph.

The first concern is whether anything will be left of the graph once we have removed all vertices in disjoint $k$-cliques. For an arbitrary graph, no, but if the graph contains a large enough independent set, the remaining graph will be sizable. A key observation is that a clique and an independent set can share no more than a single vertex. If the independence number of the graph is at least $(1/k + \epsilon)n$ for some constant $\epsilon > 0$, then at least a fraction $\epsilon/(1 - \frac{1}{k})$ of the vertices remain.

The second problem is that finding a $k$-clique in the graph requires $n^{\theta(k)}$ operations for all algorithms known, hence the above algorithm is not fully polynomial in both $n$ and $k$. However, we need not remove cliques that we do not run into, only those that get in our way. It suffices to remove the cliques as we go along. Recall that Ramsey finds both a clique and an independent set approximation. If the clique that Ramsey finds is small, then the independent set must be large, while if the clique is large, then we can remove it and repeat the process. This is formalized in algorithm 3.3.

CliqueRemoval repeatedly calls Ramsey and removes the clique found until the graph is exhausted. It then returns the largest of the independent sets found along with the sequence of cliques found. Since that collection is a partition of the vertex set into cliques, it forms an approximation to the Clique Cover problem. Moreover, if the algorithm is applied to the complement of the graph, we obtain approximations to the Clique and the Graph Coloring problems.

We can now prove tight bounds on the sizes of the approximations.

```
CliqueRemoval (G)
begin
  i ← 1
  (C_i, I_i) ← Ramsey (G)
  while G ≠ ∅ do
    G ← G − C_i
    i ← i + 1
    (C_i, I_i) ← Ramsey (G)
  od
  return ((max_{j=1}^{i} I_j), {C_1, C_2, ..., C_i})
end
```

Algorithm 3.3: Algorithm for approximating independent sets

**Theorem 3.3** *Given a graph $G$, let $k$ be the smallest integer such that $\alpha(G) > n/k$, and let $\epsilon = \alpha(G)/n - 1/k$. Define $t_s(n)$ as before.*

*The algorithm* CliqueRemoval *finds an independent set approximation $I$, and a clique cover approximation $CC$, such that*

$$|I| \geq \max(t_k(\epsilon n), t_{k+1}(\tfrac{n}{k^2})) \quad \textit{and} \quad |CC| \leq \tfrac{5n}{(\log n)^2}|I|$$

*Proof.* Let us first consider the first claim. Since the independence fraction of the graph is strictly greater than $1/k$, the algorithm must eventually find no $k$-clique, at which point $(\epsilon/(1-1/k))n \geq \epsilon n$ vertices remain in the graph. Also, the point when the algorithm finds no $(k+1)$-clique occurs even earlier, when at least $(\epsilon + 1/k - 1/(k+1)) \cdot (1/(1-1/k)) \cdot n \geq n/k^2$ vertices remain in the graph. The bound then follows from theorem 3.1.

For the second claim, we shall, for pure convenience, analyze the approximations guaranteed for the Clique and Graph Coloring problems, with the understanding that the same applies immediately to the Independent Set and Clique Cover problems, respectively. Let $C$ be the clique approximation, and $\{I_1, \ldots, I_{\text{Colors}}\}$ be the coloring approximation.

Recall that the approximations produced by Ramsey satisfy $|C_i| \cdot |I_i| \geq (\frac{1}{2}\log|G_i|)^2$. Hence, if $f(N)$ represents the value of $|I_i|$ when $|G_i| = N$, then $f(N) \geq \frac{1}{4}(\log N)^2/|C_i| \geq \frac{1}{4}(\log N)^2/|C|$. Applying the excavation lemma 1.1, we get that $\text{Colors} \leq \sum_{i=1}^{n} 4|C|/(\log i)^2 \leq 5n|C|/(\log n)^2$. ∎

Now consider the product of the two performance guarantees.

$$\frac{cl(G)}{|C|}\frac{\textsc{Colors}}{\chi(G)} \leq \frac{5n}{(\log n)^2}\frac{cl(G)}{\chi(G)}$$

Since $cl(G)$ is never greater than $\chi(G)$, this bound immediately yields the claimed $\mathcal{O}(n/(\log n)^2)$ individual bounds on the performance guarantees. For classes of instances for which the measures are apart, the performance guarantees are even stronger. In particular, random graphs almost always have a clique number asymptotically $2\log n$ and chromatic number $n/(2\log n)$, and for graphs with these parameters the product of the performance guarantees is a constant (no more than 20). This also implies that the stated relationship between the sizes of the two approximations is optimal within a constant. [1]

The above approximation and performance guarantee for the independent set (and by duality the clique) problem are the best known. The approximation for graph coloring is also the best known for graphs with chromatic number between $\sqrt{\frac{\log n}{\log\log n}}$ and $\frac{\log n}{\log\log n}$. For graphs with a smaller chromatic number the method of Blum [10] performs best, while for larger chromatic numbers the result in section 3.4 is stronger.

The time complexity of the algorithm is $\mathcal{O}(\textsc{Colors} \cdot (n+m))$.

## 3.3 Graphs with a High Independence Number

For graphs with independence ratio in the range of $(\frac{1}{3}+\epsilon, \frac{1}{2})$, the Ramsey algorithm obtains an independent set approximation of $\Omega(\sqrt{n})$ by removing triangles. We can refine the approximations in this range by removing other subgraphs that form an odd cycle.

The method starts by removing all odd cycles of size up to $2k+1$. In contrast to the case for cliques, this can be done in linear time independent of $k$. Note that a cycle of length $2k+1$ has an independence ratio $\frac{k}{2k+1}$. So if $i(G) > \frac{k}{2k+1}$, we can remove these cycles and then apply algorithm 3.4.

[1] When read with an algorithmic frame of mind, this relationship and the resulting performance guarantees can be found in a 1967 paper of Erdős [19].

```
OddCycleFreeApproximation (G, k)
 { Graph G contains no odd cycles of length 2k + 1 or shorter}
begin
  while G ≠ ∅ do
    choose any vertex v in V(G).
    V_i ← vertices of distance i from v.
    S_i ← V_i ∪ V_{i-2} ∪ ...
    Determine i such that |S_{i+1}| ≤ n^{1/(k+1)} |S_i|.
    I ← I ∪ S_i
    G ← G − S_i − S_{i+1}
  od
  return I
end
```

Algorithm 3.4: Algorithm for independent sets in graphs with no short odd cycles

Since each independent set $S_i$ selected causes only $n^{1/(k+1)}$ times as many other nodes to be removed from the graph, the graph is not exhausted until an independent set of at least $n^{k/(k+1)}$ has been collected. Assume there was no $i$ satisfying $|S_{i+1}| \leq n^{1/(k+1)}|S_i|$. Then $|S_k| > n^{1/(k+1)}|S_{k-1}| > n^{2/(k+1)}|S_{k-2}| > \cdots > n^{k/(k+1)}|S_0| = n^{k/(k+1)}$, and the problem is solved.

Since each vertex and each edge are looked at only once, the algorithm runs in linear time. On the other hand, when applied to general graphs the algorithm must be run for many different values of $k$, in which case it may be useful to combine the cycle removal process (see [36]).

Boppana's [14] improvement of the technique of Ajtai, Komlós, and Szemerédi can also be applied here. When $k$ is fixed, we can find an independent set of size $\Omega(n^{k/(k+1)}(\log n)^{1/(k+1)})$ in polynomial time for graphs with no odd cycles of length $2k+1$ or less.

### 3.3.1 Vertex Cover Approximation

While vertex cover approximation algorithms can not be applied immediately to the general independent set problem, they do apply to finding independent sets in graph with a very high independence number. We shall convert the algorithm of Monien and Speckenmeyer for that purpose. Their implementation for vertex cover required a use of a combinatorial result of Nemhauser and Trotter [37], which is not needed for the independent set algorithm.

Following Monien and Speckenmeyer [36], we first apply a technique of Nemhauser and Trotter that separates the graph into three components: the first containing nodes not in any minimum vertex cover, the second containing nodes in every minimum vertex cover, and the third with the property that its minimum vertex cover is at least half its order. We then apply our algorithm on the last component.

The algorithm can also be applied to approximate the minimum vertex cover problem. Assume we have removed all cycles from $G$ in sequence, with $2p+1$ being the length of the longest one. The remainder of the graph is bipartite, for which we can approximate the independent set exactly, hence we can ignore it without loss of generality. Then $\alpha(G) \leq \frac{p}{2p+1}$. Let $k \leq p$ such that $\frac{k+1}{2k+3} \geq \alpha(G) \geq \frac{k}{2k+1}$. Then the order of a minimum vertex cover is at least $\frac{k+2}{2k+3}n$. Since the above algorithm obtains an $n^{k/(k+1)}$ approximation to $\alpha(G)$, the remainder of the graph, of order $n(1-1/n^{1/(k+1)})$, approximates the vertex cover. Hence the approximation ratio is no more than $\frac{(2k+3)(1-1/n^{1/(k+1)})}{k+2} \approx 2 - 1/(k+1) - 2/n^{1/(k+1)}k+2$. One of the negative terms is strictly decreasing and the other one strictly increasing, meeting at $k = \frac{\log n}{2\log\log n}$, yielding a performance guarantee of $2 - \Omega(\frac{\log\log n}{\log n})$.

### 3.3.2 Bootstrapping

CliqueRemoval finds an independent set at least as large as Ramsey finds by removing the cliques it finds. It can equally well be used to find a large clique by removing the independent sets Ramsey finds. The resulting clique would again be at least as large as Ramsey finds. What if we now used the excavation-like method of CliqueRemoval on the output of itself on the graph and its complement?

This bootstrapping technique could be continued as long as we are willing to spend time, or until near fixed point. A simple argument shows that the limit of the approximation on a graph $G$ approaches the same bound as CliqueRemoval attains on graphs with independence ratio $\frac{\alpha(G)}{n-cl(G)}$.

It may be instructive to consider the class of *split* graphs with independence and clique numbers of $n/2$. Ramsey promises $\log(n/2) = \log n - 1$ sized approximation of

```
BootstrappedCliqueRemoval (G)
begin
  i ← 1
  I_i ← CliqueRemoval (G)
  C_i ← CliqueRemoval (Ḡ)
  while G ≠ ∅ do
    G ← G − C_i
    i ← i + 1
    I_i ← CliqueRemoval (G)
    C_i ← CliqueRemoval (Ḡ)
  od
  return ((max_{j=1}^{i} I_j), {C_1, C_2, ..., C_i})
end
```

Algorithm 3.5: Bootstrapped independent set algorithm

both measures, and thus CliqueRemoval finds ones of size approximately $n/(2\log n)$. A second level attains $n/2 - n/(2\log n)$, and a third level comes even closer.

## 3.4 Degree-Based Coloring Algorithms

We now show how to use the independent set approximation algorithms to improve the result of Berger and Rompel [7] described earlier.

Let $G$ be a $k$-colorable graph, let $A$ be a largest (or larger than average) color class (under some optimal coloring) in $G$, and let $I$ be a set of $\log_k n$ vertices in $G$. Let $k_{thres}$ denote $\log n/2\log\log n$, and let $n_{thres}$ be $n \cdot k_{thres}/k$. We shall introduce the term *independence quotient* of a graph as referring to the ceiling of the ratio of the order of a graph to the size of its maximum independent set.

The CliqueRemoval algorithm of section 3.2 finds an approximation of size at least $t_q(N) = \min\{t : \binom{t+q-2}{q-1} \geq N/q^2\}$, on graphs of order $N$ and independence quotient $q$.

We can now state our observation more formally as:

If $I \subseteq A$ then either

*i)* $|\overline{N}(I)| \geq n_{thres}$ (in which case we can recurse on a large subgraph), or,

*ii)* the independence quotient of $\overline{N}(I)$ is small (in which case the algorithm CliqueRemoval produces a large approximation to the independent set in $\overline{N}(I)$.)

How small is "small" and how large is "large"? If $I$ is in $A$ and $|\overline{N}(I)| \leq n_{thres}$, then the independence quotient of $\overline{N}(I)$ is no more than approximately $n_{thres}/(n/k) = k_{thres} = \log n / 2 \log\log n$. Then $\mathsf{CliqueRemoval}$ will find an independent set of size $\Omega((\log n)^3 / \log\log n)$, which for our purposes is ample.

These observations are codified in algorithm 3.6.

```
SampleIS (G,k)
{ G is k-colorable, |G| = n }
begin
  if |G| ≤ 1 then return G
  if (k ≤ k_thres) then return CliqueRemoval(G)
  forever do
    randomly pick a set I of log_k n nodes
    if I is independent then
      if |N̄(I)| ≥ n_thres then
        return (I ∪ SampleIS(N̄(I),k))
      else
        I_2 = CliqueRemoval(N̄(I))
        if |I_2| ≥ t_{k_thres+1}(|N̄(I)|) then return (I ∪ I_2)
        { else I ⊈ A }
      endif
    endif
  od
end
```

Algorithm 3.6: Algorithm for finding independent sets in $k$-colorable graphs

**Theorem 3.4** *Given a $k$-colorable graph $G$,* SampleIS*$(G,k)$ returns an independent set of size* $\log_k n \log n / (2 \max(\log(k/k_{thres}), 1))$ *in expected polynomial time.*

*Proof.* Proving the claim reduces to showing that under that assumption that $I$ is in $A$, if $|\overline{N}(I)|$ is smaller than the threshold value $n_{thres}$, then $I \cup \mathsf{SampleIS}(\overline{N}(I), k)$ is of the specified size, and otherwise $I \cup \mathsf{CliqueRemoval}(\overline{N}(I))$ is.

Assume the non-neighborhood is "small", in which case the independence quotient $q$ is no more than $\lceil |\overline{N}(I)|/|A| \rceil \leq \lceil k \cdot |\overline{N}(I)|/n \rceil \leq \lceil k_{thres} \rceil \leq k_{thres}+1$. $\mathsf{CliqueRemoval}(\overline{N}(I))$ will then produce an independent set of size at least:

$$\begin{aligned}
t_q(N) &\geq \min\{t : \tbinom{t+q-2}{q-1} \geq n/(qk)\} \\
&\geq \min\{t : \tbinom{t+k_{thres}-1}{k_{thres}} \geq n/k^2\} \\
&\geq (1/4)k_{thres}(n/k^2)^{1/k_{thres}} \\
&= (1/4)(\log n)^3/(2\log\log n\ k^{4\log\log n/\log n}) \\
&\geq (1/16)(\log n)^3/\log\log n \qquad\qquad \text{when } k \leq n^{1/(4\log\log n)}
\end{aligned}$$

Since $\log n/16\log\log n$ grows asymptotically faster than $1/(2\log k)$, the claim holds for sufficiently large $n$.

Assume the non-neighborhood is "large". The claim holds trivially for $|G| = 1$, hence inductively assume the claim holds for all graphs of size less than $n$. Then the recursive call will produce an independent set of size $(\log|\overline{N}(I)|)^2/(2\log k\max(\ell,1)) \geq (\log n - \ell)^2/(2\log k\max(\ell,1)) \geq (\log n)^2/(2\log k\max(\ell,1)) - \log n/\log k + \ell/\log k$, where $\ell$ denotes $\log(k/k_{thres})$. Now add the size of $I$, $\log_k n$, and the claim holds. ∎

Note that the only significant addition to the deterministic algorithm of Berger and Rompel is a call to the CliqueRemoval algorithm in the innermost loop. That increases the time complexity by a factor of less than $\mathcal{O}(m)$.

**Corollary 3.2** SampleIS *induces an $\mathcal{O}(n(\log\log n)^2/(\log n)^3)$ performance guarantee for graph coloring.*

*Proof.* First notice that we can run SampleIS for all values of $k$, obtaining an approximation at least as good as that of SampleIS($G$,$\chi(G)$), where $\chi(G)$ denotes the chromatic number of $G$.

By the lemma above, SampleIS($G$,$\chi(G)$) will produce a coloring of $G$ with $\mathcal{O}(n\log\chi(G)\max(\log(k/k_{thres}),1)/(\log n)^2)$ colors when $\chi(G) \geq k_{thres}$, and $\mathcal{O}(n^{1-1/\chi(G)}/\chi(G))$ colors when $\chi(G) \leq k_{thres}$ (from its call to CliqueRemoval). The approximation ratio, which is the number of colors used divided by the number of colors needed, is monotone increasing for $\chi(G) \leq k_{thres}$, and monotone decreasing for $\chi(G) \geq k_{thres}$. Thus it suffices to observe that at the point of discontinuity, $\chi(G) = k_{thres}$, both functions satisfy the claim. ∎

## 3.5 Limitations of Subgraph-Exclusion Algorithms

We will present lower bounds for a fixed class of algorithms, similar in spirit to the work of Chvátal [17] [2]. We show how most approximation algorithms for the above-mentioned problems revolve around the concept of *excluding subgraphs*, and how no algorithm within that framework can do significantly better than the algorithms presented here. The techniques used have a strong connection with graph Ramsey theory and the Ramsey-theoretic results may be of independent interest.

Let us formally define a framework that properly encaptures nearly all known algorithms with non-trivial performance guarantees.

**Definition 3.1** *An algorithm with an associated fixed graph $H$ and function $f$, is a* Ramsey-type *algorithm if, for every $H$-free graph $G$, it guarantees finding an independent set of size at least $f(|G|)$.*

**Definition 3.2** *An algorithm is a* subgraph-exclusion *algorithm if, given arbitrary graph $G$, it is of the form:*

1. *Ensure that $G$ contains no copy of the subgraph $H$, and*
2. *Apply a Ramsey-type algorithm on $G$.*

There are a few ways in which such an algorithm can exclude a subgraph $H$:

**Remove:** All copies of the forbidden subgraph, or parts of it, can be pulled out of the graph sequentially. A necessary and sufficient precondition for the removal process to retain at least a constant fraction of the vertices is that $i(H) < i(G) + \epsilon$, for some constant $\epsilon > 0$.

**Forbid:** The exclusion of the subgraph can be built into the statement of the problem. This applies particularly to the graph coloring problem. For instance, the clique on $k+1$ vertices cannot appear in $k$-colorable graphs.

---

[2] Chvátal proved an exponential lower bound on the length of certain types of proofs of a lower bound of the independence number. The number of steps in the proof is intimately tied with the time complexity of an algorithm executing the proof, and thus the lower bound carried over to a class of algorithm that includes the Independent Set algorithm of Tarjan and Trojanowski [44].

**Merge:** In certain cases, vertices can be fused together, causing a certain type of a subgraph to disappear.

The issue becomes finding graphs $H$ that force graphs free of $H$ to contain large independent sets, as well as coming up with algorithms to actually find those independent sets in $H$-free graphs. We have earlier described several such algorithms. Section 3.2 described algorithms that use cliques, and wheels, and the algorithm in section 3.3 used odd cycles. Also, the method of Blum [9, 10], described in chapter 2, uses certain color-critical subgraphs. In this section we shall illustrate that these subgraphs are in some sense the best of their kind.

The main result of this section is that excluding subgraphs other than cliques and series of odd cycles does not help much in forcing a graph to contain a large independent set. This implies that no subgraph removal algorithms, even super-polynomial ones, can yield asymptotically better performance guarantees for the maximum independent set, graph coloring, and vertex cover problem than the algorithms given.

### 3.5.1 Limitation Results for Cliques and Color-Critical Graphs

Let us extend the Ramsey function from cliques to arbitrary graphs. Let $R(H, K_t)$ denote the minimal $n$ such that every graph on $n$ vertices either contains a copy of the graph $H$ or has an independent set of size $t$. Note that $H$ does not need to be isomorphic to a *vertex induced* subgraph of $G$, only that all the edges of $H$ be contained in such a subgraph. It immediately follows that $R(H, K_t) \geq R(H', K_t)$ whenever $H'$ is an edge-subset of $H$. Obtaining an upper bound on $R(H, K_t)$ shows that not all $H$-free graphs contain all that large independent sets, showing a limitation on the power of excluding $H$.

A few definitions are in order. For a graph $H$, let $n(H)$ be the number of vertices, $m(H)$ be the number of edges, and $\rho(H)$ denote the maximum of $m(H')/n(H')$ over all subgraphs $H'$ of $H$. Extend these definitions to a collection $\mathcal{H}$ of graphs. Define $i(\mathcal{H})$ to be the maximum of $i(H)$ over all $H$ in $\mathcal{H}$. Define $\rho(\mathcal{H})$ and $\chi(\mathcal{H})$ to be the minimum of $\rho(H)$ and $\chi(H)$, respectively, over all $H$ in $\mathcal{H}$. Also, $R(\mathcal{H}, K_t)$ is the minimal $n$ such that every graph on $n$ vertices either contains a copy of some $H$ in $\mathcal{H}$ or has an independent set of size $t$.

There are some well-known relations between these quantities. One relation is $\chi(H)i(H) \geq 1$, which holds because a coloring is just a partition into independent sets. Another relation is $\chi(H) \leq 2\rho(H) + 1$, which holds because $H$ has a vertex of degree $2\rho(H)$ or less. Both relations generalize to a collection of graphs.

We will give the central theorem for a function slightly stronger than $\rho$. Define $\rho'(H) = \min \frac{m(H')-1}{n(H')-2}$ where $H'$ is a subgraph of $H$ on at least 3 vertices. Similarly extend it to a collection of graphs $\mathcal{H}$. The value of $\rho'$ is always at least as large as $\rho$, and for small graphs the improvement makes a difference.

**Theorem 3.5** $R(\mathcal{H}, K_t) = \Omega((\frac{t}{\log t})^{\rho'(\mathcal{H})})$

*Proof.* The proof follows the probabilistic method using the Lovász local lemma. We follow closely the presentation of Bollobás [12, pp.287] of a lower bound on the ordinary Ramsey numbers. We give a proof only for a singleton collection $\mathcal{H} = \{H\}$; the general case is similar.

Let $l = m(H)$ and $s = n(H)$, and let $r = \frac{l-1}{s-2}$. We claim that $R(H, K_t) \geq c(\frac{t}{\log t})^r$, for $c = 1/(4(4r)^r \binom{s}{2}^{1/(s-2)})$.

Find positive numbers $a$, $b$, and $\epsilon$ such that $0 < \epsilon < 1/4$, $a > 2(r + b - 1)$, and $b > (1+\epsilon)^2 a^l c^{s-2} \binom{s}{2}$. Such a choice is possible since if we take $\epsilon = 0$, and replace the inequalities above by equalities, the solutions for $a$ and $b$ are positive.

Consider $G$ in $\mathcal{G}(n, p)$, a random graph on $n$ vertices with edge probability $p$, with $n = c(t/\log t)^r$ and $p = a(\log t)/t$. Let $U$ be the space of all vertex subsets of size $s$, and $W$ the space of all $t$-sets. Let $A_S$ be the event that a given instance $S$ of $U$ contains the forbidden subgraph $H$, and let $B_T$ be the event that a given instance $T$ of $W$ is independent. The $A_S$'s all have the same probability, which we denote by $p_A$, and similarly $p_B$ denotes the probability of each $B_T$.

Consider the graph on $U \cup W$ in which two vertices are joined by an edge iff the corresponding subsets of $V$ have at least two vertices in common. This is precisely the graph of dependencies among the events $\{A_i : i \in U\} \cup \{B_j : j \in W\}$. Let $d_A$ be the number of events in $U$ intersecting with $A_S$, $d_B$ the number of events in $W$ intersecting with $B_T$, $d_{AB}$ the number of events in $W$ intersecting with $A_S$, and $d_{BA}$ the number of events in $W$ intersecting with $B_T$.

We have that

$$p_A \leq p^l s! = (\frac{a\log t}{t})^l s!$$

$$p_B = (1-p)^{\binom{t}{2}} \leq e^{-p\binom{t}{2}} = t^{-a(t-1)/2}$$

$$d_A \leq \binom{s}{2}\binom{n}{s-2} \leq 3n^{s-2} \leq 3c^{s-2}(\frac{t}{\log t})^{l-1}$$

$$d_B \leq \binom{n}{t} < (\frac{en}{t})^t$$

$$d_{AB} \leq \binom{n}{t} < (\frac{en}{t})^t$$

$$d_{BA} \leq \binom{t}{2}\binom{n}{s-2} < \frac{t^2}{2(s-2)!}n^{s-2} = \frac{t^2}{2(s-2)!}c^{s-2}(\frac{t}{\log t})^{l-1}$$

Bollobás derived the following version of a theorem of Lovász, for dependence graphs with two kinds of events.

**Fact 3.1** *If there are $\delta_A$, $\delta_B$ such that $1 < \delta_A < 1/(2p_A)$, $1 < \delta_B < 1/(2p_B)$,*

$$\log \delta_A \geq (1+\delta_A p_A)(d_A p_A)\delta_A + (1+\delta_B p_B)(d_{AB} p_B)\delta_B$$

*and*

$$\log \delta_B \geq (1+\delta_A p_A)(d_{BA} p_A)\delta_A + (1+\delta_B p_B)(d_B p_B)\delta_B$$

*then*

$$Pr\left[\bigcap_{i\in U} \overline{A}_i \cap \bigcap_{j\in W} \overline{B}_j\right] > 0$$

Our main claim will follow if we can find values for $\delta_A$ and $\delta_B$ that satisfy the conditions in the above fact. We claim that $\delta_A = 1+\epsilon$ and $\delta_B = t^{bt}$ will suffice, provided $t$ is sufficiently large. We find that

$$d_A p_A \leq 3c^{s-2}(\frac{t}{\log t})^{l-1} \cdot (\frac{a\log t}{t})^l s! = \mathcal{O}(\frac{\log t}{t}) = o(1)$$

and

$$d_{AB} p_B \delta_B \leq (\frac{en}{t})^t \cdot t^{-a(t-1)/2} \cdot t^{bt} = t^{t((l-1)/(s-2)-a/2+b-1)+o(t)} = o(1)$$

since by the bound on $a$, the exponent of $t$ is negative if $t$ is large enough. Hence, the first condition of the fact holds.

By the last inequality, $d_B p_B \delta_B = o(1)$. Furthermore,

$$\begin{aligned} \delta_A p_A d_{BA} &= (1+\epsilon)[a(\log t)/t]^l s! \frac{t^2}{2(s-2)!} c^{s-2} (t/(\log t))^{l-1} \\ &= (1+\epsilon) a^l c^{s-2} \binom{s}{2} (t \log t) \quad < \quad (1+\epsilon)^{-1} \log \delta_B \end{aligned}$$

because of the constraint on $b$. Since $\delta_A p_A = o(1)$ this implies the second condition, if $t$ is sufficiently large.

We have shown that $\Pr[G$ contains an $H$ or $\overline{K_t}] < 1$ and thus there exists a graph on $n$ vertices that contains no independent set of size $t$, nor a subgraph isomorphic to $H$. Hence, the Ramsey number $R(H, K_t)$ must be larger than $n = c(t/\log t)^r$.

Finally, since the above argument applies as well to any subgraph of $H$, in particular, $H'$ such that $\frac{m(H')-1}{n(H')-2} = \rho'(H)$, the fact that $R(H, K_t) \geq R(H', K_t)$ allows us to improve the exponent in the value of $n$ from $r = \frac{m(H)-1}{n(H)-2}$ to $\rho'(H) = \max_{H' \in H} \frac{m(H')-1}{n(H')-2}$. ∎

Recall that Blum's algorithm made use of subgraphs that contain two nodes that must be of the same color under any legal 3-coloring. A graph is *$k$-avoidable* iff it has a pair of vertices that get assigned the same color for every $k$-coloring of the graph. Note that this is vacuously true for non-$k$-colorable graphs. Alternatively, $k$-avoidable graphs can be characterized as being no more than one edge away from being $(k+1)$-chromatic. A collection $\mathcal{H}$ is $k$-avoidable iff every $H$ in $\mathcal{H}$ is.

**Corollary 3.3** *For every positive integer $k$, if $\mathcal{H}$ is $k$-avoidable, then $R(\mathcal{H}, K_t) = \Omega((\frac{t}{\log t})^{k/2})$.*

*Proof.* If $H \in \mathcal{H}$ is $k$-avoidable, then $H + e$ is $(k+1)$-chromatic for some edge $e$. Hence $\rho(H+e) \geq \frac{k}{2}$. But $\rho'(H) \geq \min_{H' \in H+e} \frac{m(H')-2}{n(H')-2} \geq \rho(H+e)$, when $\rho(H+e) \geq 1$. This holds for all $H$ in $\mathcal{H}$, hence the conclusion follows from theorem 3.5. ∎

This result implies that a Ramsey-type algorithm on a $k$-colorable graph that relies solely on the lack of some set of $k$-avoidable subgraphs cannot guarantee finding an independent set of size more than $\mathcal{O}(n^{2/k} \log n)$, and hence cannot guarantee a coloring with less than $\Omega(n^{1-2/k}/\log n)$ colors. As an example, no such algorithm can guarantee coloring a 3-colorable graph with less than $\Omega(n^{1/3}/\log n)$ colors.

We can make a stronger statement regarding the 3-coloring problem.

**Theorem 3.6** *If $\mathcal{H}$ is 3-avoidable, then $\rho'(\mathcal{H}) \geq \frac{3}{2} + \frac{1}{26}$.*

*Proof.* Let $H$ be a 3-avoidable graph in $\mathcal{H}$, and $H + e$ be 4-chromatic. A *4-critical* graph is a 4-chromatic graph with the property that removing any node will make it 3-colorable. Gallai [23] showed that 4-critical graphs, with the exception of $K_4$, have an edge-to-vertex ratio of at least $\frac{3}{2} + \frac{1}{26}$. If $H + e$ contains a $K_4$, then $\rho'(H) \geq \rho'(K_4 - e) = \frac{5-1}{4-2} = 2$. Otherwise, $H + e$ contains a 4-critical subgraph $H^*$, and $\rho(H + e) \geq \frac{m(H^*)}{n(H^*)} \geq \frac{3}{2} + \frac{1}{26}$ by Gallai's result. In either case, $\rho'(H) \geq \frac{3}{2} + \frac{1}{26}$ for any $H$ in $\mathcal{H}$. ∎

As a result, Ramsey-type algorithms require at least $\Omega(n^{1-1/(\frac{3}{2}+\frac{1}{26})}/\log n) = \Omega(n^{.35}/\log n)$ colors on 3-colorable graphs. Notice that Blum's technique also breaks down in the region of $n^{1/3}$ [10], even though it is not known to be of a subgraph-excluding type.

Let us now derive a limitation for general graphs. It can be shown, in the spirit of the bounds on the diagonal Ramsey function $R(s, s)$, that if $\mathcal{H}$ is a $t$-avoidable collection then $R(\mathcal{H}, K_t) = 2^{\Omega(t)}$. Hence if all graphs of order $n$ contain either a subgraph $H$ in the $t$-chromatic collection $\mathcal{H}$ or an independent set of size $t$, then $t$ must be $\mathcal{O}(\log n)$. Hence no Ramsey-type algorithm that relies solely on the lack of avoidable subgraphs can obtain a better than $\Omega(n/(\log n)^2)$ performance guarantee for graph coloring.

Our emphasis so far on graph coloring is because the lower bounds for graph coloring are also lower bounds for the independent set problem. Since $i(H) < \frac{1}{k}$ implies that $\chi(H) \geq k + 1$, corollary 3.3 holds as well for graphs with large independence ratio. Similarly, the limitation result on performance guarantees for the general coloring problem carries immediately over to the maximum independent set problem.

### 3.5.2 Limitation Results for Odd Cycles

Our next goal is to show that our cycle-based algorithm is close to optimal for graphs with independence ratio near $\frac{1}{2}$.

We need the following structural result on graphs without short odd cycles.

**Theorem 3.7** *For every positive integer $k$, if $\rho(H) \leq 1 + \frac{1}{4k+2}$, and $H$ contains no odd cycles of length $2k-1$ or less, then $i(H) \geq \frac{k}{2k+1}$.*

*Proof.* By induction on the number of vertices in $H$. If there is a vertex $v$ of degree 0 or 1, then remove $v$ and its neighbor from the graph. By induction, the remaining graph has independence ratio at least $\frac{k}{2k+1}$. But adding $v$ to the largest independent set of the remaining graph shows that $H$ itself has independence ratio greater than $\frac{k}{2k+1}$.

Thus, assume that every vertex has degree 2 or more. Suppose there is a cycle passing through only vertices of degree exactly 2. Since $H$ has no odd cycles of length $2k-1$ or less, the independence ratio of this cycle is at least $\frac{k}{2k+1}$. Then we could apply induction to the remainder of the graph and be finished.

Thus, assume there are no such cycles. Let $H'$ be the subgraph induced by the vertices of degree exactly 2. The subgraph $H'$ must be the disjoint union of paths, so $i(H') \geq \frac{1}{2}$. Since $\rho(H) \leq 1 + \frac{1}{4k+2}$, the subgraph $H'$ contains at least a fraction $1 - \frac{1}{2k+1}$ of all the vertices. Therefore $i(H)$ is at least $\frac{1}{2}(1 - \frac{1}{2k+1}) = \frac{k}{2k+1}$, which completes the proof. ∎

Finally, we can prove the following limitation result.

**Corollary 3.4** *For every positive integer $k$, if $i(\mathcal{H}) < \frac{k}{2k+1}$, then*
*$R(\mathcal{H}, K_t) = \Omega(\frac{t}{\log t})^{1+1/(4k+2)})$.*

*Proof.* By theorem 3.7 above, every $H$ in $\mathcal{H}$ either has an odd cycle of length $2k-1$ or less, or satisfies $\rho(H) \geq 1 + \frac{1}{4k+2}$. The first case implies that $\rho'(H) \geq \frac{2k-2}{2k-3} = 1 + \frac{1}{2k-3}$, thus in either case $\rho'(H) \geq 1 + \frac{1}{4k+2}$. ∎

This result implies that for a graph with independence ratio $\frac{k}{2k+1}$, no subgraph-removal algorithm can guarantee an independent set larger than $\mathcal{O}(n^{1-1/(4k+3)})$. Recall that our cycle-based algorithm will find an independent set of size $\Omega(n^{1-1/k})$ and thus the cycle-based algorithm is close to optimal.

The above result also implies that for approximately solving the vertex cover problem, no subgraph-removal algorithm can achieve a performance guarantee better than

$2 - \theta(\frac{\log\log n}{\log n})$, the performance guarantee obtained by the algorithm of Monien and Speckenmeyer.

### 3.5.3 Further Limitation Results for Coloring

A limitation of the limitation results for coloring algorithms above is that they obviously do not cover the degree-based algorithms, such as Berger/Rompel's technique [7]. It turns out, however, that it is easy to argue only a slightly weaker limitation for algorithms equipped with that additional technique.

The "degree-based" technique is of the form: find a large independent set with a small non-neighborhood, and recursively apply the technique on that non-neighborhood. The additional power of the technique comes from this recursive application. It can yield a gain of as much as a $\log n$ factor when the non-neighborhood is a constant fraction of the total size of the graph; a larger gain is however impossible unless better independent set finding algorithm come into play. Finding a large independent set, without the requirement of a small non-neighborhood, is not easy, and for the class of subgraph-exclusion algorithm we have a strong lower bound. The combination of the two techniques into a recursive application of a subgraph-exclusion independent set algorithm can then do no better than $\log n$ factor better than a basic subgraph-exclusion method, hence a lower bound of $\Omega(n/\log^3 n)$ easily follows.

## 3.6 Discussion

The pervasive problem is resolving the question of the approximability of Graph Coloring and Independent Set.

The lower bounds will without doubt be improved eventually. The recent amplifications of Berman and Schnitger [8] and Feige et al. [22] are promising and fertile for further study.

It is highly unlikely that the performance guarantees we have given will remain the best possible. Not much be may needed to knock off some logarithmic or doubly logarithmic factors. All the algorithms given run in time with a low polynomial, and a proper analysis of the more "heavy duty" methods, such as local search, is bound to yield some improvement. Randomization be also be another source of improvements.

Nevertheless, the last section of this chapter that current methodology is insufficient to significantly improve the upper bounds. In fact, none of the other approaches attempted to date seem likely to yield a significant improvement. Given the mounting evidence we conjecture that these problems are inherently hard to solve even approximately.

**Conjecture 1** GRAPH COLORING *is* $\Omega(n/polylog)$ *approximable.*

This, of course, is not universally believed [43]. Since INDEPENDENT SET is strictly harder to approximate than GRAPH COLORING, the conjecture applies to it as well, but it is an interesting question if their approximabilities differ asymptotically.

# Chapter 4

# On-Line Approximations

*Ósvinnur maður*
*vakir um allar nætur*
*og hyggur að hvívetna;*
*þá er móður,*
*er að morgni kemur,*
*allt er víl sem var.*

*The unwise man*
*is awake all night,*
*and ponders everything over;*
*when morning comes*
*he is weary in mind,*
*and all is a burden as ever.*

## 4.1 Our Contributions

We present lower bounds, and some algorithms, that characterize the approximability of on-line coloring surprisingly well. For the deterministic case, we show that for every on-line coloring algorithm, there is a $k$-colorable graph on $k2^{k-1}$ nodes on which the algorithm uses $2^k - 1$ colors. This implies a $2n/(\log n)^2(1+o(1))$ lower bound on the approximability of deterministic on-line coloring. For the randomized case, we show that the previous argument also holds, within a factor of $k$: for every randomized algorithm, there is a $k$-colorable graph on $\mathcal{O}(k^2 2^k)$ nodes on which the algorithm uses an expected number of $2^{k-1}$ colors. Hence randomized on-line coloring is $\Omega(n/(\log n)^3)$ approximable. Notice that all the lower bounds given are for the transparent models, and thus apply equally to the more restrictive ordinary models.

We prove two optimal lower bounds on the approximability of coloring with lookahead. For lookahead $\ell < (\log n)/2$, we show that the above tight deterministic lower bound holds within lower order term. For $\ell = \Omega(\log^3 n)$, we show that the approximability of both deterministic and randomized on-line coloring is $\theta(n/\ell)$.

We give a randomized coloring algorithm against an oblivious adversary that uses no more than $\mathcal{O}(n^{(k-2)/(k-1)}(\log n)^{1/(k-2)})$ colors, implying a $\mathcal{O}(n/\log n)$ performance guarantee.

Finally, we consider the independent set problem in the on-line context. We review two plausible definitions of an on-line independent set algorithm, and prove tight upper and lower bounds for both problems, for both deterministic and randomized algorithms. We finally consider two means of loosening the restrictiveness of the model, allowing the algorithm lookahead and making certain node degree information available, and prove tight bounds in both cases.

## 4.2 Lower Bounds for On-Line Coloring

### 4.2.1 Transparent Coloring

A *valid coloring* of a graph is a partitioning of the nodes into color classes such that nodes of the same color are non-adjacent. In the *transparent* on-line model, both algorithm and adversary incrementally color the graph. For clarity we we say that the algorithm assigns the vertices *bins* while the adversary assigns them *colors.* Consistent assignments require that nodes in same bin must be independent, as well as nodes of same color. The on-line model implies an ordering of the vertices; each vertex is considered in the perspective of the coloring and bin assignments of the *previous* nodes. The process of coloring a graph in our transparent model is thus an indeterminate number of rounds of the form:

> Adversary presents a vertex $v$ adjacent to some of the previous vertices.
>
> Algorithm assigns $v$ a bin consistent with its previous bin assignments.
>
> Adversary assigns $v$ a color consistent with its previous coloring.

Define the *hue* of a bin to be the set of colors of vertices in the bin, and the hue of a vertex to be the colors not used by its previous neighbors. Notice that the color assigned to a vertex must be in its hue. Finally, a vertex is said to be *compatible* with a bin if it is non-adjacent to all vertices in the bin.

A simple algorithm strategy is to allocate a bin for every possible (non-empty) hue, and assign each vertex to the bin with the same hue. The vertex must be compatible with that bin since the colors of the other vertices make up the vertex hue. The bins can be allocated incrementally as more colors appear in the graph, hence $2^k - 1$ bins suffice for every $k$-chromatic graph, even if $k$ is not known in advance.

There is also a simple adversary strategy that forces any such algorithm to use all those bins. Assign each vertex some hue not used by any bin at that point, and make it adjacent to all vertices whose color is not in the hue. This ensures that the algorithm can only assign the vertex to bins whose hue is a subset of the chosen hue. Given that no bin had that exact same hue, the hue of the bin selected must be a strict subset. The vertex can therefore be assigned a valid color that differs from other vertices in the bin. Since a new color can be added to each bin no more than $k$ times, the algorithm must have introduced bins for each of the $2^k - 1$ different hues once $k(2^k - 1)$ vertices have been presented.

We can refine the above strategies for both algorithm and adversary to obtain an optimal minmax value for the problem viewed as a two-player game. It implies optimal bounds for algorithm and adversary performance for every choice of $n = |V(G)|$ and $\chi(G)$.

A good algorithm strategy ensures that each bin contains as many nodes as possible and thus, if necessary, many colors. On the other hand, a good adversary strategy strives to use fewest nodes possible for each bin and thus restrict them to having small hues. We shall show that the following strategies are optimal:

*Adversary: Choose a minimal unused hue $X$, and proceed as above.*

*Algorithm: Assign the vertex to a bin with a maximal compatible hue.*

Let

$$B = \sum_{i=1}^{q-1} \binom{k}{i} + rem \qquad \text{and} \qquad N = \sum_{i=1}^{q-1} i \cdot \binom{k}{i} + q \cdot rem$$

for some $q$, and $0 \leq rem < \binom{k}{q}$. Intuitively, assume the bins are occupying the smallest hues, one per hue. Then $B$ is their count and $N$ is the sum of the cardinalities of their hues.

**Theorem 4.1** *$B$ bins are necessary and sufficient to color transparently all $k$-colorable graphs of order at most $N$.*

*Proof.* The algorithm strategy ensures that no two bins have the same hue, because if one suddenly attains a new color that makes its hue identical to that of another bin, the latter bin had a larger hue that was compatible with the vertex introduced.

The adversary strategy ensures that if any bin has nodes with $t$ different colors, the number of bins with fewer than $t$ colors is always at least the total number of hues with fewer than $t$ colors minus one. Moreover, this holds in the step just before the algorithm introduces a new bin, hence after that introduction the number of bins is at least the total number of hues. Finally, recall that we have shown that the basic adversary method ensures that all vertices in the same bin will have distinct colors.

We obtain the optimal game value from the conjunction of the properties of both players' strategies. At the time when the $B$-th bin is introduced, there is a single bin for the first $B$ hues in some smallest-first linear order of the hues. The sum of the cardinalities of those hues, which equals the total number of vertices in the bins, is $N$.

∎

Our main results follow immediately.

**Corollary 4.1** *For every transparent coloring algorithm, there is a graph on $k2^{k-1}$ vertices for which the algorithm uses $2^k - 1$ bins.*

**Corollary 4.2** *Transparent on-line coloring is precisely $2n/(\log n)^2(1 + o(1))$ approximable.*

Notice that whether or not the algorithm uses random choices is irrelevant to our arguments.

**Corollary 4.3** *$2n/(\log n)^2(1+o(1))$ is a lower bound on approximability of randomized on-line coloring against adaptive adversary, as well as deterministic on-line coloring.*

Notice that neither player needed all the information available to them in our model. The algorithm used only the colors of the neighbors of the incoming vertex, not, for instance, the actual edges. Similarly, the adversary needed only the bin hues.

### 4.2.2 Transparent Coloring with Lookahead

In this section we consider the concept of *lookahead*, where the algorithm is shown a limited portion of the remaining graph at the time it must make its coloring decision. In the problem of transparent coloring with lookahead $\ell$, the adversary presents $\ell$ vertices at a time, that the algorithm must irrevocably assign to bins, after which they are irrevocably assigned colors by the adversary. This is slightly different from the standard definition, in which the algorithm always sees the $\ell$ next vertices, but the latter can easily be simulated with a small constant overhead factor.

In order to prove a lower bound, we must construct a graph, whole "blocks" at a time, that requires many bins but few colors. The graph is constructed simply as a sequence of these blocks, with color and bin assignments of previous blocks known to both algorithm and adversary.

The subgraph for the block is constructed in three parts. As before, we specify the edges from the new vertices to those of previous blocks, and afterwards the colors of those vertices, but now we must also specify in advance the edges internal to the block, which means specifying a type of subgraph on $\ell$ vertices. In what follows, we describe the subgraph construction, connections to previous blocks, and coloring strategy, followed by performance claims and results.

The subgraph $H$ that we, the adversary, choose for the lookahead block is the Turán graph, a sequence of disjoint cliques. More specifically, a sequence of $s$ disjoint cliques of size $t$ each, where $s$ and $t$ depend on the amount of lookahead according to the following table. Let $K$ denote $k/2 - f(k) - g(k)$, where $f(k)$ and $g(k)$ grow asymptotically slower than $k$, to be defined more precisely later.

| $\ell = \|V(H)\|$ | $s = \alpha(H)$ | $t = cl(H)$ |
|---|---|---|
| $1 \ldots K$ | $\ell$ | $1$ |
| $K \ldots K^3$ | $\sqrt{\ell K}$ | $\sqrt{\ell / K}$ |
| $K^3 \ldots$ | $\ell / K$ | $K$ |

When choosing edges from new to old vertices, the collection of bin hues used by the algorithm is the deciding factor. A new vertex is connected (made adjacent) to either all or none of the previous vertices of each color. Specifying a hue for a vertex suffices to completely specify its adjacencies (to nodes in previous blocks). The main task is therefore to find a "good" collection of hues to assign the vertices.

Intuitively, a collection is "good" if each hue contains approximately half the available colors, and if any subcollection has a small intersection. The reason is that we need to ensure that progress made on one vertex affects as little as possible progress made on other vertices – the hues of the vertices must be mutually widely apart. Formally,

**Definition 4.1** *A collection of subsets of a $k$-element set is* well separated *if:*

1. *The size of each subset is within $f(k)$ of $k/2$.*

2. *For any $p$, $2 \leq p \leq k/2 - g(k)$, the intersection of any $p$ subsets is no more than $k/2 - f(k) - p$.*

*where $f(k)$ and $g(k)$ grow asymptotically slower than $k$.*

As long as fewer than half of all possible hues on $k$ colors are used by some bin, we can always find a large well-separated collection.

**Theorem 4.2** *Any collection of at least $2^{k-1}$ subsets of a $k$-element set contains a well separated subcollection of size $k^3$, where $f(k) = 2\sqrt{k \log k}$ and $g(k) = 5 \log k$.*

Using the probabilistic method we can show that almost every collection of $3k^3$ elements is well-separated and has an $k^3$ intersection with the $2^{k-1}$ unused hues. The following lemma is due to Jaikumar Radhakrishnan.

**Lemma 4.1** *Almost every collection of $\mathcal{O}(k^3)$ subsets of a $k$-element set has the property that*

1. *The size of each element differs by no more than $c_1\sqrt{k \log k}$ from $k/2$.*

2. *Any pairwise intersection is no larger than $k/4 + c_2\sqrt{k \log k}$.*

3. *Any intersection of $c_3 \log k$ elements is no larger than $c_4 \log k$*

*where $c_1 = 2$, $c_2 = 2$, $c_3 = 3$, and $c_4 = 5$.*

*Proof.* Consider a uniformly random collection of $\mathcal{O}(k^3)$ subsets. The question whether a given element is contained in a given subset is a Bernoulli trial. The bounds on the sizes of each element, pairwise intersection, and log-wise intersection therefore all follow from bounds on the tail of the binomial distribution.

Let $A_1, A_2, \ldots, A_t$ be fixed subsets of a $k$-element set. Chernoff's bound [12] gives

$$Pr[|A_1| \;>\; k/2 + c_1\sqrt{k \log k}] \;\leq\; 2k^{-2(c_1)^2}$$

and,

$$Pr[|A_1 \cap A_2| \;>\; k/4 + c_2\sqrt{(k/2)\log k}] \;\leq\; 2k^{-(c_2)^2 \cdot 8/3}$$

and simple counting yields

$$Pr[| \cap_{i=1}^{t} A_i| > s] \;\leq\; \binom{k}{s} 2^{-st}$$

Hence, the probability that all the $M = \mathcal{O}(k^3)$ elements chosen have the correct size, and correct pairwise and log-wise intersection is at least

$$1 - 2M \cdot k^{-2(c_1)^2} - 2\binom{M}{2} \cdot k^{-(c_2)^2 \cdot 8/3} - \binom{M}{c_3 \log k}\binom{k}{c_4 \log k} 2^{-c_3 \cdot c_4 \cdot \log^2 k}$$

which is positive for the constants chosen. ∎

To prove the theorem using the lemma it suffices to notice that the intersection of a randomly chosen hue collection of $3k^3$ subsets with a collection of some $2^{k-1}$ subsets (the unused hues) is of size $k^3$ with overwhelming probability.

We say that we *make progress* with a vertex, if the vertex adds a new color to some bin. The key claim is that we can assign the vertices a valid coloring so that at least one vertex from every clique makes progress. Only the first $K^2$ cliques are relevant; the remaining nodes are a "filler" that we expect nothing from.

**Claim 4.1** *Given the preceding construction, at least one node from each of the first $K^2$ cliques makes progress.*

Each lookahead block is colored as follows: Choose $K^2$ nodes, one node from each clique, so that no more than $K$ were placed in any one bin. This can be done in many ways, for instance by greedily choosing up to $K$ nodes from each bin. We can color the nodes so that all of them make progress, since by the well-separation property no bin can amass a hue larger than that of the vertices placed in it. The remaining vertices are assigned arbitrary compatible colors. There are $k - |\text{vertex hue}| \leq k/2 - f(k)$ compatible colors but only $K = k/2 - f(k) - g(k)$ clique vertices, hence they can all be assigned compatible colors.

The following observation links the amount of progress made directly to our goal of forcing the algorithm to use many colors.

**Observation 4.1** *Given an algorithm and adversary for on-line coloring with lookahead $\ell$, if the adversary makes a progress of $\rho$ on each lookahead block, then after $\frac{kB\ell}{\rho}$ nodes have been introduced the algorithm must have used at least $B$ bins. Moreover, if $\ell \leq K$, then $\frac{KB\ell}{\rho}(1 + o(1))$ nodes suffice.*

To see why this observation is true, it suffices to note that after $\frac{kB\ell}{p}$ nodes the adversary has made $kB$ progress according to the hypothesis, and that each bin can make progress no more than $k$ times. If the amount of lookahead is limited, then each bin can progress no more than the size of the largest hue placed in it, which does not exceed $\frac{k}{2} + f(k) = K(1 + o(1))$.

By observation 4.1, we immediately obtain the following lower bounds for coloring with lookahead.

**Theorem 4.3** *For every transparent on-line algorithm **A** with lookahead $\ell$, there is a $k$-colorable graph on $H(\ell, k)2^{k-1}$ vertices for which **A** uses at least $2^{k-1}$ bins, where*

$$H(\ell, k) = \begin{cases} K(1 + o(1)) & \ell \leq K \\ k\sqrt{\ell/K} \quad \textit{when} & K \leq \ell \leq K^3 \\ k \cdot \ell/K^2 & K^3 \leq \ell \end{cases}$$

For an upper bound, we can simply use the transparent algorithm of the previous section, ignoring the lookahead. It follows that a lookahead of $\mathcal{O}(k)$ doesn't help in the transparent model.

**Corollary 4.4** *Transparent on-line coloring with lookahead* $(\log n)/2$ *is* $2n/(\log n)^2(1+o(1))$ *approximable.*

*Proof.* For every $n$ and algorithm $\mathbf{A}$ using lookahead of up to $(\log n)/2$, there exists a $k = \log n$ colorable graph on $n$ vertices for which $\mathbf{A}$ must use $2n/\log n(1 + o(1))$ colors. ∎

There is a simple method to take advantage of considerable lookahead – one that does not rely on transparency.

**Theorem 4.4** *With a lookahead of* $\ell$*, any graph* $G$ *can be on-line colored deterministically with no more than* $\chi(G) \cdot \lceil n/\ell \rceil$ *colors.*

*Proof.* Partition the vertices of the input into blocks of size $\ell$. The algorithm sees the last vertex of the block before it has to color the first vertex. Hence it can off-line color each block with $\chi(G)$ colors (possibly requiring exponential time). ∎

Combining the upper and lower bounds we find that off-line coloring each lookahead block, disregarding the previous vertices, suffices to optimally online coloring with $\Omega(k^3)$ lookahead.

**Corollary 4.5** *On-line coloring with* $\ell = \Omega(\log^3 n)$ *lookahead is* $\Theta(n/\ell)$ *approximable.*

### 4.2.3 Randomized Algorithms Against Oblivious Transparent Adversary

In this section we prove lower bounds for randomized algorithms against oblivious transparent adversary. (The arguments of previous sections apply equally to randomized algorithms against on-line adaptive adversary). By a result of Yao [50] it suffices to show that there is a distribution on $k$-colorable $n$-vertex graphs for which every deterministic (transparent) algorithm uses many colors on average.

We shall essentially use the uniform distribution of those graph that the adaptive adversary may construct. These graphs can be described as all possible sequences of incremental additions of nodes as follows: $v_i$ is adjacent to all of $\{v_1, \ldots, v_{i-1}\}$ that are colored with one of a given set of colors, and is colored by one of the remaining colors. To ensure that the graphs produced by this process are all distinct, we additionally stipulate that the first $k$ nodes form a clique. Notice that the graphs constructed by these random incremental additions are all distinct and occur with equal probability.

**Claim 4.2** *At each step of the oblivious construction, the probability of making progress is at least $1/(2k)$, assuming that the algorithm has used no more than $2^{k-1}$ bins.*

*Proof.* The probability that $X$, the chosen hue, is not found among the bins is at least $1/2$, independent of the color content of the bins. Also the probability that the value in $\overline{X}$ does not occur in the bin specified by the algorithm is at least $1/|X|$, which is at least $1/k$. ∎

This allows us to make equivalent amount of progress as in the deterministic case, if we use $2k$ times as many nodes. That immediately implies a $\Omega(n/\log^3 n)$ lower bound on the approximability of randomized on-line algorithms, but with a little more effort we can get something stronger, namely that the same holds even if we allow significant lookahead.

An oblivious construction for a lookahead block of $K^3$, which generalizes easily to other values of $\ell$, is as follows. For each vertex, we randomly choose a hue, connect it to vertices in previous blocks accordingly, and randomly select a color for it. The vertices of the lookahead block are then combined into a sequence of $K^2$ disjoint cliques of size $K$ each. Because of the adjacencies within the cliques, the colors of some clique nodes will conflict. However, with high probability about half of the initial colors will be compatible. The remaining nodes will be recolored with any valid color (which exists for the same reason as in the deterministic case), but since their colors are no longer independently chosen they will be ignored in the analysis.

**Claim 4.3** *For randomized coloring with lookahead $K^3$, the above strategy makes $\Omega(K^2)$ expected progress.*

*Proof.* Notice that the probability that the random collection of hues was not well-separated is negligible, and can be ignored. Of the $K^3/2$ nodes that were assigned a compatible initial color, we shall see that expected number of $K^2/10$ of them will make progress.

Consider a bin and the $t$ vertices placed in it, during the processing of a given lookahead block. The expected number of those vertices that made progress is at least $\frac{t}{2k}$. Write off all the $t$ vertices placed in the bin, and for each of them that made progress, write off the other nodes within same clique. The amount of progress made on the remaining nodes will be independent of the progress made on the bin considered, hence we the we can resume this marking process and tally the score. For every node that makes progress, $2k + (K - 1)$ nodes are removed (or fewer), hence a total of $\frac{K^3/2}{5K(1+o(1))}$ nodes make progress, expected. ∎

Notice that we can link expected amount of progress made to the expected number of bins required in the same way as observation 1 does for the deterministic case. This implies the following theorem.

**Theorem 4.5** *For any randomized on-line coloring algorithm against a transparent oblivious adversary with lookahead of $\ell$, there is a $k$-colorable graph on $\mathcal{O}(2^{k-1}\max(\ell/k, k^2))$ vertices for which the algorithm uses $2^{k-1}$ expected number of colors.*

We can strengthen corollary 4.5 to include randomized algorithms.

**Corollary 4.6** *On-line coloring with $\ell = \Omega(\log^3 n)$ lookahead is $\Theta(n/\ell)$ approximable, independent of randomization.*

### 4.2.4 Variations

There are numerous imaginable ways of allowing the algorithm greater flexibility (or the adversary lesser), not all of which seem to help in developing effective algorithms. One suggested alternative to lookahead is to allow the algorithm to "recolor" a portion of the nodes, i.e. to independently reassign them bins at some point in the coloring process. This would be useful if few nodes were the cause of a poor coloring while overall the on-line coloring was reasonable. It is not too hard to see that our argument is not affected much by allowing for significant amount of recolorings. Essentially, we can make progress on all nodes that are not recolored, hence the lower bound holds within a constant factor unless almost all, $n - o(n)$, of the vertices are recolored.

## 4.3 Randomized Coloring Algorithm

In an off-line framework we can simply select nodes of high degree and recursively color their neighborhoods. In the on-line framework that is not possible: we can't choose which node is presented next, nor can we know its degree. This makes it difficult to design an effective deterministic algorithm. To foil the adversary somewhat we shall use random bits. The approximation of a randomized algorithm is the maximum expected number of colors used on any graph (and any ordering of the graph). We shall assume that the adversary is *oblivious* in that she must pick the graph before we make any choices, as opposed to incrementally building the graph according to the choices we make.

The idea for the algorithm is to find nodes with reasonably large neighborhoods, recursively color each of those $(k-1)$-colorable neighborhoods with relatively few colors, and try to cover most of the graph with those neighborhoods. As the terminating case in the recursion, we are led to 2-colorable, or bipartite, graphs. As we have seen, there is a simple algorithm, which we shall call BipartiteColor, that colors bipartite graphs using no more than $2 \log n$ colors.

The main algorithm works as follows. We assume to begin with that we know $n$, the number of vertices in the graph. We allocate a number of colors (call them *greedy bins*), and randomly select one of them as a *partitioning bin.* We color each incoming vertex with one of those colors whenever we can (this is the function of the algorithm PartialGreedyColor). If we can't (in which case PartialGreedyColor returns the failure symbol $\perp$), the incoming node must be adjacent to some vertex in each greedy bin. In particular it must be adjacent to some vertex in the partitioning bin. A *residue bin* is allocated for each node in the partitioning bin. A node that cannot be placed in any of the greedy bins is placed into the residue bin associated with some vertex in the partitioning bin that the node was adjacent to. Each residue bin represents a subset of the neighborhood of the associated node, hence it must be $(k-1)$-colorable. The above process is then repeated for the incoming vertex, with the nodes in the residue bins forming the current graph.

To keep track of all the various greedy and residue bins allocated, the algorithm incrementally forms a *coloring tree.* The tree is layered into chromatic levels: the graph induced by nodes belonging to some residue bin has a chromatic number strictly less than the parent graph. Each vertex $t$ in the tree contains the following elements: $n$, current number of elements in tree; $C$, collection of greedy classes; $R$, collection of children of this node, i.e. the residue classes; and $r$, the index of the partitioning color class. Coloring means placing each vertex in a greedy bin at some location in the coloring tree. Either it is placed in the greedy bins at the root, or placed in a residue bin, which means passing it one of the children of the root. This recursive behavior is identical at each level.

```
Color(v)
{ Assign a color to v, compatible with color assignments to previous nodes}
var
  static ColorTree T ← ∅
  static k ← 2
begin
  On-LineColor(T, k, v)
  if (Not-k-colorable error)
     k ← k + 1
     ReplaceSubtree(T, n, k)
     On-LineColor(T, k, v)
  endif
end
```

```
On-LineColor (T, k, v)
{ Assign a color to the vertex v. }
{ T is a coloring tree, of colorability k }
{ n, C, R, r are elements of the current node of T }
begin
  if (k ≤ 2)
    if (IsBipartite(T ∪ {v}))
       BipartiteColor(T,v);
    else
       signal(Not-k-colorable error)
  else
    n ← n + 1
    if (n is a power of 2)
       ReplaceSubtree(T, n, k)
    if (PartialGreedyColor(T, k, v) == ⊥)
      Choose some node v_j in the partitioning class C_r adjacent to v.
      On-LineColor(R_j, k − 1, v)
    endif
  endif
end
```

### 4.3.1 Analysis of the Approximation

The heart of the analysis argument is contained in the following theorem.

**Theorem 4.6** *Let $V_k = \{v_i, \ldots, v_m\}$ be the set of vertices colored during the time that the value of $k$ in algorithm* Color *is unchanged. Then no more than*

$$\mathcal{B}(m,k) = \mathcal{O}\left(\frac{k-1}{(k-2)^{(k-2)/(k-1)}} m^{(k-2)/(k-1)} (\log m)^{1/(k-1)}\right)$$

**ReplaceSubtree** $(T,\ n,\ k)$
{ Throw away colors in subtree, and double allocation of greedy bins }
begin
  $T \leftarrow \emptyset$
  Allocate $s(n,k)$ greedy color classes.
  Choose an integer $r$ uniformly random from $\{1\ \ldots\ s(n,k)\}$
end

*expected number of colors are used to color $V_k$.*

### Notation

Let $A(G,n,k)$ denote the expected number of colors the algorithm uses to color a $k$-colorable graph $G$ on $n$ vertices, when $n$ is known in advance. Let $B(G,n,k)$ similarly be the expected number of colors when $n$ is not known in advance. Let $c$ denote a real-valued constant to be defined later. We shall need the following functions and constants:

$$s'(n,k) = n^{(k-2)/(k-1)}(\log n)^{1/(k-1)}$$
$$c_k^s = (\tfrac{2c}{k-2})^{(k-2)/(k-1)}$$
$$s(n,k) = \lceil c_k^s \cdot s'(n,k) \rceil$$
$$c_k^B = c_k^s \cdot 2^{c/(k-1)} \cdot (k-1) = 2^c \cdot \frac{k-1}{(k-2)^{(k-2)/(k-1)}} \text{ [1]}$$
$$\mathcal{B}(n,k) = c_k^B \cdot s'(n,k)$$
$$\mathcal{A}(n,k) = (k-1)c_k^s s'(n,k)$$

Recall that the number of greedy bins allocated was $s(n,k)$. To maintain sanity and simplicity of presentation, we shall ignore the ceiling in the function.

### Analysis

Given the above notation, the theorem can be restated more concisely as follows.

**Theorem 4.6** $B(G,n,k) \le \mathcal{B}(n,k)$

Proof by induction on $k$. For the base case, observe that $B(G,n,2) = A(G,n,2) \le 2\log n = 2 \cdot s'(n,2) = \mathcal{A}(n,2) = \mathcal{B}(n,2)$.

[1] This expression comes from the resolution of the recursive definitions derived in this section.

We prove the inductive steps in two parts. First, we show that if the inductive assumption, $B(G,n,k-1) \leq \mathcal{B}(n,k-1)$, holds, then $A(G,n,k) \leq \mathcal{A}(n,k)$. Then we show that if $A(G,n,k) \leq \mathcal{A}(n,k)$ then $B(G,n,k) \leq \mathcal{B}(n,k)$. The two parts imply the conclusion.

First let us prove a bound on a funny function that will crop up. Let $c_k^f$ denote $\frac{(k-3)^{(k-3)/(k-2)}}{k-2}$.

**Lemma 4.2** *For all $k \geq 3$, $x^{1/(k-2)}(n-x)^{1-1/(k-2)} \leq c_k^f n$*

*Proof.* Notice that $\max_x x^{1/(k-2)}(n-x)^{1-1/(k-2)} = \sqrt[k-2]{\max_x f(x)}$ where $f(x) = x \cdot (n-x)^{k-3}$. Since $f'(x) = (n-x)^{k-4}[(n-x)-(k-3)\cdot x]$, the function has a breakpoint when $n - x = (k-3)x$, i.e. $x = \frac{n}{k-2}$, which must be a maxima since the value is greater than at the other breakpoint $x = n$. The maximal value of the function is $(n/(k-2))^{1/(k-2)} \cdot (\frac{k-3}{k-2}n)^{(k-3)/(k-2)} = \frac{(k-3)^{(k-3)/(k-2)}}{k-2}$ ∎

Now, the first part of the theorem.

**Claim 4.4** *If $B(G,n,k-1) \leq \mathcal{B}(n,k-1)$, then $A(G,n,k) \leq \mathcal{A}(n,k)$.*

*Proof.* In what follows, $n_i$ is the number of elements of in greedy bin $i$, while $m_j$ is the number of elements in residue bin $j$. Let $g$ denote the total number of elements placed in greedy bins, and $m$ the total number of elements placed in residue bins.

Then,

$$\begin{aligned}
A(G,n,k) &\leq s + \frac{1}{s}\sum_{i=1}^{s}(\text{Cost of coloring assuming partitioning bin } i) \\
&= s + \frac{1}{s}\sum_{i=1}^{s}\sum_{j=1}^{n_i} B(G_j^i, m_j^{(i)}, k-1) \\
&\leq s + \frac{1}{s}\max_{n_i,m_j}\{\sum_{i=1}^{s}\sum_{j=1}^{n_i}\mathcal{B}(m_j^{(i)}, k-1) \mid n = \sum_i n_i + \sum_j m_j\} \\
&= s + \frac{1}{s}\max_{g,m_j}\{\sum_{t=1}^{g}\mathcal{B}(m_{j_t}^{(i_t)}, k-1) \mid n = g + \sum_t m_{j_t}\} \\
&= s + \frac{1}{s}\max_{g,m}\{g \cdot \mathcal{B}(s \cdot \frac{m}{g}, k-1) \mid n = m + g\} \\
&= s + \frac{1}{s}\max_{g,m}\{g \cdot c_{k-1}^B \cdot (s \cdot \frac{m}{g})^{(k-3)/(k-2)} \cdot (\log(s \cdot \frac{m}{g}))^{1/(k-2)}\} \\
&\leq s + \max_g\{g \cdot (\frac{n-g}{g})^{(k-3)/(k-2)}\} \cdot c_{k-1}^B \cdot s^{-1/(k-2)} \cdot (\log n)^{1/(k-2)}
\end{aligned}$$

Step 3 uses the inductive hypothesis. Step 4 uses that since $\mathcal{B}(n,k)$ is a concave function, the sum of the subproblems is maximized when their sizes are all equal.

If we now apply lemma 4.2, the rest of the claim follows using straightforward algebra.

$$\begin{aligned}
A(G,n,k) &\leq s + c_k^f \cdot n \cdot c_{k-1}^B \cdot s^{-1/(k-2)} \cdot (\log n)^{1/(k-2)} \\
&= s + c_k^f \cdot n \cdot c_{k-1}^B \cdot [c_k^s \cdot n^{(k-2)/(k-1)} \cdot (\log n)^{1/(k-1)}]^{-1/(k-2)} \cdot (\log n)^{1/(k-2)} \\
&= s + c_k^f \cdot c_k^{s\,-1/(k-2)} \cdot c_{k-1}^B \cdot n^{(k-2)/(k-1)} \cdot (\log n)^{1/(k-1)} \\
&= s + \frac{(k-3)^{(k-3)/(k-2)}}{k-2} \cdot (\frac{k-2}{2^c})^{1/(k-1)} \cdot [2^c \cdot \frac{k-2}{(k-3)^{(k-3)/(k-2)}}] \cdot s'(n,k) \\
&= [(\frac{2^c}{k-2})^{(k-2)/(k-1)} + (2^c)^{(k-2)/(k-1)} \cdot (k-2)^{1/(k-1)}] \cdot s'(n,k) \\
&= [1 + (k-2)] \cdot (\frac{2^c}{k-2})^{(k-2)/(k-1)} \cdot s'(n,k) \\
&= (k-1) \cdot c_k^s \cdot s'(n,k) \\
&= \mathcal{A}(n,k)
\end{aligned}$$

∎

Now the second half of the theorem.

**Claim 4.5** *If $A(G,n,k) \leq \mathcal{A}(n,k)$, then $B(G,n,k) \leq \mathcal{B}(n,k)$.*

*Proof.* In general, we do not know the number of nodes in advance, at any level in the recursion. The algorithm fixes a value for $n$, and when that count has been exceeded, it doubles its expectation on the total number of vertices. We obtain $d+1$ subproblems, $d = \lceil \log n \rceil$, coloring the nodes $\{1\}, \{2,3\}, \ldots, \{2^{d-2}, \ldots, 2^{d-1}-1\}, \{2^{d-1}, \ldots, n\}$. Let $N$ denote the total number of nodes colored at the end of the last doubling step (i.e. $N = 2^{d-1}$).

Then,

$$\begin{aligned}
B(G,n,k) &= \sum_{i=1}^{d-1} (\#\text{ colors used on elements } 2^{i-1}, \ldots, 2^i - 1) \\
&\quad +(\#\text{ colors used on elements } 2^i, \ldots, n) \\
&\leq \sum_{i=0}^{d-2} A(2^i, k) + (\#\text{ greedy bins used on last step})
\end{aligned}$$

$$
\begin{aligned}
&+(\#\text{ residue bins used on last step})\\
\leq\ & \sum_{i=1}^{d-1} \mathcal{A}(\frac{N}{2^i}, k) + s(n,k) + \mathcal{A}(n-N,k)
\end{aligned}
$$

The first term can be analyzed as follows.

$$
\begin{aligned}
\sum_{i=1}^{d-1} \mathcal{A}(\frac{N}{2^i}, k) &= \sum_{i=1}^{d-1} c_k^A (\frac{N}{2^i})^{(k-2)/(k-1)} (\log \frac{N}{2^i})^{1/(k-1)}\\
&\leq c_k^A N^{(k-2)/(k-1)}\ (\log N)^{1/(k-1)} \sum_{i=1}^{d-1} 2^{-i(k-2)/(k-1)}\\
&= \mathcal{A}(N,k) \sum_{i=1}^{d-1} (2^{-(k-2)/(k-1)})^i\\
&= \mathcal{A}(N,k)\ \frac{1-2^{-d(k-2)/(k-1)}}{1-1/(2^{-(k-2)/(k-1)})}\\
&\leq \mathcal{A}(N,k)\cdot 1/(2^{(k-2)/(k-1)}-1)
\end{aligned}
$$

An inspection or messy approximation shows that $\frac{1}{2^{(k-2)/(k-1)}-1} \leq 1 + \frac{2\sqrt{2}}{k-1}$, which is tight when $k=3$.

Hence,

$$
\begin{aligned}
B(G,n,k) &\leq \sum_{i=1}^{d-1} \mathcal{A}(N/2^i, k) + \mathcal{A}(n-N,k) + s(n,k)\\
&\leq (1+\frac{2\sqrt{2}}{k-1})\mathcal{A}(N,k) + \mathcal{A}(n-N,k) + s(n,k)\\
&\leq (1+\frac{2\sqrt{2}}{k-1})(\mathcal{A}(N,k) + \mathcal{A}(n-N,k)) + \frac{1}{k-1}\mathcal{A}(n,k)\\
&\leq (1+\frac{2\sqrt{2}}{k-1})2^{1/(k-1)}\mathcal{A}(n,k) + \frac{1}{k-1}\mathcal{A}(n,k)\\
&\leq (1+\frac{2\sqrt{2}+1}{k-1})2^{1/(k-1)}\mathcal{A}(n,k)\\
&\leq e^{(2\sqrt{2}+1+\lg 2)/(k-1)}\mathcal{A}(n,k)
\end{aligned}
$$

Setting $c = (\log e)(2\sqrt{2}+1)+1 \leq 6.6$ will satisfy the second claim, and hence the theorem. ∎

**Performance guarantee**

**Theorem 4.7** *The algorithm* Color *on-line colors every graph on $n$ vertices using no more than $\mathcal{O}(n^{(\chi(G)-2)/(\chi(G)-1)}(\log n)^{1/(\chi(G)-1)})$ colors, without* a priori *knowledge of* *either $n$ or $\chi(G)$.*

*Proof.* Color maintains an lower bounded estimate on the chromatic number of the graph. This estimate is incremented only when the current graph has proven to be non $k$-colorable.

Consider the nodes colored while the estimate was $i$. If we denote their count by $n_i$, we know that On-LineColor used no more than $\mathcal{B}(n_i, i)$ colors to color those nodes. This can be bounded from above by $\mathcal{B}(n_i, k)$, which is a concave function. By Jensen's inequality the sum of the of these values over all $i$ is maximized when all node counts are equal. The total number of colors used by Color is thus $(k-1)\ \mathcal{B}(\frac{n}{k-1}, k) \leq (k-1)^{1/(k-1)}\ \mathcal{B}(n, k)$. ∎

**Corollary 4.7** *The performance guarantee of* Color *is $\mathcal{O}(n/\log n)$.*

*Proof.* The ratio of the size of the approximation to the chromatic number of the graph is $\mathcal{O}(n^{(k-2)/(k-1)}(\log n)^{1/(k-1)}/k)$. This function is maximized when $k = \log n - \log\log n$, at which point its value is $\mathcal{O}(n/(2(\log n - \log\log n))) = \mathcal{O}(n/\log n)$. ∎

## 4.4 On-Line Independent Set

How does one define an on-line variant of the independent set problem? As for any on-line graph problem, the nodes are fed one by one along with edges to previous nodes, and the algorithm must make an irrevocable decision on the current node before moving on. The question is what this decision should be.

The most natural version, that we shall call problem A, is to maintain a *single* independent set, and ask if a node should be placed in it or not. However natural this may be it is also inherently impractical: if a node is ever placed in the independent set, then the remaining nodes can be set to be adjacent to it, and only the trivial single node set is found even though all the other $n-1$ nodes were independent. Even randomization doesn't buy us much, as these tight that we shall give show.

| | | |
|---|---|---|
| Deterministic | : | $n-1$ |
| Randomized | : | $\theta(n)$ |

We shall instead focus on a slightly more flexible version, problem B, where the algorithm can maintain multiple independent sets (essentially color the graph) and return the largest independent set found at the end of the game. This version faithfully represent "on-lineness" in that the algorithm must irrevocably decide for every pair of nodes if they are to be in the same independent set, and models somewhat more intelligent on-line strategies.

We can obtain the following tight bounds on variations of this model:

| | | |
|---|---|---|
| Deterministic | : | $n/4$ |
| Randomized | : | $\theta(n/\log n)$ |
| Degree selection | : | $\theta(n)$ |
| Lookahead $\ell$ | : | $\theta(n/\ell)$ |

The variant named "Degree selection", is a relaxation of the on-line framework, where the algorithm is given information about the size of the graph, the degrees of the nodes, and is allowed to select the next node that appears.

### 4.4.1 Arguments

We focus on problem $B$. For a deterministic assignment, First-Fit will ensure that each bin (except perhaps the last one) contain at least one node not belonging to a given independent set in the graph. Thus as long as the graph is not a complete clique, we will always find an independent pair. More generally, if the independence number of the graph is $\frac{s-1}{s}n$ for some real value $s$, then First-Fit will find an independent set of size $\lceil s \rceil$.

We shall show that this is the best possible, focusing on the case $s = 2$. We construct a graph with $n/2$ "good" nodes and $n/2$ "bad" nodes in such a way that each node is non-adjacent to previous good nodes and adjacent to previous bad nodes. Depending on whether the node will be designated as bad or good, it will be adjacent or non-adjacent, respectively, to all remaining nodes. The graphs constructed will be a class of *split graphs*, which are graphs whose vertices are contained in exactly one clique and one independent set. An incoming node will be designated bad if it was placed in a bin previously containing a node (which must be good), or when we are out of good nodes. The construction guarantees that no bin will contain more than 2 nodes. This generalizes to graphs with $\frac{n}{k}$ bad nodes and $\frac{k-1}{k}n$ good nodes, for which a node is made bad if it is the $k$-th node placed in its bin.

Notice that this adversary is transparent. Nevertheless the construction guarantees "zero visibility" to the algorithm in that all nodes are identical when presented. Thus this argument holds also for algorithms that can choose the next node presented depending on adjacencies or non-adjacencies to previous nodes. That class includes the Ramsey algorithm with a default (lexicographic first) pivoting strategy.

In the case when randomization is allowed, we can apply the algorithm of previous section, using Wigderson's observation of how such algorithms can be used to approximate cliques. In graphs with a maximum independent set of size at least $n^{(k-2)/(k-1)}$, it will find one of size $k$, for a performance guarantee of $n/\log n$.

For a lower bound, the graphs used in the deterministic case will apply here also when chosen uniformly randomly. That is, let each node be "bad" with probability one half (or alternatively, randomly permute $n/2$ good and $n/2$ bad nodes). The algorithm can do no better than assigning a random collection of elements to each bin, mimicking the randomized greedy independent set algorithm. The size of the largest such bin will be almost surely $(1-\epsilon)\log_k n$, for graphs with $\alpha = n/k$. The probability that a given independent set collects more than $t$ nodes is $2^{-t}$, hence the probability that the largest of at most $n$ such sets is of size greater than $2\log n$, is $n \cdot 2^{-2\log n} = \frac{1}{n}$. Hence, the expected size of the largest independent set found is $\mathcal{O}(\log n)$.

The number of nodes in a given independent set is geometrically distributed, with $p = 1/2$ in the above graphs. Hence, the expected size of each such set is 2, hence the expected approximability of any randomized algorithm for problem A is no more than $n/4$.

One way of loosening the restrictiveness of the on-line model is to make some local information about the rest of the graph available to the algorithm. The type of information we consider here is the post-degrees of all current vertices at any point. We shall construct a highly regular graph that has similar properties to the split graphs used above.

The graph constructed consists of four vertex disjoint sets: two independent sets $A$ and $B$ of size $n/3$ each, and two cliques $C$ and $D$ of size $n/6$ each. The incoming vertices will alternately be adjacent to all previous nodes in $A$ and $C$, or those in $B$ and $D$. A node will be made a $C$ ($D$) node if the bin it is placed in already has two $B$ ($A$) nodes, otherwise it will be a $B$ ($A$) node. This construction guarantees that no bin will contain more than two nodes each in $A$ and $B$, and one node each in $C$ and $D$. Hence, the approximation ratio for these graphs can be no less than $n/18$.

We are still to see how the degrees can be made available. Each node in $C$ will be adjacent to exactly the previous $A$ and $C$ nodes, and the future $B$ and $C$ nodes. Since at any point the nodes are known to alternate between being $B \cup C$ nodes and not, knowing these post-degrees gives the algorithm no information. Note that again the algorithm can be allowed to pick the next nodes according to adjacencies.

Another way of providing the algorithm with more visibility is to allow lookahead, as in the coloring problem. An immediate algorithm strategy is to compute the maximum independent set in each lookahead block, and use the largest of those. By the pigeonhole principle, its size must be at least $\lceil \alpha(G)/(\#\text{ of lookahead blocks}) \rceil = \alpha(G) \cdot \ell / n$, yielding an approximation ratio of $n/\ell$. Hence, lookahead helps to the extent of what can be gleaned from each block.

This bound can be matched fairly well by the adversary with a padding argument: each lookahead block will contain a single "useful" node and the remainder will be adjacent to every node in the graph. The original lower bound argument then yields $n/(4\ell)$.

The final on-line variant we consider is when the algorithm is allowed to "recolor" some of the nodes, or reassign them to different bins. While this is not effective for coloring, it can be used to circumvent the independent set problem. Each reassignment can move a good node into a safe bin, accumulating as large of an independent set as the number of recolorings.

## 4.5 Open questions

The primary issue is to close the gap on the approximability of deterministic on-line coloring. To improve the lower bound, the adversary argument must utilize the advantage of off-line coloring the graph. Transparency provides the algorithm with information about the future course of action; it could always compute all legal $k$-colorings but doesn't know which one will be used. Can the lack of such information be used to derive stronger bounds?

Another interesting issue is the real worth of lookahead. Is it a threshold function, i.e. is it the case that if on-line coloring is $f(n)$ approximable then so is coloring with lookahead of $n/f(n)$, or do there exist algorithms that capitalize on medium-size lookahead as well as the graph structure?

# References


[1] M. Ajtai, J. Komlós, and E. Szemerédi. A note on Ramsey numbers. *J. Combin. Theory Ser. A*, 29:354–360, 1980.

[2] N. Alon. Personal communications cited in [47], Sept. 1989.

[3] L. Babai, C. Lund, and L. Fortnow. Non-determistic exponential time has two-prover interactive protocols. In *Proc. 31st Ann. IEEE Symp. on Found. of Comp. Sci.*, pages 16–25, Oct. 1990. volume I.

[4] R. Bar-Yehuda and S. Even. A $2 - \frac{\log\log n}{2\log n}$ performance ratio for the weighted vertex cover problem. Technical Report #260, Technion, Haifa, Jan. 1983.

[5] D. Bean. Effective coloration. *J. Symbolic Logic*, 41:469–480, 1976.

[6] S. Ben-David, A. Borodin, R. Karp, G. Tardos, and A. Wigderson. On the power of randomization in online algorithms. In *Proc. 22nd Ann. ACM Symp. on Theory of Computing*, pages 379–386, May 1990.

[7] B. Berger and J. Rompel. A better performance guarantee for approximate graph coloring. *Algorithmica*, 5(4):459–466, 1990.

[8] P. Berman and G. Schnitger. On the complexity of approximating the independent set problem. In *Proc. Symp. Theoret. Aspects of Comp. Sci. Lecture Notes in Comp. Sci. # 349*, pages 256–268. Springer-Verlag, 1989.

[9] A. Blum. An $\tilde{\mathcal{O}}(n^{.4})$ approximation algorithm for 3-coloring. In *Proc. 21st Ann. ACM Symp. on Theory of Computing*, pages 535–542, 1989.

[10] A. Blum. Some tools for approximate 3-coloring. In *Proc. 31st Ann. IEEE Symp. on Found. of Comp. Sci.*, pages 554–562, Oct. 1990.

[11] A. Blum. Interactive protocols and clique lower bounds. Exam report, MIT, Mar. 1991.

[12] B. Bollobás. *Random Graphs.* Academic Press, 1985.

[13] B. Bollobás and P. Erdős. Cliques in random graphs. *Math. Proc. Cambridge Philos. Soc.*, 80:419–427, 1976.

[14] R. Boppana. Private communication.

[15] O. Bray. *The Elder or Poetic Edda. Part I – The Mythological Poems*, volume II of *Viking Club Translation Series.* King's Weighhouse Rooms, London, 1908.

[16] D. Bruschi, D. Joseph, and P. Young. A structural overview of NP optimization problems. In *Proc. 2nd Intl. Symp. on Optimal Algorithms, Lecture Notes in Computer Science # 401*, pages 205–231, Bulgaria, June 1990. Springer-Verlag.

[17] V. Chvátal. Determining the stability number of a graph. *SIAM J. Comput.*, 6(4), Dec. 1977.

[18] P. Crescenzi and A. Panconesi. Completeness in approximation classes. In *Proc. 17th Conf. on Fundamentals of Computation Theory. Lecture Notes in Computer Science # 380*, pages 116–126. Springer-Verlag, Aug. 1989.

[19] P. Erdős. Some remarks on chromatic graphs. *Colloq. Math.*, 16:253–256, 1967.

[20] P. Erdős and G. Szekeres. A combinatorial problem in geometry. *Compositio Math.*, 2:463–470, 1935.

[21] R. Fagin. Generalized first-order spectra, and polynomial-time recognizable sets. In R. Karp, editor, *Complexity and Computations.* AMS, 1974.

[22] U. Feige, S. Goldwasser, L. Lovász, S. Safra, and M. Szegedy. Approximating clique is almost NP-complete. In *Proc. 32nd Ann. IEEE Symp. on Found. of Comp. Sci.*, pages 2–12, Oct. 1991. To appear.

[23] T. Gallai. Kritische graphen I. *Publ. Math. Inst. Hungar. Acad. Sci.*, 8:165–192, 1963. (See Bollobás, B. *Extremal Graph Theory.*, Academic Press, 1978, page 285).

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

[25] G. R. Grimmett and C. J. H. McDiarmid. On colouring random graphs. *Math. Proc. Cambridge Philos. Soc.*, 77:313–324, 1975.

[26] A. Gyárfás and J. Lehel. On-line and first fit colorings of graphs. *J. Graph Theory*, 12(2):217–227, 1988.

[27] S. Irani. On-line coloring inductive graphs. In *Proc. 31st Ann. IEEE Symp. on Found. of Comp. Sci.*, pages 470–479, Oct. 1990.

[28] D. S. Johnson. Worst case behavior of graph coloring algorithms. In *Proc. 5th Southeastern Conf. on Combinatorics, Graph Theory, and Computing. Congressus Numerantium X*, pages 513–527, 1974.

[29] G. Kant and J. van Leeuwen. Graph coloring algorithms. Manuscript, 1990.

[30] H. A. Kierstead. The linearity of first-fit colorings of interval graphs. *SIAM J. Disc. Math.*, 1991.

[31] H. A. Kierstead and W. T. Trotter. An extremal problem in recursive combinatorics. In *Proc. 12th Southeastern Conf. on Combinatorics, Graph Theory, and Computing. Congressus Numerantium XXXIII*, pages 143–153, 1981.

[32] P. G. Kolaitis and M. N. Thakur. Logical definability of NP optimization problems. Technical Report UCSC-CRL-90-48, University of California, Santa Cruz, 1990.

[33] Ólafur Briem. *Eddukvæði.* Iðunn, Reykjavík, Iceland, 1985.

[34] N. Linial and V. Vazirani. Graph products and chromatic numbers. In *Proc. 30th Ann. IEEE Symp. on Found. of Comp. Sci.*, pages 124–128, 1989.

[35] L. Lovász, M. Saks, and W. T. Trotter. An online graph coloring algorithm with sublinear performance ratio. *Discrete Math.*, 75:319–325, 1989.

[36] B. Monien and E. Speckenmeyer. Ramsey numbers and an approximation algorithm for the vertex cover problem. *Acta Inf.*, 22:115–123, 1985.

[37] G. L. Nemhauser and L. Trotter. Vertex packings: Structural properties and algorithms. *Math. Programming*, 8:232–248, 1975.

[38] P. Orponen and H. Mannila. On approximation preserving reductions: complete problems and robust measures. *To appear*, 1987.

[39] A. Panconesi and D. Ranjan. Quantifiers and approximation. In *Proc. 22nd Ann. ACM Symp. on Theory of Computing*, pages 446–456, May 1990.

[40] C. H. Papadimitriou and M. Yannakakis. Optimization, approximation, and complexity classes. *J. Comput. Syst. Sci.*, 43:425–440, 1991.

[41] J. B. Shearer. A note on the independence number of triangle-free graphs. *Discrete Math.*, 46:83–87, 1983.

[42] H. U. Simon. On approximate solutions for combinatorial optimization problems. *SIAM J. Disc. Math.*, 3(2):294–310, May 1990.

[43] E. Szemerédi. Private communication, 1991.

[44] R. E. Tarjan and A. E. Trojanowski. Finding a maximum independent set. *SIAM J. Comput.*, 6(3):537–546, Sept. 1977.

[45] P. B. Taylor, W. H. Auden, and P. H. Salus. *The Elder Edda – A Selection.* Faber and Faber, London, 1969.

[46] S. Vishwanathan. Personal communications, 1991.

[47] S. Vishwanathan. Randomized online graph coloring. *J. Algorithms*, 13:657–669, Dec. 1992.

[48] A. Wigderson. Personal communications.

[49] A. Wigderson. Improving the performance guarantee for approximate graph coloring. *J. ACM*, 30(4):729–735, 1983.

[50] A. C.-C. Yao. Probabilistic computations: Towards a unified measure of complexity. In *Proc. 18th Ann. IEEE Symp. on Found. of Comp. Sci.*, pages 535–542, 1977.

# Vita

## Magnús Már Halldórsson

**1978–1982** Diploma, Hamrahlíð Junior College, Reykjavík, Iceland.

**1982–1985** B.S. in Computer and Information Science, and Mathematics, University of Oregon, Eugene, Oregon.

**1985–1987** Software Support Consultant, Kristján Ó Skagfjörð, Reykjavík, Iceland.

**1986–1991** Graduate work in Computer Science, Rutgers, The State University of New Jersey. Teaching and Research Assistant.

**1990** Ravi B. Boppana and Magnús M. Halldórsson.
Approximating maximum independent sets by excluding subgraphs.
In *Proc. of 2nd Scandinavian Workshop on Algorithm Theory. Lecture Notes in Computer Science #447*, pages 13–25, July 1990, Springer-Verlag.

**1990** M.S. in Computer Science.

**1991** Ph.D. in Computer Science.

**1991-1992** Visiting Researcher, Tokyo Institute of Technology, Tokyo, Japan.