Diss. ETH No. 20044

Combinatorial Games on Graphs

A dissertation submitted to
ETH Zürich

for the degree of
Doctor of Sciences

presented by
Heidi Gebauer
M.Sc. ETH in Computer Science
born August 21, 1981
citizen of Zürich, Switzerland

accepted on the recommendation of
Prof. Dr. Emo Welzl, examiner
Prof. Dr. Tomasz Łuczak, co-examiner
Prof. Dr. Tibor Szabó, co-examiner

2011

Abstract

The theory of combinatorial games is a rapidly developing subject of modern combinatorics. It is concerned with the study of certain two-player-games, ranging from popular games as Tic-Tac-Toe and Connect Four to merely abstract games defined in graph-theoretic terms. Besides being of interest on their own right, combinatorial games are strongly connected to other areas of combinatorics, such as the probabilistic method, Ramsey theory, random graphs and extremal graph theory. The study of this subject inspired the derivation of several important results in theoretical computer science. One such highlight is the first algorithmization of the famous Lovász Local Lemma by Beck.

Though some scattered results about combinatorial games appeared much earlier, the systematic research on this subject originated from a seminal paper of Erdős and Selfridge. Later the study of combinatorial games was developed greatly by Beck in a series of influential works. Since then this field has grown more and more, and quite a few new and important results have been published.

An intriguing phenomenon in the theory of combinatorial games is the mysterious relationship to probability theory: Informally, the outcome of a game involving two clever opponents can often be predicted quite accurately by studying the setting where the same game is played by two players who act completely randomly. This phenomenon, which is often referred to as the *random graph intuition*, was first pointed out by Chvátal and Erdős, only a few years after the seminal paper of Erdős and Selfridge was published. So far, the random graph intuition has proven to be valid in various games, and we only know surprisingly few games where it fails.

In this thesis we address several issues arising in the study of combinatorial games. First, we verify the random graph intuition for certain games, thereby solving some open problems by Beck. Second, we consider some games for which the question "who wins?" is already solved. For these games we are concerned with finding *fast* winning

strategies, i.e., strategies which guarantee that a player can win in very few moves. Third, we study a game-theoretic variant of the famous Ramsey-numbers. Finally, we construct a special class of binary trees with certain structural properties. These trees allow us to disprove a conjecture of Beck on combinatorial games, and, moreover, they also lead to a new result in the field of satisfiability of Boolean formulas. This is yet another example for the fact that combinatorial games connect to various problems in (sometimes seemingly unrelated) other areas.

Zusammenfassung

Die kombinatorische Spieltheorie ist ein junges, sich rasant entwickelndes Gebiet der Kombinatorik, auf dem in den letzten Jahren grosse Fortschritte erzielt wurden. Es beschäftigt sich mit der Analyse einer bestimmten Klasse von Spielen für zwei Mitspieler – sie beinhaltet sowohl bekannte, verbreitete Gesellschaftsspiele wie Tic Tac Toe oder Vier Gewinnt, als auch eher abstrakte Spiele, deren Beschreibung auf graphentheoretischen Begriffen basiert. Die kombinatorische Spieltheorie ist eng verbunden mit anderen Gebieten der Kombinatorik, wie beispielsweise der Ramsey-Theorie, der extremalen Graphentheorie, der probabilistischen Methode und der Theorie der randomisierten Graphen. Einige Resultate in diesem Themenbereich dienten überdies als Inspiration für die Herleitung bedeutender Ergebnisse in der Theoretischen Informatik. Ein solcher Glanzpunkt ist beispielsweise die erste Algorithmisierung des berühmten Lovász Local Lemma von Beck.

Obschon einzelne Resultate über kombinatorische Spiele bereits viel früher publiziert worder waren, wird die wegweisende Arbeit von Erdős und Selfridge weithin als Beginn der systematischen Forschung auf diesem Gebiet angesehen. Das Studium der kombinatorischen Spiele wurde später von Beck in einer Reihe von einflussreichen Arbeiten bedeutend weiterentwickelt. In der Zwischenzeit ist dieses Gebiet mehr und mehr gewachsen, und es wurden verschiedene wichtige Resultate publiziert.

Ein sehr verblüffendes Phänomen in der Theorie der kombinatorischen Spiele sind die überraschenden Parallelen zur Wahrscheinlichkeitstheorie: Der Ausgang eines Spiels lässt sich, vereinfacht gesagt, relativ oft erstaunlich genau voraussagen, indem man das Szenario analysiert, bei welchem zwei komplett zufällige Spieler gegeinander agieren. Dieses Phänomen, welches oft als *random graph intuition* bezeichnet wird, wurde als erstes von Chvátal und Erdős aufgezeigt, lediglich wenige Jahre nach der Publikation der bahnbrechenden Arbeit von Erdős und Selfridge. Bis jetzt konnte diese Intuition für diverse Spiele bestätigt werden, und es ist erst für überraschend wenige Spiele bekannt, dass die Intuition in

diesem Fall nicht zutrifft.

In dieser Arbeit beschäftigen wir uns mit verschiedenen Aspekten, die beim Studium der kombinatorischen Spiele auftreten.

Zunächst verifizieren wir die *random graph intuition* für einige ausgewählte Spiele, was uns unter anderem ermöglicht, mehrere von Beck gestellte offene Probleme zu lösen. Ausserdem betrachten wir einige Spiele, für welche die Frage "welcher Spieler gewinnt?" bereits beantwortet ist. Für diese Spiele entwickeln wir sogenannte *schnelle Strategien*, das bedeutet Strategien, welche garantieren, dass ein Spieler schon in relativ wenigen Zügen gewinnen kann. Desweiteren studieren wir eine spieltheoretische Variante der berühmten Ramsey-Zahlen. Schlussendlich konstruieren wir eine spezielle Klasse von binären Bäumen mit bestimmten strukturellen Eigenschaften. Mithilfe dieser Bäume können wir zum einen eine Vermutung von Beck über kombinatorische Spiele widerlegen, und zum andern ein neues Ergebnis über die Erfüllbarkeit von logischen Formeln herleiten. Dies illustriert einmal mehr die Tatsache, dass kombinatorische Spiele eng mit einer Vielzahl von Problemen in anderen (manchmal scheinbar unverwandten) Gebieten zusammenhängen.

Acknowledgements

I am deeply grateful to my supervisor Tibor Szabó for his scientific and moral support during my PhD studies. Thanks to his unerring sense for interesting problems, his numerous stimulating ideas and insightful comments, and also his witty humor, it was a great pleasure to be his student.

I would like to especially thank Emo Welzl for letting me be part of his research group GREMO and for providing a very pleasant and inspiring research atmosphere. I have greatly benefited from his knowledge, his valuable advices and his impressive mathematical intuition.

I am also indepted to Tomasz Łuczak for reviewing this thesis, being my co-examiner and raising intriguing questions.

It was a joy to collaborate with Tobias Christ, Andrea Francke, Jiří Matoušek, Robin Moser, Dominik Scheder, Tibor Szabó, Gábor Tardos, Takeaki Uno and Emo Welzl. I would like to thank them for fruitful discussions.

I really appreciated the company of the current and former members of GREMO: Robert Berke, Yves Brise, Tobias Christ, Andrea Francke, Bernd Gärtner, Anna Gundert, Franziska Hefti, Timon Hertli, Michael Hoffmann, Martin Jaggi, Vincent Kusters, Robin Moser, Gabriel Nivasch, Andreas Razen, Leo Rüst, Andrea Salow, Dominik Scheder, Eva Schuberth, Sebastian Stich, Marek Sulovský, Patrick Traxler, Floris Tschurr, Uli Wagner, and Philipp Zumstein.

In particular, I am thankful to Andrea Salow for resolving all kinds of administrative issues, and for launching and organizing the popular MiLuDi; Dominik, Robin and Timon for reviewing parts of this thesis; Bernd for introducing me to Scratch; Uli for organizing the Reading Seminar and selecting interesting papers; Andrea, Anna and Robin for pretending not to be annoyed when their office was abused as a meeting point; Martin and Michael for helping me in many, many struggles with Unix and Latex; Tobias for interesting discussions about didactics; Yves for being awesome at pool; Sebastian for giving an instructive talk about

the CAB routing problem; and Vincent for introducing us to Snoepgoed brought by Sinterklaas.

I am also grateful to my boyfriend Joris for his help and patience.

My special thanks go to my parents for their love and support and all the many things which can not be expressed in a few words.

Contents

1 Introduction

The theory of combinatorial games is a rapidly developing subject of modern combinatorics. It is concerned with analyzing certain perfect information games involving two players, ranging from well-known classical games as Tic-Tac-Toe and Connect Four to completely abstract games defined in graph-theoretic terms. Besides being an interesting field on its own, the theory of combinatorial games is strongly connected to quite a few other branches of combinatorics, such as the probabilistic method, Ramsey theory, random graphs and extremal graph theory. The study of this subject inspired the derivation of several important results in theoretical computer science, including many derandomization techniques. One such highlight is the first algorithmization of the famous Lovász Local Lemma by Beck [16].

Though some scattered results about combinatorial games appeared much earlier, the seminal paper of Erdős and Selfridge is widely considered the origin of the systematic research on this subject, which afterwards was greatly developed by Beck in a series of influential papers. Since then the field of combinatorial games has grown more and more, and quite a few new and important results have been published.

For further discussion we introduce the following notation. For a given nonempty set X and some subset $\mathcal{F} \subseteq 2^X$ ("2^X" denotes the power set, i.e., the set consisting of all subsets of X) we consider the *combinatorial*

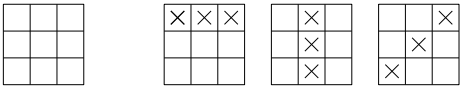


Figure 1.1: The Tic Tac Toe Board (on the left) and some of its winning sets.

game (sometimes also called positional game) $(X, \mathcal{F})$ where two players alternately take turns at claiming a previously unclaimed element of X. We refer to X and $\mathcal{F}$ as the "board" and the "family of winning sets", respectively. For example, the ordinary Tic-Tac-Toe game can be represented by the combinatorial game $(X, \mathcal{F})$ where X consists of the nine squares (arranged in a 3×3 array) and $\mathcal{F}$ is the set of all square-triples in a line (horizontal, vertical, or diagonal). Figure 1.1 shows an illustration. In such games as Tic-Tac-Toe, the player who occupies *first* a winning set completely is the winner. In the language of combinatorial games, these games are called *strong games*. Apart from a handful of very simple cases like Tic-Tac-Toe, strong games are so far very hard to analyze and, considering the currently known methods, it seems that extensive case distinctions are inevitable.

The situation significantly changes when we consider the *weak version* of these games instead. Here, the goal of the players is not symmetric: One player, called Maker, aims to fully occupy a winning set, whereas the other player, called Breaker, tries to prevent Maker from achieving his goal. In other words, Breaker wins if, after all elements of X have been claimed, Maker does not possess all elements of some winning set.

It turns out that weak games exhibit interesting combinatorial structures and connect to quite a few other problems. So, they are more natural in the sense that their analysis is not only simpler but can sometimes also be applied to other areas. We will point out some of these connections in Section 1.1.

Clearly, there are no draws in weak games: Every weak game has the property that either Maker has a winning strategy (i.e., playing against an arbitrary strategy of Breaker, Maker can fully occupy a winning set) or Breaker has a winning strategy. In the first case we say that the game is a *Maker's win* and in the second case we say that the game is a *Breaker's win*.

Throughout this thesis we will only consider weak games, thus we abbreviate them by "games". Unless otherwise stated we will assume

that Maker starts the game, although for most of the games of our interest we are after asymptotic statements which do not depend on which player makes the first move. Classical examples of such games are played on the edge set of the complete graph on n vertices (so $X = E(K_n)$). Often the family of winning sets consists of all edge-sets which have some (usually monotone) graph-property P. Such games, which will be called *graph games*, are, e.g., the connectivity game, where Maker's goal is to occupy a spanning tree (i.e., P is the property that the edge-set contains a spanning subgraph) or the clique game, where Maker aims to build a clique of some fixed size.

We will sometimes also consider a generalized version of games where Maker and Breaker do not claim one but several elements of X in one round: In an $(m : b)$ *game* Maker claims m elements and Breaker claims b elements per turn.

In Section 1.1 we have a closer look at some selected connections between combinatorial games and other problems. In Sections 1.2–1.5 we describe the game-variations we study and give a short introduction to each chapter of this thesis.

1.1 Connection to Other Problems

1.1.1 Satisfiability of Boolean Formulas

We consider so called CNF formulas. For example,

$$\mathcal{F} = (x \vee \bar{y} \vee z) \wedge (x \vee a \vee \bar{z}) \wedge (a \vee b \vee c) \wedge (x \vee \bar{y} \vee \bar{z})$$

is a CNF formula. Formally, we define a CNF formula as the conjunction of *clauses* where a clause is the disjunction of *literals* (a literal is either a Boolean variable or its negation). An interesting subclass is the class of CNF formulas where every clause contains *exactly* k literals. These formulas are called k-CNF formulas. For example, $\mathcal{F}$ is a 3-CNF formula. For every assignment of true/false-values to its variables, we can evaluate a given formula: e.g., by setting a to true and all other variables to false, $\mathcal{F}$ evaluates to true, whereas by setting all variables to false, $\mathcal{F}$ evaluates to false. A given CNF formula $\mathcal{G}$ is called *satisfiable* if there is an assignment where $\mathcal{G}$ evaluates to true. We call such an assignment a *satisfying assignment*. The problem of deciding whether a given CNF formula is satisfiable, often abbreviated by SAT, plays a major role in theoretical computer science; e.g., it is often referred to as the 'mother' of NP-complete problems.

The following is a well-known result.

Every k-CNF formula with fewer than 2^k clauses is satisfiable. (1.1)

This result is clearly tight as the k-CNF formula consisting of all possible k-clauses on k fixed variables is not satisfiable. One way to prove (1.1) is to consider the probability distribution where every variable is set to true with probability $\frac{1}{2}$, independently of the other variables. It can be checked that the probability that this random assignment satisfies all clauses is strictly larger than zero. This argument yields that a satisfying assignment exists but it does not (at least not in an obvious way) provide an algorithm for finding such an assignment substantially faster than the exhaustive search approach.

Method of Conditional Expectation The famous Erdős-Selfridge Theorem [42] states a result similar to (1.1) for games:

If every winning set of a game has size exactly k and the number of winning sets is smaller than 2^{k-1} then Breaker has a winning strategy.

In their proof, Erdős and Selfridge showed that Breaker can succeed using the following strategy. In each of his moves he selects (among all not yet claimed elements of X) the element of X which minimizes the expected number of winning sets completely occupied by Maker, with respect to the probability distribution where every remaining element is assigned to either player independently with probability 1/2.

Their approach of developing an algorithm which proceeds step by step, choosing each step in such a way that a carefully chosen expectation value is minimized (or maximized), turned out to be applicable to a bunch of other problems and thus became known as "the method of conditional expectation". For example, it can be used to give an algorithmic proof for (1.1): Consider the algorithm which in each step picks an arbitrary variable (which has not been set to true or false yet) and chooses its Boolean value in such a way that the expected number of satisfied clauses is maximized, with respect to the probability distribution where every remaining variable is assigned a value of $\{\mathsf{true}, \mathsf{false}\}$ uniformly and independently. It can be shown that this algorithm efficiently computes a satisfying assignment for every given k-CNF formula with at most $2^k - 1$ clauses.

Satisfiability in Terms of Games In a game $(X, \mathcal{F})$ Maker can always use a strategy of the following kind. At the beginning he divides all

elements of X into pairs and whenever Breaker claims one vertex of a pair he takes the other one (for simplicity we assume here that the cardinality of X is even and that Maker lets Breaker start the game). Such a strategy is called a *pairing strategy*. Every CNF formula $\mathcal{G}$ can be considered as the game where the board consists of the set of literals, the family of winning sets contains all clauses of $\mathcal{G}$ and, as an additional restriction, Maker is required to use the pairing strategy where every literal x is paired with its negated version $\bar{x}$. With every course of such a game we can associate the assignment which sets a literal to true if and only if it was claimed by Breaker. If $\mathcal{G}$ is satisfiable then Breaker can win this game by fixing a satisfying assignment and claiming those literals which are set to true by this assignment. Conversely, if Breaker has a winning strategy then the corresponding assignment satisfies $\mathcal{G}$. So the satisfiability of $\mathcal{G}$ is equivalent to Breaker having a winning strategy in the corresponding game.

The Lovász Local Lemma The famous Lovász Local Lemma allows to extend (1.1) to a much more powerful statement, replacing the *global* constraint on the size of the formula with an appropriate *local* constraint. We say that two clauses *intersect* if they share at least one variable.

$$\begin{aligned} &\textit{Every } k\textit{-CNF formula where every clause intersects} \\ &\textit{at most } \frac{2^k}{e} - 1 \textit{ other clauses is satisfiable.} \end{aligned} \tag{1.2}$$

It was a long-standing open problem to *derandomize* (1.2), i.e., to develop an algorithm which efficiently finds a satisfying assignment for every given k-CNF formula fulfilling the condition of (1.2). In 1991, Beck achieved a breakthrough by proving in [16] that a polynomial-time algorithm exists which finds a satisfying assignment for every k-CNF formula in which each clause intersects at most $2^{k/48}$ other clauses. This very first derandomization of the Lovász Local Lemma was inspired by his research in the theory of combinatorial games.

In the meantime the $\frac{k}{48}$ in the exponent has been successively improved in a series of papers to $\frac{k}{8}$ [8], $\frac{k}{4}$ [72], $\frac{k}{2}$ [63] until Moser and Tardos [62] gave a fully constructive version of the Lovász Local Lemma, with the implication that for every k-CNF formula fulfilling the condition of (1.2) their algorithm finds a satisfying assignment in (probabilistic) polynomial time.

Our Contribution In Chapter 5 we construct a special class of binary trees which have implications to games and – due to the connection between games and SAT described above – also to SAT. In the context of games we consider a conjecture of Beck which basically states that a similar phenomenon as the Local Lemma holds for games. More precisely, it says that in every game $(X, \mathcal{F})$ where every winning set intersects fewer than $2^{k-1} - 1$ other winning sets, Breaker has a winning strategy. We refute this conjecture.

In the context of SAT we consider the following problem. For every k we let $f(k)$ denote the largest integer s such that every k-CNF formula where every variable occurs in at most s clauses is satisfiable. $f(k)$ is a value with many interesting properties, which has been widely studied. The Local Lemma can be used to give a lower bound on $f(k)$. We provide a construction of appropriate unsatisfiable k-CNF formulas which shows that this lower bound is asymptotically tight. More precisely, we prove that $f(k) = (1 + o(1))\frac{2^{k+1}}{ek}$. Here, $o(1)$ is some function that tends to 0 as k grows.

1.1.2 Random Graphs

In this subsection we restrict on graph games, i.e., the board is the edge set of a complete graph on n vertices and Maker's goal is to create a graph which possesses some fixed monotone property P. Recall that the variant of a game where Maker claims m edges per turn and Breaker claims b edges per turn is called an $(m : b)$-game. An interesting paradigm, which was pointed out first by Chvátal and Erdős [34], and was later investigated further in many papers of Beck [15, 17, 18, 19] and Bednarska and Łuczak [22], is the *random graph intuition*: Let $\mathcal{G}$ denote an $(m : b)$ game and let P denote the corresponding graph property Maker aims to achieve. In the *random game* the players are replaced with "random players" which select their edges in each round completely at random; i.e., *RandomMaker* claims m random unclaimed edges per move and *RandomBreaker* claims b random unclaimed edges per move. So, by the end of the game RandomMaker's graph looks like a random graph $G(n, M)$ with $M = \lceil \frac{m}{m+b}\binom{n}{2} \rceil$ edges (following the standard notation, $G(n, M)$ denotes the probability space of graphs where each graph with n vertices and M edges occurs with the same probability). The random graph intuition basically says that if $G(n, M)$ contains P with high probability, then this indicates that Maker has a winning strategy in $\mathcal{G}$, and, conversely, if $G(n, M)$ with high probability does not contain

P, then this indicates that Breaker has a winning strategy in $\mathcal{G}$.

An example supporting the random graph intuition is the *q-clique game*, where Maker's goal is to occupy all edges of a clique on q vertices. It is well-known that with high probability the size of the largest clique in $G(n, \frac{1}{2}\binom{n}{2})$ is $(2-o(1))\log n$, thus the threshold where the random $(1:1)$ q-clique game turns from a RandomMaker's win to a RandomBreaker's win is around $q = (2-o(1))\log n$. Due to a result of Beck [21] the situation is the same in the deterministic case: The largest value q such that Maker has a winning strategy in the $(1:1)$ q-clique game is $(2-o(1))\log n$.

So far, for various games it has been proved that they support the random graph intuition, see, e.g., [18, 22, 49, 73]. For a small number of games it has been established that the random graph intuition fails: In the *diameter game* the graph property Maker aims to achieve is that the diameter is at least two (i.e., every pair of vertices has distance at most two in his graph). It is known (and not very difficult to show) that in the random graph $G(n, \frac{1}{2}\binom{n}{2})$ with high probability every pair of vertices has distance at most two. Hence, RandomMaker wins the random $(1:1)$ diameter game with high probability. However, Balogh, Martin and Pluhár [11] proved that actually Breaker has a strategy to win the $(1:1)$ diameter game, which yields that the probabilistic intuition fails in this case. Second, for a given b, let $\mathcal{G}_{\mathsf{NP}}(b)$ denote the $(1:b)$ *non-planarity game*, where the graph property Maker aims to achieve is non-planarity. That is, Maker wins if at the end of the game his graph has no planar embedding. (Note that if $b \leq \frac{n}{6} - 1$, Maker's graph finally contains $\frac{1}{1+b}\binom{n}{2} > 3n-6$ edges and hence is non-planar. Thus, for $b \leq \frac{n}{6} - 1$, Maker will always win, no matter what strategy he uses.) Due to known results about random graphs, for $b \leq (1-o(1))n$, RandomMaker wins the random version of $\mathcal{G}_{\mathsf{NP}}(b)$ with high probability. On the other hand, Hefetz, Krivelevich, Stojaković and Szabó [52] showed, applying a result of Bednarska and Pikhurko [23], that Breaker has a strategy to win $\mathcal{G}_{\mathsf{NP}}(b)$ for $b \geq \frac{n}{2}$. This means that the random graph intuition fails for every b with $\frac{n}{2} \leq b \leq (1-o(1))n$.

It is an interesting open problem to determine suitable criteria which guarantee that for a given game the random graph intuition (or maybe some weaker version of it) holds.

Our Contribution We verify the random graph intuition for a few other games (e.g., the *connectivity game* where Maker's goal is to build a spanning tree), thereby solving two problems Beck [21] included in his list of

the *seven most humiliating open problems* of the field of combinatorial games. Moreover, we show that, though the random graph intuition is supported by the $(1:1)$ clique game, it fails for the $(m:m)$ clique game for every $m \geq 6$. On first sight this is surprising because it seems that, since each player can claim equally many edges per turn as his opponent, the $(m:m)$ game is "as fair as" the $(1:1)$ game. However, it turns out that the larger m gets the more Maker can profit from it. Thus, the outcome of the $(m:m)$ clique game is significantly different from the outcome of the $(1:1)$ game, and as a consequence the random graph intuition fails for large enough m.

1.2 Verifying the Random Graph Intuition

A classical, well-studied graph game is the *Shannon switching game* analyzed completely by Lehman [61]. In the Shannon switching game both players take *one* edge per turn and Maker tries to occupy the edges of a spanning tree. Lehman proved that Maker wins this game "easily". Here by "easily" we mean that Maker does not need the full edge-set of K_n, he wins even if the game is played on the restricted board consisting of the edges of two edge-disjoint spanning trees.

Chvátal and Erdős [34] suggested to even out this advantage of Maker by allowing Breaker to occupy b edges in each round instead of just one. The integer $b = b(n) > 1$ is called the *bias* of Breaker. Chvátal and Erdős provided a strategy for Maker to occupy a spanning tree even if Breaker plays with bias $(\frac{1}{4} - o(1))\frac{n}{\ln n}$. They also showed that Breaker, playing with a bias $(1 + o(1))\frac{n}{\ln n}$ can occupy all edges incident to some vertex, thus of course making it impossible for Maker to build a spanning tree. Motivated partially by this problem, Beck [13] devised a general potential function method to deal with biased combinatorial games under much more general circumstances. He then used his criterion to infer a strategy for Maker for occupying a spanning tree even if Breaker's bias is as large as $(\ln 2 - o(1))\frac{n}{\ln n} \approx 0.69\frac{n}{\ln n}$.

We let $b_{\mathcal{T}}$ denote the largest bias b of Breaker such that Maker, taking one edge in each turn, can occupy a spanning tree while Breaker takes b edges in each turn. The above results yield that

$$0.69\frac{n}{\ln n} \leq b_{\mathcal{T}} \leq (1 + o(1))\frac{n}{\ln n}.$$

From the theory of random graphs it is known that the threshold M where the probability that the random graph $G(n, M)$ contains a span-

ning tree turns from almost 0 to almost 1, is $M = (\frac{1}{2} + o(1))n \ln n$. Thus the random graph intuition (c.f. Subsection 1.1.2) suggests that $b_{\mathcal{T}} = (1 + o(1))\frac{n}{\ln n}$. In our main result of Chapter 2 we show that

$$b_{\mathcal{T}} = (1 + o(1))\frac{n}{\ln n}, \tag{1.3}$$

which supports the random graph intuition. Our proof is not based on the potential function technique of Beck, rather on the analysis of a quite natural strategy of Maker, involving a delicate inductive argument.

It is well-known from the theory of random graphs that several natural graph properties, like hamiltonicity (i.e., containing a Hamilton cycle), c-connectivity (i.e., after removing any set of $c-1$ vertices the resulting graph is still connected), or minimum degree at least c, for constant c, all have the same sharp threshold edge number $\frac{1}{2}n \ln n$. In an intuitive language this says that the main reason a random graph is not connected (not hamiltonian, respectively) is that it contains a vertex of degree zero.

We let $\mathcal{T}$, $\mathcal{H}$, $\mathcal{C}_c$ and $\mathcal{D}_c$ denote the graph properties connectivity, hamiltonicity, c-connectivity and minimum degree at least c, respectively, and for every property $\mathcal{P} \in \{\mathcal{H}, \mathcal{C}_c, \mathcal{D}_c\}$ we let – analogously to $b_{\mathcal{T}}$ – $b_{\mathcal{P}}$ denote the largest bias b such that Maker, taking one edge per turn, has a strategy to build a graph with property $\mathcal{P}$, playing against a Breaker with bias b. Note that $\mathcal{T} = \mathcal{C}_1$. Moreover, the aforementioned result of Chvátal and Erdős that Breaker, playing with a bias $(1 + o(1))\frac{n}{\ln n}$, can occupy all edges incident to some vertex, implies that

$$b_{\mathcal{D}_1} \leq (1 + o(1))\frac{n}{\ln n}. \tag{1.4}$$

Krivelevich and Szabó [59] established that $(\ln 2 - o(1))\frac{n}{\ln n}$ is a lower bound for $b_{\mathcal{H}}, b_{\mathcal{C}_c}$, and $b_{\mathcal{D}_c}$. Motivated by the extremely tight correlation between the appearance of the properties $\mathcal{T}$ and $\mathcal{D}_1$ in the random graph, they conjectured that $b_{\mathcal{T}} = b_{\mathcal{D}_1}$ for all n large enough. While this conjecture is still open, (1.3), together with (1.4) and the inequality $b_{\mathcal{T}} \leq b_{\mathcal{D}_1}$, does establish its asymptotic correctness.

Finally, we improve the lower bound of Krivelevich and Szabó for the family $\mathcal{D}_c$, provided c is an arbitrary constant. We show that

$$b_{\mathcal{D}_c} = (1 + o(1))\frac{n}{\ln n},$$

which means that the random graph intuition is valid asymptotically for the minimum-degree-c game as well. In Chapter 2 we generalize this

result to the case where c is upper bounded by a slowly growing function in n (which is around $\ln \ln n$). Furthermore, we show how Maker can merge his strategies for constructing a spanning tree and a graph of minimum degree c to obtain a strategy which allows him to accomplish both of these goals at the same time.

The results presented in Chapter 2 are joint work with Tibor Szabó [49].

1.3 On the Clique-Game

The q-clique game is a widely studied graph game (recall that in this game Maker's goal is to build a clique of size q). An immediate question is how large q can be (in terms of n) such that Maker can achieve a K_q in the game on K_n. Remarkably, for the ordinary $(1:1)$ game the exact solution to this question is known! The following theorem is due to Beck (log stands for the binary logarithm).

Theorem 1.1. *(Theorem 6.4, [21]) The largest q such that Maker has a winning strategy in the $(1:1)$ q-clique game is $q = \lfloor 2\log n - 2\log\log n + 2\log e - 3 + o(1) \rfloor$.*

As pointed out in Subsection 1.1.2, Theorem 1.1 supports the random graph intuition. In Chapter 3 we study three variations of the clique game.

The Biased Game

We call an $(m:b)$ game *biased* if $m \neq 1$ or $b \neq 1$ (i.e., if at least one of the players claims more than one edge per turn). In contrast to the $(1:1)$ clique game, which has been analyzed in depth, for the biased clique game not so much is known.

Let $f_n(m,b)$ denote the largest q such that Maker can occupy a K_q in the $(m:b)$ game on K_n. The random graph intuition suggests that $f_n(m,b)$ is roughly $\frac{2}{\log(m+b)-\log m}\log n$ (see, e.g., [27]). Beck formulated the following open problem which, for the case where m and b are constants, reads as follows.

Open Problem 1.2. *(Open Problem 30.2, [21])*
Let $g_n(m,b) = \left(\frac{2}{\log(m+b)-\log m} + o(1)\right)\log n$. Is it true that $f_n(m,b) = g_n(m,b)$?

We remark that in [21] the function g was specified more precisely. Note that by Theorem 1.1 we have $f_n(1,1) = g_n(1,1)$ for n large enough. Moreover, Beck [21] proved that $f_n(m,1) \geq g_n(m,1)$ for every m and every large enough n.

We will show that for infinitely many m, b the values $f_n(m,b)$ and $g_n(m,b)$ are substantially different. To this end we prove that for every constants m, b, in the $(m:b)$ game Maker has a strategy to occupy a K_q with $q = \left(\frac{m}{\log(b+1)} - o(1)\right) \cdot \log n$. This yields that for constant $m \geq 6$ and large enough n,

$$f_n(m,m) \geq \left(\frac{m}{\log(m+1)} - o(1)\right) \log n > g_n(m,m) = (2 + o(1)) \log n.$$

Building a Clique Fast

So far we considered the issue of building a q-clique on a complete graph containing as few vertices as possible (in terms of q). Another issue is to build a clique *fast*.

Open Problem 1.3. *(Open Problem 25.1, [21]) Playing the* $(1:1)$ *game on the infinite complete graph* K_∞ *(or at least on a "very large" finite* K_n*), how long does it take for Maker to build a* K_q*?*

Let $s(q)$ denote the minimum number of moves Maker needs to build a K_q. Theorem 1.1 implies that Maker can build a K_q on K_n with $n = (1 + o(1))q2^{\frac{q}{2}}$. Hence, $s(q) \leq \frac{1}{2}\binom{n}{2} \leq q^2 2^q$ for large enough q. The best known upper bound on $s(q)$ is $s(q) \leq 2^{q+2}$, which has been discovered by Beck [20] and, independently, by Pekeč [65]. From the other side, Breaker can prevent Maker from building a K_q in $2^{\frac{q}{2}}$ moves, provided q is large enough [20]; thus $s(q) \geq 2^{\frac{q}{2}}$. Beck asks whether the upper bound $O(2^q)$ can be improved. We will show that $s(q) \leq 2^{\frac{2q}{3}} \mathsf{poly}(q)$, where $\mathsf{poly}(q)$ is some polynomial in q.

Building a Tournament

A third variation of the q-clique game is the so called *q-tournament game* (sometimes abbreviated by *tournament game*). A *tournament* is a directed graph where every pair of vertices is connected by a single directed edge. The q-tournament game is played on K_n. At the beginning Breaker fixes an arbitrary tournament T_q on q vertices. Maker and Breaker then alternately take turns in claiming one unclaimed edge e and selecting

one of the two possible orientations. Maker wins if his graph contains a copy of the goal tournament T_q; otherwise Breaker wins [1]. Note that a winning strategy for Breaker in the q-clique game allows him to prevent Maker from achieving any tournament T_q on q vertices. Hence, Theorem 1.1 yields that for $q = (2 - o(1)) \log n$, Breaker has a winning strategy in the q-tournament game. From the other side, Beck [21] showed that Maker has a winning strategy for $q = (\frac{1}{2} - o(1)) \log n$. Actually, he even proved the stronger statement that Maker has a strategy to achieve that his graph contains a copy of all possible T_q.

The random graph intuition suggests that Maker already has a winning strategy if $q = (1 - o(1)) \log n$. Beck [21] included the following open problem in his list of the *seven most humiliating open problems* of the field of combinatorial games.

Open Problem 1.4. *Is it true that Maker has a winning strategy in the q-tournament game for $q = (1 - o(1)) \log n$?*

We will prove that the answer to Open Problem 1.4 is "yes".

1.4 Size Ramsey Number of Bounded Degree Graphs for Games

In Chapter 4 we study graph games which are not played on a complete graph but on some *sparse graph*. In our main contribution we show that for every graph G on n vertices with maximum degree d there is a graph H with at most cn edges (where c is a constant depending on d but not on n) such that Maker has a strategy to occupy a copy of G in the $(1 : 1)$ game on H. This is a result about a game-theoretic variant of the Ramsey numbers, which we describe in the following.

Ordinary Ramsey Numbers and a Game-Theoretic Variant The Ramsey number $r(G)$ of a graph G is the smallest number N such that in any two-coloring of the edges of the complete graph K_N there is a monochromatic copy of G. For example, we clearly have that for every subgraph

[1] We note that here the orientations chosen by Breaker are irrelevant. The rule that Breaker also has to orient his edges is convenient when studying adapted versions of the tournament game; e.g: At the beginning Breaker fixes T_q and arbitrarily colors its edges with red and blue. Then Maker and Breaker take turns, as described above. Additionally, an edge is colored red if it has been claimed by Maker, and blue if it has been claimed by Breaker. Maker wins if there is a (correctly colored) copy of T_q.

H of G, $r(H) \leq r(G)$ since every monochromatic copy of G contains a monochromatic copy of H. Moreover, it is well-known that for the complete graph K_n on n vertices it holds that $\mathsf{poly}(n)2^{\frac{n}{2}} \leq r(K_n) \leq 4^n\mathsf{poly}(n)$. Due to a result of Chvátal, Rödl, Szemerédi and Trotter [36], we have that for every graph G on n vertices of maximum degree d (the class of graphs we are focussing on in Chapter 4), there is a constant c (depending on d but not on n) such that every two-coloring of K_{cn} contains a copy of G; thus $r(G) \leq cn$.

A game-theoretic variant of the Ramsey number was introduced by Beck [12]: Let $r'(G)$ denote the smallest N such that Maker has a strategy to occupy a copy of G in the game on K_N (we assume here that Maker and Breaker each claim one edge per turn). A standard strategy-stealing argument shows that $r'(G) \leq r(G)$. Suppose, for a contradiction, that $r'(G) > r(G)$. Thus, for $N := r(G)$, Breaker has a strategy to prevent Maker from building a copy of G in the game on K_N. But by definition of $r(G)$ this implies that by the end of the game Breaker's graph contains a copy of G. Since Maker starts the game he can now "steal" Breaker's strategy by starting with an arbitrary first move and then following Breaker's strategy (if this strategy calls for something he occupied before he takes an arbitrary edge: no extra move is disadvantageous for him). This enables him to occupy a copy of G.

Hence, due to the result of Chvátal, Rödl, Szemerédi and Trotter, we immediately obtain that for every graph G of maximum degree d we have that $r'(G) \leq r(G) \leq c(d)n$. Further progress was made when Beck [18] showed that if $N \geq \mathsf{poly}(d)3^d \cdot n$ then in the game on K_N Maker has a strategy to create a *universal* graph for the class of graphs on n vertices of maximum degree d, i.e., a graph that contains all such graphs.

Size Ramsey Numbers and a Game-Theoretic Variant A graph H is called *G-Ramsey* if every two-coloring of the edges of H contains a monochromatic copy of G. The size Ramsey number $\hat{r}(G)$, introduced by Erdős, Faudree, Rousseau and Schelp [41], is the smallest number M such that there exists a graph H with M edges where H is G-Ramsey. It is known that $\hat{r}(K_n) = \binom{r(K_n)}{2}$, which means that among all K_n-Ramsey graphs, the complete graph $K_{r(K_n)}$ minimizes both the number of vertices and the number of edges. Naturally, for sparse graphs the situation is quite different: For many sparse graphs G (as cycles and trees of fixed maximum degree) it has been proved that $\hat{r}(G)$ is linear in n [14, 45, 51]. From the other side, Rödl and Szemerédi [68] showed

that there exists a graph G on n vertices of maximum degree 3 where

$$\hat{r}(G) \geq cn \log^{\frac{1}{60}} n,$$

ruling out the possibility, raised by Beck and Erdős (see [35]), that for every d there is a constant $c = c(d)$ such that for any graph G on n vertices of maximum degree d, $\hat{r}(G) \leq cn$.

The best known upper bound for the class of *all* graphs with constant maximum degree is due to Kohayakawa, Rödl, Schacht and Szemerédi [56], who derived that for every natural number d there exists a constant $c = c(d)$ such that for every graph G on n vertices of maximum degree d,

$$\hat{r}(G) \leq cn^{2-\frac{1}{d}} \log^{\frac{1}{d}} n. \tag{1.5}$$

Similarly as for the ordinary Ramsey number, there is a game-theoretic variant of the size Ramsey number: For every graph G we let $\hat{r}'(G)$ denote the smallest M such that there exists a graph H with M edges such that Maker has a strategy to occupy a copy of G in the game on H. By a similar strategy stealing argument as above we get that $\hat{r}'(G) \leq \hat{r}(G)$.

We investigate $\hat{r}'(G)$ for graphs G with constant maximum degree d. Clearly, for d-regular graphs G where $d \geq 1$ we have that $\hat{r}'(G)$ is at least linear in n. From the other side, (1.5) has so far been the best known upper bound (to our knowledge) for $\hat{r}'(G)$. We close this gap by showing that $\hat{r}'(G) \leq cn$ for some constant c depending on d but not on n.

Related Work Feldheim and Krivelevich [40] studied a slightly different problem: For a given graph G and some $N \geq r'(G)$ they investigated the minimum number of moves Maker needs to build a copy of G in the game on K_N (note that due to the definition of $r'(G)$ this number is at most $\binom{N}{2}$). They found, using a powerful theorem of Alon, Krivelevich, Spencer and Szabó about discrepancy games, that if G is a sparse graph on n vertices and N is large enough then Maker can succeed in a linear number of rounds (in n). More precisely, they showed that for every integer d there are constants $c = c(d), c' = c'(d)$ such that for every graph G on n vertices of maximum degree d and every $N > cn$, Maker has a strategy to occupy a copy of G in the game on K_N in at most $c'n$ rounds. We remark that they actually proved this result for the more general class of d-degenerate graphs. The values of the constants are $c = d^{11}2^{2d+9}$ and $c' = d^{11}2^{2d+7}$.

Their proof contains quite a few concepts and features which are very helpful for our problem. So in our construction we will use many ingredients of their approach.

1.5 Special Trees With Implications to SAT and Games

In Chapter 5 we construct a special class of binary trees, which have implications on games and SAT.

1.5.1 Trees

By a *binary tree* we mean a rooted tree where every node has either two or no children. We say that a leaf v of a binary tree is *l-close* to a node w if w is an ancestor of v, at distance at most l from v. For positive integers k and d, we call a binary tree T a *(k,d)-tree* if (i) every leaf has depth at least k and (ii) for every node u of T there are at most d leaves that are k-close to u. Clearly, every binary tree with all leaves at depth at least k is a $(k,2^k)$-tree. The following will be the main ingredient in proving some new results on games and SAT.

$$A\left(k,\left(\frac{2}{e}+O(1/\sqrt{k})\right)\frac{2^k}{k}\right)\text{-tree exists.} \tag{1.6}$$

1.5.2 Implications to Games

Here we regard the very general class of games introduced at the beginning of this chapter: For a set X and some family $\mathcal{F}\subseteq 2^X$ of winning sets we denote by $(X,\mathcal{F})$ the game where Maker and Breaker alternately claim an element of X, and Maker's goal is to completely occupy a winning set. Since a *hypergraph* $\mathcal{G}$ is defined as a pair (V,E), where V is a finite set whose elements are called *vertices* and E is a family of subsets of V, called *hyperedges*, we can consider every game $(X,\mathcal{F})$ as a hypergraph. This will simplify our notation.

A hypergraph is *k-uniform* if every hyperedge contains exactly k vertices. The famous Erdős-Selfridge Theorem [42] (c.f. Subsection 1.1.1) states that for each k-uniform hypergraph with less than 2^{k-1} hyperedges Breaker has a winning strategy. It can be shown that this upper bound on the number of hyperedges is best possible. Beck conjectured that

furthermore a similar phenomenon as the Lovász Local Lemma holds for games.

Neighborhood Conjecture. *(Open Problem 9.1(a), [21]) Assume that $\mathcal{G}$ is a k-uniform hypergraph where every hyperedge intersects at most $2^{k-1}-1$ other hyperedges. Is it true that Breaker has a winning strategy on $\mathcal{G}$?*

The best known result in the direction of the Neighborhood Conjecture is that if every vertex of a k-uniform hypergraph occurs in at most $\lfloor \frac{k}{2} \rfloor$ hyperedges then Breaker has a winning strategy. The proof uses Hall's Theorem to assign two of its vertices to every hyperedge such that no vertex is assigned to more than one hyperedge. If Breaker pairs the two corresponding vertices (i.e., whenever Maker claims a vertex, he takes the vertex assigned to the same hyperedge) he can occupy at least one vertex in every hyperedge, thus he has a winning strategy.

We now describe how an appropriate (k,d)-tree yields a counterexample to the Neighborhood Conjecture. With every $(k-1,d)$-tree T we associate the k-uniform hypergraph $\mathcal{G}_T$ whose vertices are the nodes of T and whose hyperedges are the vertex sets of paths that start at a leaf and go up $k-1$ levels. Figure 1.2 depicts such a hypergraph. By first claiming the root and then pairing every node with its sibling (i.e. the node having the same parent) Maker can finally occupy all vertices on some path from the root to a leaf, which by assumption contains a hyperedge. Thus, Maker has a winning strategy on $\mathcal{G}_T$. Moreover, it can be seen that by construction, every node occurs in at most d hyperedges. Hence every hyperedge intersects at most dk other hyperedges. By choosing T to be the tree from (1.6) we obtain that every hyperedge of $\mathcal{G}_T$ intersects at most $\left(\frac{2}{e}+o(1)\right)\frac{2^{k-1}}{k-1}k \leq (1+o(1))\,\frac{2^k}{e}$ other hyperedges, refuting the Neighborhood Conjecture. The construction of $\mathcal{G}_T$ relies on the proof that suitable (k,d)-trees exists, which is rather involved and long. So we will additionally give a second counterexample for the Neighborhood Conjecture which is slightly weaker (in the sense that the hypergraph has more hyperedges) but can be obtained by a simpler construction.

1.5.3 Implications to SAT

The satisfiability of Boolean formulas is the archetypical NP-hard problem. Following the notation introduced at the beginning of this chapter, a k-CNF formula is a conjunction of clauses that are the disjunction of exactly k distinct literals. The problem of deciding whether a k-CNF

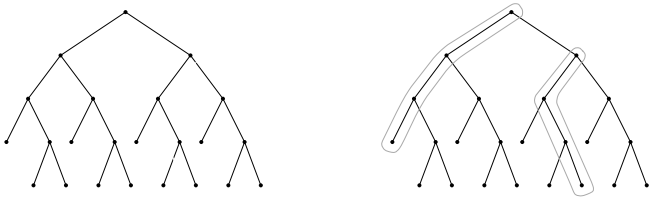


Figure 1.2: A $(3, 6)$-tree T (on the left) and some exemplary hyperedges of $\mathcal{G}_T$ (on the right).

formula is satisfiable is denoted by k-SAT, it is solvable in polynomial time for $k = 2$, and is NP-complete for every $k \geq 3$ as shown by Cook [37].

Papadimitriou and Yannakakis [64] have shown that k-SAT is even MAX-SNP-complete for every $k \geq 2$.

The first level of difficulty in satisfying a CNF formula arises when two clauses share at least one variable. For a finer view into the transition to NP-hardness, a grading of the class of k-CNF formulas can be introduced, that limits how much clauses interact locally. A k-CNF formula is called a (k, s)-CNF formula if every variable appears in at most s clauses. The problem of satisfiability of (k, s)-CNF formulas is denoted by (k, s)-SAT.

Hardness Jump

Tovey [76] proved that while every $(3, 3)$-CNF formula is satisfiable (due to Hall's Theorem), the problem of deciding whether a $(3, 4)$-CNF formula is satisfiable is already NP-hard. Dubois [39] showed that $(4, 6)$-SAT and $(5, 11)$-SAT are also NP-complete.

Kratochvíl, Savický, and Tuza [58] defined the value $f(k)$ to be the largest integer s such that every (k, s)-CNF formula is satisfiable. They also generalized Tovey's result by showing that for every $k \geq 3$, $(k, f(k)+1)$-SAT is already NP-complete. In other words, for every $k \geq 3$ the (k, s)-SAT problem goes through a kind of "complexity phase transition" at the value $s = f(k)$. On the one hand the $(k, f(k))$-SAT problem is trivial by definition in the sense that every instance of the problem is a "YES"-instance. On the other hand the $(k, f(k) + 1)$-SAT problem is already NP-hard, so the problem becomes hard from being trivial just by

allowing one more occurrence of each variable. For large values of k this might seem astonishing, as the value of the transition is exponential in k: one might think that the change of just one in the parameter should have hardly any effect.

The complexity hardness jump is even greater: the problem of (k, s)-SAT is also MAX-SNP-complete for every $s > f(k)$ as was shown by Berman, Karpinski, and Scott [24] (generalizing a result of Feige [44] who showed that $(3, 5)$-SAT is hard to approximate within a certain constant factor).

The determination of where this complexity hardness jump occurs is one of the main topics of Chapter 5.

Known Bounds

For a lower bound the best tool available is the Lovász Local Lemma (c.f. Subsection 1.1.1). We call a pair of clauses sharing at least one variable an *intersecting pair*, and for a clause C of a CNF formula $\mathcal{F}$ we denote by *the neighborhood* $\Gamma(C)$ of C the set of other clauses (excluding C itself) intersecting C. A straightforward consequence of the Local Lemma states that if $|\Gamma(C)| \leq 2^k/e - 1$ for every clause C of a k-CNF formula $\mathcal{F}$ then $\mathcal{F}$ is satisfiable. A natural question is how tight this bound is: Analogously to $f(k)$ let $l(k)$ denote the largest integer r such that every k-CNF formula $\mathcal{F}$ for which $|\Gamma(C)| \leq r$, for every clause C of $\mathcal{F}$, is satisfiable. With this notation the Local Lemma implies that

$$l(k) \geq \left\lfloor \frac{2^k}{e} \right\rfloor - 1. \tag{1.7}$$

The order of magnitude of this bound is trivially optimal: $l(k) < 2^k - 1$ follows from the unsatisfiable k-CNF formula consisting of all possible k-clauses on only k variables.

In [48] a hardness jump is proved for the function l: deciding the satisfiability of k-CNF formulas with maximum neighborhood size at most $\max\{l(k) + 2, k + 3\}$ is NP-complete.

As observed by Kratochvíl, Savický and Tuza [58] the bound (1.7) immediately implies

$$f(k) \geq \left\lfloor \frac{l(k)}{k} \right\rfloor + 1 \geq \left\lfloor \frac{2^k}{ek} \right\rfloor. \tag{1.8}$$

From the other side Savický and Sgall [70] established that $f(k) = O\left(k^{0.74} \cdot \frac{2^k}{k}\right)$. This was improved by Hoory and Szeider [53] who came

within a logarithmic factor: $f(k) = O\left(\log k \cdot \frac{2^k}{k}\right)$, which is the previously best known upper bound.

Our Contribution

We determine the asymptotics of $f(k)$, thereby settling some questions from [48]. We show that the lower bound (1.8) can be strengthened by a factor of 2 and that this bound is tight. That is,

$$f(k) = \left(\frac{2}{e} + O\left(\frac{1}{\sqrt{k}}\right)\right)\frac{2^k}{k}. \tag{1.9}$$

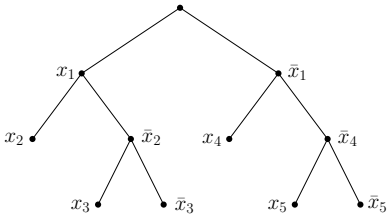
We quickly describe how a (k,d)-tree yields an unsatisfiable (k,d)-CNF formula: Let T be a (k,d)-tree and suppose that we assign to every non-leaf node w of T a variable x_w, and label one of its children with the literal x_w and the other with the negated version $\bar{x}_w$. With every leaf v of T we then associate a clause C_v by walking along a path of length $k-1$ from v towards the root and taking the disjunction of all labels encountered on this path (i.e., the labels of all nodes to which v is $(k-1)$-close). Finally we let $\mathcal{F}_T$ denote the conjunction of all such clauses C_v. Figure 1.3 gives an illustration. $\mathcal{F}_T$ is unsatisfiable, for if an assignment α is given, it defines a path from the root to a leaf, say v, by always proceeding to the unique child whose label is mapped to false by α; thus C_v is violated by α. Moreover, the defining property of (k,d)-trees guarantees that no variable appears in more than d clauses. Hence $\mathcal{F}_T$ is an unsatisfiable (k,d)-CNF formula and therefore (1.6) gives the upper bound in (1.9).

Since the Local Lemma was fully algorithmized by Moser and Tardos [62] we now have that not only every (k,s)-CNF formula for $s = (1 + o(1))\frac{2^{k+1}}{ek}$ has a satisfying assignment but there is also an algorithm that *finds* such an assignment in probabilistic polynomial time. Moreover, for just a little bit larger value of the parameter s one cannot find a satisfying assignment efficiently, simply because already the decision problem is NP-hard.

Our construction also shows that the lower bound (1.7) on $l(k)$ is asymptotically tight:

$$l(k) = \left(\frac{1}{e} + O\left(\frac{1}{\sqrt{k}}\right)\right)2^k.$$

The results in Chapter 5 are obtained jointly with Tibor Szabó and Gábor Tardos [50].



$$\mathcal{F}_T = (x_2 \vee x_1) \wedge (x_3 \vee \bar{x}_2) \wedge (\bar{x}_3 \vee \bar{x}_2) \wedge (x_4 \vee \bar{x}_1) \wedge (x_5 \vee \bar{x}_4) \wedge (\bar{x}_5 \vee \bar{x}_4)$$

Figure 1.3: A $(2,3)$-tree T and $\mathcal{F}_T$.

1.6 Notation

Throughout this thesis log denotes the binary logarithm and ln denotes the natural logarithm. Ceiling and floor signs are routinely omitted whenever they are not crucial for clarity. Let G be a graph and let $V(G)$ and $E(G)$ denote the set of vertices and the set of edges of G, respectively. For $U \subseteq V(G)$, $E_G(U)$ denotes the set of edges spanned by U and, similarly, for disjoint subsets $U, W \subseteq V(G)$, $E_G(U,W)$ denotes the set of edges with one endpoint in U and the other in W. When there is no danger of confusion we sometimes omit the index G. Furthermore, $G\backslash U$ denotes the graph obtained by taking G and deleting all the vertices of U. The *subgraph induced by* U, $G[U]$, denotes the graph obtained from deleting all vertices of $V(G)\backslash U$ in G.

2

Verifying the Random Graph Intuition

2.1 Introduction

Recall that $b_{\mathcal{T}}$ ($b_{\mathcal{D}_c}$ respectively) denotes the largest bias b of Breaker such that Maker, taking one edge in each turn, can occupy a spanning tree (a graph with minimum degree c, respectively) while Breaker takes b edges in each turn. Due to a result of Chvátal and Erdős [34],

$$b_{\mathcal{D}_1} \leq \frac{n}{\ln n} + O\left(\frac{n \ln\ln n}{\ln^2 n}\right) = (1+o(1))\frac{n}{\ln n}. \tag{2.1}$$

Clearly, $b_{\mathcal{T}} \leq b_{\mathcal{D}_1}$. Hence (2.1) yields that $b_{\mathcal{T}} \leq (1+o(1))\frac{n}{\ln n}$. In our first result we show that this bound is asymptotically optimal.

Theorem 2.1. *Maker has a strategy to build a spanning tree while playing against a Breaker with bias* $b := (\ln n - \ln\ln n - 6)\frac{n}{\ln^2 n}$*, provided* n *is large enough.*

The constant 6 in the error term could be improved somewhat, but we do not know whether the second order term is best possible.

(2.1) gives that $b_{\mathcal{D}_c} \leq (1+o(1))\frac{n}{\ln n}$ for every $c \geq 1$. We establish that $b_{\mathcal{D}_c} = (1+o(1))\frac{n}{\ln n}$ for every constant c, which means that the random graph intuition is valid asymptotically for the minimum-degree-c game as well.

Theorem 2.2. *Let $c = c(n) < \frac{\ln \ln n}{3}$. Maker has a strategy to build a graph with minimum degree at least c while playing against Breaker with bias $b := (\ln n - \ln \ln n - (2c+3))\frac{n}{\ln^2 n}$, provided n is large enough.*

As a third example of the surprising validity of the random graph intuition, for any constant $\epsilon > 0$ there is a $\delta = \delta(\epsilon) > 0$ such that Maker is able to build a graph with minimum-degree at least $\delta \ln(n)$ while playing against a Breaker's bias $(1-\epsilon) \cdot \frac{n}{\ln(n)}$.

Theorem 2.3. *Let $\epsilon > 0$ be a constant. Then Maker has a strategy to build a graph with minimum degree at least $\frac{\epsilon}{3(1-\epsilon)} \ln n$ while playing against a Breaker's bias of $(1-\epsilon) \cdot \frac{n}{\ln n}$.*

The order $\ln n$ for the largest achievable minimum degree against a bias of $(1-\epsilon)\frac{n}{\ln n}$ is obviously best possible: Maker has at most $\frac{\binom{n}{2}}{b+1}$ edges by the end, which allows a minimum degree at most $\frac{\ln n}{1-\epsilon}$.

Finally, by merging the strategies of Maker for achieving a spanning tree and a graph of minimum degree c, it can be proven that Maker has a strategy to accomplish both of these goals at the same time.

Theorem 2.4. *Let $c = c(n) < \frac{\ln \ln n}{3}$. Maker has a strategy to build a connected graph with minimum degree c while playing against Breaker with bias $b := (\ln n - \ln \ln n - (2c+5))\frac{n}{\ln^2 n}$, provided n is large enough.*

As in the minimum-degree case we can show that Theorem 2.4 remains true if we let $\epsilon > 0$ be a constant and replace c by $\delta \ln(n)$ with $\delta > 0$ being a constant depending on ϵ only.

Theorem 2.5. *Let $\epsilon > 0$ be a constant. Then Maker has a strategy to build a connected graph with minimum degree $\frac{\epsilon}{3(1-\epsilon)} \ln n$ while playing against a Breaker's bias of $(1-\epsilon) \cdot \frac{n}{\ln n}$.*

Notation We denote by H_s the sth harmonic number $\sum_{j=1}^{s} \frac{1}{j}$ and often use the well-known fact that for every positive integer s

$$\ln s \leq H_s \leq \ln s + 1. \tag{2.2}$$

2.2 Building a Spanning Tree

Proof of Theorem 2.1: An important message of Beck's potential function argument (which allowed him to prove $b_{\mathcal{T}} \geq (\ln 2 - o(1))\frac{n}{\ln n}$) was that instead of concentrating on "making" a spanning tree, one concentrates on "breaking" into every cut – the success of which would then imply connectivity. In our proof we abandon this dual approach and plainly focus on the original goal: building a spanning tree.

We assume that Breaker starts the game. Otherwise Maker can start with an arbitrary first move, then follow his strategy. If his strategy calls for something he occupied before he takes an arbitrary edge; no extra move is disadvantageous for him.

In the following proof by a *component* we always mean a connected component of Maker's graph. For a vertex v, we denote by $C(v)$ the component containing v. We call a component *dangerous* if it contains at most $2b$ vertices. By the *degree of a vertex* v (or $\deg(v)$ in short) we always mean the degree of v in Breaker's graph.

We define a *danger function* on the vertex set of K_n. Let

$$\mathsf{dang}(v) = \begin{cases} \deg(v) & \text{if } C(v) \text{ is dangerous} \\ -1 & \text{otherwise} \end{cases}$$

Maker's Strategy At the beginning every vertex is *active*. For his ith move, Maker identifies a vertex v_i with the largest danger value among active vertices (ties are broken arbitrarily) and he occupies one arbitrary free edge connecting $C(v_i)$ to another component. He deletes v_i from the set of active vertices and calls v_i *deactivated*. This strategy of Maker will be denoted by S_M.

The following observation follows easily from S_M by induction on the number of rounds.

Observation 2.6. *Every component contains exactly one active vertex.* □

Note that due to Observation 2.6 Maker can always make a move according to S_M unless his graph is a tree (thus he won) or Breaker occupied a cut (and Breaker won). Hence during our analysis Maker's graph is always a forest.

Proof of Maker's Win Suppose, for a contradiction, that Breaker has a strategy S_B to win the $(1 : b)$ connectivity game against Maker. Let

B_i and M_i denote the ith move of Breaker and Maker, respectively, in the game where they play against each other using their respective strategies S_B and S_M. Let g be the length of this game, i.e., g is the smallest integer such that Breaker finished occupying all edges in a cut $(K, V \setminus K)$ in move B_g. We call this the end of the game. Note that $g \leq n-1$, as Maker's strategy does not allow him to occupy a cycle.

Let $|K| \leq |V \setminus K|$. Observe that $|K| \leq 2b$, since otherwise Breaker would have had to occupy at least $2b(n-2b) > gb$ edges in $g < n$ rounds, a contradiction for large n. This implies that during the game there is always at least one dangerous component. Since Maker's strategy prefers to deactivate active vertices in dangerous components we also have the following.

Observation 2.7. *Vertex v_i is in a dangerous component at and before its deactivation.* □

In his last move Breaker takes b edges to completely occupy all edges between K and $V \setminus K$. In order to be able to do that, directly before Breaker's last move all vertices of K must have degree at least $n-2b-b$. Let $v_g \in K$ be an arbitrary active vertex; by Observation 2.6 there is one in each component inside K.

Recall that $v_1, \ldots, v_{g-1}$ were defined during the game. For $0 \leq i \leq g-1$, let $I_i = \{v_{g-i}, \ldots v_g\}$. For a non-empty subset $I \subseteq V$, let $\overline{\mathsf{dang}}_{B_i}(I) = \frac{\sum_{v \in I} \mathsf{dang}(v)}{|I|}$ denote the average danger value of the vertices in I directly before move B_i of Breaker. Analogously, $\overline{\mathsf{dang}}_{M_i}(I)$ denotes the average danger value before M_i.

The following lemma is a consequence of Maker's strategy. It considers the change of danger during Maker's move.

Lemma 2.8. *For every i, $1 \leq i \leq g-1$, directly before M_{g-i} we have that $\overline{\mathsf{dang}}_{M_{g-i}}(I_i) \geq \overline{\mathsf{dang}}_{B_{g-i+1}}(I_{i-1})$.*

Proof: All the vertices $v_{g-i+1}, \ldots, v_g$ constituting I_{i-1} are in a dangerous component directly before B_{g-i+1}, so their danger value does not change during M_{g-i}. Hence $\overline{\mathsf{dang}}_{M_{g-i}}(I_{i-1}) = \overline{\mathsf{dang}}_{B_{g-i+1}}(I_{i-1})$. Maker deactivated v_{g-i} in M_{g-i}, because its danger was maximum among active vertices. All vertices of I_{i-1} were still active before M_{g-i}, thus $\mathsf{dang}(v_{g-i}) \geq \max\{\mathsf{dang}(v_{g-i+1}), \ldots, \mathsf{dang}(v_g)\}$ implying $\overline{\mathsf{dang}}_{M_{g-i}}(I_i) \geq \overline{\mathsf{dang}}_{M_{g-i}}(I_{i-1})$ and the lemma follows. □

The following lemma bounds the change of the danger value during Breaker's moves. The first estimate is used during the first part of the

game. It will guarantee the existence of many vertices with large average degree, which eventually leads to a contradiction. For the rounds closer to the end we need a stronger inductive statement, which is provided by the second estimate of the lemma.

Lemma 2.9. *Let i be an integer, $1 \leq i \leq g-1$.*

(*i*) $\overline{\mathsf{dang}}_{M_{g-i}}(I_i) - \overline{\mathsf{dang}}_{B_{g-i}}(I_i) \leq \frac{2b}{i+1}$

(*ii*) $\overline{\mathsf{dang}}_{M_{g-i}}(I_i) - \overline{\mathsf{dang}}_{B_{g-i}}(I_i) \leq \frac{b+i+a(i-1)-a(i)}{i+1}$, *where $a(i)$ denotes the number of edges spanned by I_i which Breaker took in the first $g-i-1$ rounds.*

Proof: All the components $C(v_{g-i}), \ldots, C(v_g)$ are dangerous before M_{g-i}. Since components do not change during Breaker's move the danger value of the vertices of I_i depend solely on their degrees. In B_{g-i} Breaker claims b edges, so the increase of the sum of degrees of $v_{g-i}, \ldots, v_g$ during B_{g-i} is at most $2b$. Hence $\overline{\mathsf{dang}}(I_i)$ increases by at most $\frac{2b}{i+1}$, which proves (*i*).

For (*ii*), we will be more careful. Let e_{double} denote the number of edges taken in B_{g-i} whose both endpoints are in I_i. Then the increase of $\sum_{j=0}^{i} \deg(v_{g-j})$ during B_{g-i} is at most $b + e_{\text{double}}$. Hence $\overline{\mathsf{dang}}(I_i)$ increases by at most $\frac{e_{\text{double}}+b}{i+1}$. We now bound e_{double}. By definition, Breaker occupied $a(i)$ edges spanned by I_i in his first $g-i-1$ moves. So, all in all, Breaker occupied $a(i) + e_{\text{double}}$ edges spanned by I_i in his first $g-i$ moves. On the other hand, we know that among these edges exactly $a(i-1)$ are spanned by $I_{i-1} = I_i \setminus \{v_{g-i}\}$ and there are at most i edges in I_i incident to v_{g-i}. Hence $a(i) + e_{\text{double}} \leq a(i-1) + i$, giving us $e_{\text{double}} \leq i + a(i-1) - a(i)$. □

Using Lemmas 2.8 and 2.9 we will derive that before B_1, $\overline{\mathsf{dang}}(I_{g-1}) > 0$. This is of course in contradiction with the fact that at the beginning of the game every vertex has danger value 0.

Let $k := \lfloor \frac{n}{\ln n} \rfloor$. For the analysis, we split the game into two parts: The main game, and the end game consisting of the last k rounds.

Recall that the danger value of v_g directly before B_g is at least $n - 3b$.

Assume first that $k > g$.

$$\begin{aligned}\overline{\mathsf{dang}}_{B_1}(I_{g-1}) &= \overline{\mathsf{dang}}_{B_g}(I_0) \\ &\quad + \sum_{i=1}^{g-1}\left(\overline{\mathsf{dang}}_{M_{g-i}}(I_i) - \overline{\mathsf{dang}}_{B_{g-i+1}}(I_{i-1})\right) \\ &\quad - \sum_{i=1}^{g-1}\left(\overline{\mathsf{dang}}_{M_{g-i}}(I_i) - \overline{\mathsf{dang}}_{B_{g-i}}(I_i)\right).\end{aligned}$$

Hence,

$$\begin{aligned}\overline{\mathsf{dang}}_{B_1}(I_{g-1}) &\geq n - 3b + \sum_{i=1}^{g-1} 0 - \sum_{i=1}^{g-1}\frac{b+i+a(i-1)-a(i)}{i+1} \\ &\geq n - 3b - b(H_g - 1) - (g-1) - \frac{a(0)}{2} \\ &\quad + \sum_{i=1}^{g-2}\frac{a(i)}{(i+2)(i+1)} + \frac{a(g-1)}{g} \\ &\geq n - b(H_g + 2) - g \qquad [\text{since } a(0) = 0 \text{ and } a(i) \geq 0] \\ &\geq n - b(\ln k + 3) - k \qquad [\text{since } g \leq k] \\ &\geq n - \frac{n}{\ln n}(\ln n - \ln\ln n + 3) - \frac{n}{\ln n} \\ &\geq \frac{n\ln\ln n}{\ln n} - O\left(\frac{n}{\ln n}\right) \\ &> 0.\end{aligned}$$

Assume now that $k \leq g$. We then have

$$\begin{aligned}\overline{\mathsf{dang}}_{B_1}(I_{g-1}) &= \overline{\mathsf{dang}}_{B_g}(I_0) + \sum_{i=1}^{g-1}\left(\overline{\mathsf{dang}}_{M_{g-i}}(I_i) - \overline{\mathsf{dang}}_{B_{g-i+1}}(I_{i-1})\right) \\ &\quad - \sum_{i=1}^{k-1}\left(\overline{\mathsf{dang}}_{M_{g-i}}(I_i) - \overline{\mathsf{dang}}_{B_{g-i}}(I_i)\right) \\ &\quad - \sum_{i=k}^{g-1}\left(\overline{\mathsf{dang}}_{M_{g-i}}(I_i) - \overline{\mathsf{dang}}_{B_{g-i}}(I_i)\right).\end{aligned}$$

Thus,

$$\overline{\mathsf{dang}}_{B_1}(I_{g-1}) \geq n - 3b + \sum_{i=1}^{g-1} 0 - \sum_{i=1}^{k-1}\frac{b+i+a(i-1)-a(i)}{i+1} - \sum_{i=k}^{g-1}\frac{2b}{i+1}$$

$$
\begin{aligned}
&\geq n - 3b - b(H_k - 1) - (k-1) - \frac{a(0)}{2} \\
&\quad + \sum_{i=1}^{k-2} \frac{a(i)}{(i+2)(i+1)} + \frac{a(k-1)}{k} - 2b(H_g - H_k) \\
&\geq n - b(2H_g - H_k + 2) - k \\
&\geq n - b(2\ln n - \ln k + 4) - k \\
&\geq n - \left(\frac{n}{\ln n} - \frac{n \ln\ln n}{\ln^2 n} - 6\frac{n}{\ln^2 n} \right) (\ln n + \ln\ln n + 5) \\
&\quad - \frac{n}{\ln n} \\
&\geq \frac{n(\ln\ln n)^2}{\ln^2 n} \\
&> 0.
\end{aligned}
$$

□

2.3 Achieving Large Minimum Degree

Proof of Theorem 2.2: As in the previous proof we assume that Breaker starts the game. We say that the game ends when either all vertices have degree at least c in Maker's graph (and Maker won) or one vertex has degree at least $n - c$ in Breaker's graph (and Breaker won). With $\deg_M(v)$ and $\deg_B(v)$ we denote the degree of a vertex v in Maker's graph and in Breaker's graph, respectively. A vertex v is called *dangerous* if $\deg_M(v) \leq c - 1$. To establish Maker's strategy we define the *danger value* of a vertex v as $\mathsf{dang}(v) := \deg_B(v) - 2b \cdot \deg_M(v)$.

Maker's Strategy S_M Before his ith move Maker identifies a dangerous vertex v_i with the largest danger value, ties are broken arbitrarily. Then, as his ith move Maker claims an edge incident to v_i. We refer to this step as "easing v_i".

Observation 2.10. *Maker can always make a move according to his strategy unless no vertex is dangerous (thus he won) or Breaker occupied at least $n - c$ edges incident to a vertex (and Breaker won).* □

Observation 2.11. *Vertex v_i is dangerous any time before Maker's ith move.* □

Suppose, for a contradiction, that Breaker, playing with a bias b, has a strategy S_B to win the min-degree-c game against Maker who plays with bias 1. Let B_i and M_i denote the ith move of Breaker and Maker, respectively, in the game where they play against each other using their respective strategies S_B and S_M. Let g be the length of this game, i.e., the maximum degree of Breaker's graph becomes larger than $n-1-c$ in move B_g. We call this the end of the game.

For a set $I \subseteq V$ of vertices we let $\overline{\mathsf{dang}}(I)$ denote the average danger value $\frac{\sum_{v\in I}\mathsf{dang}(v)}{|I|}$ of the vertices of I. When there is risk of confusion we add an index and write $\mathsf{dang}_{B_i}(v)$ or $\mathsf{dang}_{M_i}(v)$ to emphasize that we mean the danger-value of v *directly before* B_i or M_i, respectively.

In his last move Breaker takes b edges to increase the maximum Breaker-degree of his graph to at least $n-c$. In order to be able to do that, directly before Breaker's last move B_g there must be a dangerous vertex v_g whose Breaker-degree is at least $n-c-b$. Thus $\mathsf{dang}_{B_g}(v_g) \geq n-c-b-2b(c-1)$.

Recall that $v_1, \ldots, v_{g-1}$ were defined during the game. For $0 \leq i \leq g-1$, let $I_i = \{v_{g-i}, \ldots v_g\}$.

The following lemma estimates the change in the average danger during Maker's move.

Lemma 2.12. *Let i, $1 \leq i \leq g-1$,*

(i) if $I_i \neq I_{i-1}$, then $\overline{\mathsf{dang}}_{M_{g-i}}(I_i) - \overline{\mathsf{dang}}_{B_{g-i+1}}(I_{i-1}) \geq 0$.

(ii) if $I_i = I_{i-1}$, then $\overline{\mathsf{dang}}_{M_{g-i}}(I_i) - \overline{\mathsf{dang}}_{B_{g-i+1}}(I_{i-1}) \geq \frac{2b}{|I_i|}$.

Proof: For part (i), we have that $v_{g-i} \notin I_{i-1}$. Since danger values do not increase during Maker's move we have $\overline{\mathsf{dang}}_{M_{g-i}}(I_{i-1}) \geq \overline{\mathsf{dang}}_{B_{g-i+1}}(I_{i-1})$. Before M_{g-i} Maker selected to ease vertex v_{g-i} because its danger was highest among dangerous vertices. Since all vertices of I_{i-1} are dangerous before M_{g-i} we have that $\overline{\mathsf{dang}}(v_{g-i}) \geq \max(\mathsf{dang}(v_{g-i+1}), \ldots, \mathsf{dang}(v_g))$, which then implies that $\overline{\mathsf{dang}}_{M_{g-i}}(I_i) \geq \overline{\mathsf{dang}}_{M_{g-i}}(I_{i-1})$. Combining the two inequalities establishes part (i).

For part (ii), we have that $v_{g-i} \in I_{i-1}$. In M_{g-i} $\deg_M(v_{g-i})$ increases by 1 and $\deg_M(v)$ does not decrease for any other $v \in I_i$. Besides, the degrees in Breaker's graph do not change during Maker's move. So $\mathsf{dang}(v_{g-i})$ decreases by $2b$, whereas $\mathsf{dang}(v)$ do not increase for any other vertex $v \in I_i$. Hence $\overline{\mathsf{dang}}(I_i)$ decreases by at least $\frac{2b}{|I_i|}$, which implies (ii). □

The next lemma bounds the change of the danger value during Breaker's moves.

Lemma 2.13. *Let i be an integer, $1 \leq i \leq g-1$.*

(*i*) $\overline{\mathsf{dang}}_{M_{g-i}}(I_i) - \overline{\mathsf{dang}}_{B_{g-i}}(I_i) \leq \frac{2b}{|I_i|}$

(*ii*) $\overline{\mathsf{dang}}_{M_{g-i}}(I_i) - \overline{\mathsf{dang}}_{B_{g-i}}(I_i) \leq \frac{b+|I_i|-1+a(i-1)-a(i)}{|I_i|}$, *where $a(i)$ denotes the number of edges spanned by I_i which Breaker took in the first $g-i-1$ rounds.*

Proof: Let e_{double} denote the number of those edges with both endpoints in I_i which are occupied by Breaker in B_{g-i}. Then the increase of $\sum_{v\in I_i} \deg_B(v)$ during B_{g-i} is at most $b + e_{\text{double}}$. Since the degrees in Maker's graph do not change during Breaker's move the increase of $\overline{\mathsf{dang}}(I_i)$ (during B_{g-i}) is at most $\frac{b+e_{\text{double}}}{|I_i|}$.

Part (*i*) is then immediate after noting that $e_{\text{double}} \leq b$.

For (*ii*), we bound e_{double} more carefully. By definition, Breaker occupied $a(i)$ edges spanned by I_i in his first $g-i-1$ moves. So, all in all, Breaker occupied $a(i) + e_{\text{double}}$ edges spanned by I_i in his first $g-i$ moves. On the other hand, we know that among these edges exactly $a(i-1)$ are spanned by $I_{i-1} \supseteq I_i \setminus \{v_{g-i}\}$ and there are at most $|I_i|-1$ edges in I_i incident to v_{g-i}. Hence $a(i) + e_{\text{double}} \leq a(i-1) + |I_i| - 1$, giving us $e_{\text{double}} \leq |I_i| - 1 + a(i-1) - a(i)$. □

The following estimates for the change of average danger during one full round are immediate corollaries of the previous two lemmas.

Corollary 2.14. *Let i be an integer, $1 \leq i \leq g-1$.*

(*i*) *if $I_i = I_{i-1}$, then* $\overline{\mathsf{dang}}_{B_{g-i}}(I_i) - \overline{\mathsf{dang}}_{B_{g-i+1}}(I_{i-1}) \geq 0$.

(*ii*) *if $I_i \neq I_{i-1}$, then* $\overline{\mathsf{dang}}_{B_{g-i}}(I_i) - \overline{\mathsf{dang}}_{B_{g-i+1}}(I_{i-1}) \geq -\frac{2b}{|I_i|}$

(*iii*) *if $I_i \neq I_{i-1}$, then*

$$\overline{\mathsf{dang}}_{B_{g-i}}(I_i) - \overline{\mathsf{dang}}_{B_{g-i+1}}(I_{i-1}) \geq -\frac{b+|I_i|-1+a(i-1)-a(i)}{|I_i|},$$

where $a(i)$ denotes the number of edges spanned by I_i which Breaker took in the first $g-i-1$ rounds.

Using Corollary 2.14 we derive that before B_1, $\overline{\mathsf{dang}}(I_{g-1}) > 0$, which contradicts the fact that at the beginning of the game every vertex has danger value 0.

Let $k := \lfloor \frac{n}{\ln n} \rfloor$. For the analysis, we split the game into two parts: The main game, and the end game which starts when $|I_i| \leq k$.

Let $|I_g| = r$. Let $i_1 < \ldots < i_{r-1}$ be those indices for which $I_{i_j} \neq I_{i_j - 1}$. Note that $|I_{i_j}| = j + 1$. Observe that by definition $a(i_{j-1}) \geq a(i_j - 1)$.

Recall that the danger value of v_g directly before B_g is at least $n - c - b(2c-1)$.

Assume first that $k > r$.

$$\begin{aligned}
\overline{\mathsf{dang}}_{B_1}(I_{g-1}) &= \overline{\mathsf{dang}}_{B_g}(I_0) + \sum_{i=1}^{g-1} \left(\overline{\mathsf{dang}}_{B_{g-i}}(I_i) - \overline{\mathsf{dang}}_{B_{g-i+1}}(I_{i-1}) \right) \\
&\geq \overline{\mathsf{dang}}_{B_g}(I_0) \\
&\quad + \sum_{j=1}^{r-1} \left(\overline{\mathsf{dang}}_{B_{g-i_j}}(I_{i_j}) - \overline{\mathsf{dang}}_{B_{g-i_j+1}}(I_{i_j-1}) \right) \\
&\quad \text{[by Corollary 2.14}(i)\text{]} \\
&\geq \overline{\mathsf{dang}}_{B_g}(I_0) - \sum_{j=1}^{r-1} \frac{b + j + a(i_j - 1) - a(i_j)}{j+1} \\
&\quad \text{[by Corollary 2.14}(iii)\text{]}
\end{aligned}$$

Hence,

$$\begin{aligned}
\overline{\mathsf{dang}}_{B_1}(I_{g-1}) &\geq \overline{\mathsf{dang}}_{B_g}(I_0) - bH_r - r - \frac{a(0)}{2} \\
&\quad + \sum_{j=2}^{r-1} \frac{a(i_{j-1})}{(j+1)j} + \frac{a(i_{r-1})}{r} \quad [\text{since } a(i_{j-1}) \geq a(i_j - 1)] \\
&\geq \overline{\mathsf{dang}}_{B_g}(I_0) - bH_k - k \quad [\text{since } a(0) = 0 \text{ and } r \leq k] \\
&\geq n - c - b(2c + \ln k) - k \\
&\geq n - \frac{n}{\ln n}(2c + \ln n - \ln\ln n) - \frac{n}{\ln n} - c \\
&\geq \frac{n \ln\ln n}{3 \ln n} - \frac{n}{\ln n} - c \\
&> 0 \quad [\text{for large } n]. \qquad (2.3)
\end{aligned}$$

Assume now that $k \leq r$.

$$\begin{aligned}
\overline{\mathsf{dang}}_{B_1}(I_{g-1}) &= \overline{\mathsf{dang}}_{B_g}(I_0) + \sum_{i=1}^{g-1} \left(\overline{\mathsf{dang}}_{B_{g-i}}(I_i) - \overline{\mathsf{dang}}_{B_{g-i+1}}(I_{i-1}) \right) \\
&\geq \overline{\mathsf{dang}}_{B_g}(I_0) \\
&\quad + \sum_{j=1}^{r-1} \left(\overline{\mathsf{dang}}_{B_{g-i_j}}(I_{i_j}) - \overline{\mathsf{dang}}_{B_{g-i_j+1}}(I_{i_j-1}) \right) \\
&\quad \text{[by Corollary 2.14}(i)\text{]}
\end{aligned}$$

Thus,

$$\begin{aligned}
\overline{\mathsf{dang}}_{B_1}(I_{g-1}) &\geq \overline{\mathsf{dang}}_{B_g}(I_0) \\
&\quad + \sum_{j=1}^{k-1} \left(\overline{\mathsf{dang}}_{B_{g-i_j}}(I_{i_j}) - \overline{\mathsf{dang}}_{B_{g-i_j+1}}(I_{i_j-1}) \right) \\
&\quad + \sum_{j=k}^{r-1} \left(\overline{\mathsf{dang}}_{B_{g-i_j}}(I_{i_j}) - \overline{\mathsf{dang}}_{B_{g-i_j+1}}(I_{i_j-1}) \right),
\end{aligned}$$

and therefore,

$$\begin{aligned}
\overline{\mathsf{dang}}_{B_1}(I_{g-1}) &\geq \overline{\mathsf{dang}}_{B_g}(I_0) - \sum_{j=1}^{k-1} \frac{b + j + a(i_j - 1) - a(i_j)}{j+1} \\
&\quad - \sum_{j=k}^{r-1} \frac{2b}{j+1} \text{[by Corollary 2.14}(iii) \text{ and } (ii)\text{]} \\
&\geq \overline{\mathsf{dang}}_{B_g}(I_0) - b(2H_r - H_k) - k - \frac{a(0)}{2} \\
&\quad + \sum_{j=2}^{k-1} \frac{a(i_{j-1})}{(j+1)j} + \frac{a(i_{k-1})}{k} \\
&\geq n - c - b(2c - 1 + 2H_n - H_k) - k \\
&\quad \text{[since } n \geq r \text{ and } a(0) = 0\text{]} \\
&\geq n - c \\
&\quad - \left(\frac{n}{\ln n} - \frac{n \ln \ln n}{\ln^2 n} - \frac{(2c+3)n}{\ln^2 n} \right) (\ln n + \ln \ln n + 2c + 2) \\
&\quad - \frac{n}{\ln n}
\end{aligned}$$

$$\begin{aligned} &\geq \frac{n(\ln\ln n)^2}{\ln^2 n} \quad \text{[for } n \text{ large enough]} \\ &> 0. \end{aligned} \tag{2.4}$$

□

Proof of Theorem 2.3: The previous proof works line by line, we only have to adapt the last few lines of the calculations of (2.3) and (2.4). For (2.3), we have

$$\begin{aligned} \overline{\mathsf{dang}}_{B_1}(I_{g-1}) &\geq n - c - b(2c + \ln k) - k \\ &\geq n - c - (1-\epsilon)\cdot\frac{n}{\ln n}\cdot\left(\frac{2\epsilon}{3(1-\epsilon)}\cdot\ln n + \ln n - \ln\ln n\right) \\ &\quad -\frac{n}{\ln n} \\ &\geq \frac{\epsilon}{3}n \\ &> 0. \end{aligned}$$

For (2.4), we obtain

$$\begin{aligned} \overline{\mathsf{dang}}_{B_1}(I_{g-1}) &\geq n - c - b(2c + 2\ln n - \ln k + 1) - k \\ &\geq n - c \\ &\quad -(1-\epsilon)\cdot\frac{n}{\ln n}\cdot\left(\frac{2\epsilon}{3(1-\epsilon)}\cdot\ln n + \ln n + \ln\ln n + 2\right) \\ &\quad -\frac{n}{\ln n} \\ &\geq \frac{\epsilon}{4}n \quad \text{[for large } n\text{]} \\ &> 0. \end{aligned}$$

□

2.4 Building a Connected Graph with High Minimum Degree

Proof of Theorem 2.4: To establish a suitable strategy for Maker we basically merge his strategies for occupying a spanning tree and achieving a graph of min-degree c. We will adopt most of the terminology used in the two corresponding proofs, but sometimes with a slightly modified content. We assume that Breaker starts the game and denote by $\deg_M(v)$

and $\deg_B(v)$ the degree of a vertex v in Maker's graph and in Breaker's graph, respectively. We adopt the concept of active vertices as well, at the beginning each vertex is *active*. After each of his moves Maker deletes one or two vertices from the set of active vertices. The corresponding vertices are called *deactivated*.

By a *component*, we always refer to a connected component of Maker's graph. A component is called *dangerous* if it contains at most $2bc$ vertices. In contrast to Section 2.2 we will have active vertices in dangerous components only. We call a vertex v *dangerous* if it is active or has degree at most $c-1$ in Maker's graph.

We define a *danger function* on the vertex set. Let

$$\mathsf{dang}(v) = \begin{cases} \deg_B(v) & \text{if } v \text{ is active} \\ \deg_B(v) - 2b \cdot \deg_M(v) & \text{otherwise} \end{cases}$$

Maker's Strategy S_M. If there are no dangerous vertices left then Maker occupies an arbitrary free edge connecting two components. Otherwise, for his ith move Maker identifies a dangerous vertex v_i with the largest danger value (ties are broken arbitrarily) and *eases* v_i by doing the following. If v_i is active then Maker claims an arbitrary edge connecting $C(v_i)$ to another component C' and deactivates v_i. In case C' also had an active vertex and $|C(v_i)| + |C'| > 2bc$, then Maker deactivates the active vertex of C' as well. If v_i is not active then Maker claims an arbitrary edge e incident to v_i. In case a new component C emerges upon the selection of e, Maker deactivates some of the (at most two) active vertices of C arbitrarily such that C has one or zero active vertex depending on whether C is dangerous or not, respectively.

Note that Maker can always make a move according to his strategy unless his graph is connected and has minimum degree at least c (thus he won) or Breaker occupied either a cut or an $(n-c)$-star (and Breaker won).

The following is an immediate consequence of the strategy S_M.

Observation 2.15. *Every dangerous component contains exactly one active vertex whereas other components do not have active vertices.* □

Since v_i is only defined for moves when there are still dangerous vertices, we have the following.

Observation 2.16. *Vertex v_i is dangerous any time before Maker's ith move.* □

Proof of Maker's Win Suppose, for a contradiction, that Breaker, playing with a bias b, has a strategy S_B to win the game in question against Maker who plays with bias 1. Let B_i and M_i denote the ith move of Breaker and Maker, respectively, in the game where they play against each other using their respective strategies S_B and S_M. Let g be the length of this game, i.e., g is the smallest integer such that in move B_g Breaker finished occupying either all edges in a cut $(K, V\backslash K)$ or all edges of an $(n-c)$-star. We call this the end of the game.

Proposition 2.17. $g < (c+1) \cdot n$

Proof: Each move of Maker is used either to decrease the number of components or to ease a vertex (occasionally both). Since the number of components can be decreased at most $n-1$ times and each vertex can be eased at most c times (thereafter it stops being dangerous), Maker can make at most $n-1+cn$ moves. □

Analogously to Section 2.3, for a set $I \subseteq V$ of vertices we let $\overline{\mathsf{dang}}(I)$ denote the average danger value $\frac{\sum_{v\in I}\mathsf{dang}(v)}{|I|}$ of the vertices of I. When there is risk of confusion we again add an index and write $\mathsf{dang}_{B_i}(v)$ or $\mathsf{dang}_{M_i}(v)$ to emphasize that we mean the danger-value of v *directly before* B_i or M_i, respectively.

Observation 2.18. *Before Breaker's last move B_g there is a dangerous vertex v_g with* $\mathsf{dang}(v_g) \geq n - b \cdot (2c+1)$

This can be seen by distinguishing two cases.

Case 1. After B_g Breaker has completely occupied an $(n-c)$-star.
In order to be able to do that, directly before B_g there must be a vertex v_g with $\deg_M(v_g) < c$ and $\deg_B(v_g) \geq n-c-b$. Thus $\mathsf{dang}_{B_g}(v_g) \geq n-c-b-2b(c-1) > n-b\cdot(2c+1)$.

Case 2. After B_g Breaker has completely occupied a cut $(K, V\backslash K)$ with $|K| \leq |K\backslash V|$.
In order to be able to do that, directly before B_g all vertices of K must have degree at least $n-|K|-b$ in Maker's graph. Observe that $|K| \leq 2bc$ since otherwise Breaker would had to occupy at least $2bc(n-2bc) > gb$ (by Proposition 2.17) edges in g rounds, a contradiction for large n. Hence by Observation 2.15 K contains an active vertex v_g, whose danger value is at least $\deg_B(v_g) \geq n-|K|-b \geq n-2bc-b = n-b(2c+1)$. □

Since v_g is dangerous before Breaker's last move it is dangerous throughout the whole game. So before each move of Maker there is at least one dangerous vertex, implying that in each move Maker eases a vertex and $v_1, \ldots, v_{g-1}$ are all defined during the game. For $0 \leq i \leq g-1$, let $I_i = \{v_{g-i}, \ldots, v_g\}$.

For an estimate of the change in the average danger during Maker's move the statement of Lemma 2.12 is valid; we still copy it here since its proof has to be slightly adapted.

Lemma 2.19. *Let i, $1 \leq i \leq g-1$,*

(i) if $I_i \neq I_{i-1}$, then $\overline{\mathsf{dang}}_{M_{g-i}}(I_i) - \overline{\mathsf{dang}}_{B_{g-i+1}}(I_{i-1}) \geq 0$.

(ii) if $I_i = I_{i-1}$, then $\overline{\mathsf{dang}}_{M_{g-i}}(I_i) - \overline{\mathsf{dang}}_{B_{g-i+1}}(I_{i-1}) \geq \frac{2b}{|I_i|}$.

Proof: Part (i) can be shown by following the proof of part (i) of Lemma 2.12 word by word. For part (ii) we have that $v_{g-i} \in I_{i-1} = I_i$. We distinguish two cases.

Case 1. v_{g-i} was active before M_{g-i}.
Due to Maker's strategy v_{g-i} is deactivated after M_{g-i}, implying that during M_{g-i} $\mathsf{dang}(v_{g-i})$ decreases by $2b \cdot \deg_M(v_{g-i})$ (where $\deg_M(v_{g-i})$ refers to the moment directly after M_{g-i}). Since after M_{g-i} v_{g-i} is not isolated in Maker's graph (otherwise it would still be active) it has positive degree in Maker's graph, implying that $\mathsf{dang}(v_{g-i})$ decreased by at least $2b$. No vertices increased their danger value during Maker's move, hence $\overline{\mathsf{dang}}(I_i)$ decreases by at least $\frac{2b}{|I_i|}$, which implies *(ii)*.

Case 2. v_{g-i} was already deactivated before M_{g-i}.
In this case we can proceed along similar lines as in the proof of part (ii) of Lemma 2.12. □

The rest of the proof agrees with the one of Theorem 2.2 mutatis mutandis, the only difference being in the calculation that $\mathsf{dang}(v_g)$ is lower bounded by $n - b(2c+1)$ instead of $n - c - b(2c-1)$. □

The proof of Theorem 2.5 follows similarly to Theorem 2.3.

3
On the Clique-Game

3.1 Introduction

In this chapter we study three variations of the clique game.

The Biased Game

Recall that $f_N(m,b)$ denotes the largest q such that Maker can build a K_q in the $(m:b)$ game on K_N. Beck [21] showed that

$$\begin{aligned} f_N(1,1) &= \lfloor 2\log N - 2\log\log N + 2\log e - 3 + o(1)\rfloor \\ &= (2+o(1))\log N. \end{aligned} \tag{3.1}$$

Furthermore, Beck [21] defined a function $g_N(m,b)$ which for constant m,b evaluates to $\left(\frac{2}{\log(m+b)-\log m}+o(1)\right)\log N$. The definition of $g_N(m,b)$ is motivated by the so called Biased Meta-Conjecture by Beck: an adaptation of the random graph intuition which also considers some particular criteria guaranteeing a Maker's win. While $f_N(1,1)=g_N(1,1)$ (by (3.1)) and $f_N(m,1)\geq g_N(m,1)$ (by a result in [21]) for every m and every large enough N, we will show in Section 3.3 that for infinitely many m,b the values $f_N(m,b)$ and $g_N(m,b)$ are substantially different.

Theorem 3.1. *Let m, b be constants. In the $(m:b)$ game Maker has a strategy to occupy a K_q with $q = \left(\frac{m}{\log(b+1)} - o(1)\right) \cdot \log N$.*

In particular, for constant $m \geq 6$ and large enough N,

$$f_N(m,m) \geq \left(\frac{m}{\log(m+1)} - o(1)\right) \log N > g_N(m,m) = (2+o(1)) \log N. \tag{3.2}$$

This connects to the following open problem by Beck.

Open Problem 3.2. *(Open Problem 31.1, [21])*

(a) Is it true that in the $(2:2)$ game Maker has a strategy to occupy a K_q for $q = 2\log N - 2\log\log N + O(1)$?

(b) Is it true that in the $(2:2)$ game Breaker has a strategy to prevent Maker from occupying a K_q for $q = 2\log N - 2\log\log N + O(1)$?

Open Problem 3.2 is still unsolved but (3.2) points out that in the $(6:6)$ game Maker has a strategy to occupy a K_q with $q = 2.13 \log N$. So, if in Open Problem 3.2 "$(2:2)$" was replaced with "$(6:6)$", then the answer to (a) would be "yes" whereas the answer to (b) would be "no". Hence, it is plausible that the answers are similar in Open Problem 3.2 as well.

Building a Clique Fast

In Section 3.4 we investigate the minimum number of moves $s(q)$ Maker needs to build a K_q in the game on an arbitrarily large K_N. (3.1) implies the following.

Corollary 3.3. *Maker can build a K_q on K_N with $N = (1+o(1))q2^{\frac{q}{2}}$.*

Hence $s(q)$ is upper bounded by $\frac{1}{2}\binom{N}{2} \leq q^2 2^q$. According to the best known bounds, $s(q)$ is sandwiched between $2^{\frac{q}{2}}$ (due to Beck [20]) and 2^{q+2} (a result obtained independently by Beck [20] and Pekeč [65]). In our second theorem we improve the upper bound on $s(q)$.

Theorem 3.4. *Maker has a strategy to build a K_q in $2^{\frac{2q}{3}}\mathsf{poly}(q)$ moves.*

Building a Tournament

Finally, we derive a strategy for Maker for occupying a large tournament, which supports the random graph intuition.

Theorem 3.5. *If $q \leq (1-o(1)) \log N$, then Maker has a winning strategy in the q-tournament game.*

We will prove Theorem 3.5 in Section 3.5.

Notation

Suppose that we consider a game played on the edge set of a graph G. Then, for every vertex $v \in V(G)$ we let $d_B(v)$ denote the degree of v in Breaker's graph.

3.2 Proof Sketches for Theorems 3.1, 3.4 and 3.5

We first sketch a very natural strategy for Maker in the ordinary $(1:1)$ game, which allows him to build a K_q with $q = (1 - o(1)) \log N$ in $2N$ moves. For the clique size this bound is weaker than (3.1) by a factor of 2; however, appropriate adaptations of our strategy form new Maker's strategies for some variations of the clique game, allowing us to prove Theorem 3.1, Theorem 3.4 and Theorem 3.5.

Maker's Strategy We consider the following strategy S for Maker. Maker first selects an arbitrary vertex v_1 of the vertex set of K_N. As his ith move Maker claims a free edge (v_1, w_i) until all edges incident to v_1 are occupied. We refer to this sequence of moves as *processing* v_1.

Note that by applying S Maker achieves that at least $r := \frac{N-1}{2}$ vertices $w_1, w_2, \ldots, w_r$ are connected to v_1 in his graph. So he can restrict himself to building a $(q-1)$-clique in the subgraph induced by $\{w_1, \ldots, w_r\}$, which has roughly $\frac{N}{2}$ vertices. This suggests that by applying S recursively Maker can, for $q \approx \log N$, build a q-clique. We note that, actually, there is an obstacle which has to be taken into account: It is possible that before Maker finishes processing v_1, Breaker already claimed some edges in the subgraph induced by $\{w_1, w_2, \ldots, w_r\}$, which might be a handicap for Maker. However, this can be resolved by ignoring all vertices whose degree in Breaker's graph is larger than some carefully chosen threshold t, and setting up a more involved recurrence. (A detailed description will be given in the sequel.)

We now sketch three modifications of S which can be applied to some variations of the clique game.

The Biased Game For the biased clique game, we consider the following modification of S: At the beginning, instead of selecting *one* vertex v_1, Maker occupies an m-clique on some vertex set $\{v_1, v_2 \ldots, v_m\}$. (In the more detailed analysis in Section 3.3 we will show how this can be achieved.) As long as there are vertices v for which $(v, v_1), \ldots, (v, v_m)$ are all unclaimed, as his move, Maker fixes such a v and connects v to $v_1, \ldots, v_m$. In this way Maker can achieve that in his graph roughly $\frac{N}{b+1}$ vertices are adjacent to *every* $v_i \in \{v_1, \ldots, v_m\}$. So he can restrict himself to occupying a $(q-m)$-clique on the graph G' induced by the set of those vertices w which are adjacent to every v_i with $1 \leq i \leq m$ in his graph.

A handwaving analysis (neglecting again the fact that Breaker might have claimed edges of G') gives that Maker can build a K_q if $N \geq (b+1)^{\frac{q}{m}}$, i.e., if $q \leq m \log_{b+1}(N) = \frac{m}{\log(b+1)} \log N$. This is close to the bound we will prove in Section 3.3.

Building a Clique Fast Let $N \approx q2^{\frac{2q}{3}}$. Maker proceeds in two phases. In the first phase he applies S: This allows him to occupy roughly $\frac{N}{2}$ edges of the form $(v_1, w_1), (v_1, w_2), \ldots, (v_1, w_{\frac{N}{2}})$. Again we will not take into account that Breaker might have claimed edges in the subgraph induced by $\{w_1, \ldots, w_{\frac{N}{2}}\}$. Thus Maker can proceed recursively on the subgraph induced by $\{w_1, \ldots, w_{\frac{N}{2}}\}$.

By applying S exactly $\frac{q}{3}$ times he can obtain vertices $v_1, \ldots, v_{\frac{q}{3}}$ and $w_1, \ldots, w_{\frac{N}{2^{q/3}}} = w_{q2^{q/3}}$ such that *every* w_j is connected with *every* v_i in his graph. This will take him roughly $\sum_{i=1}^{q/3} \frac{N}{2^i} \leq N = q2^{\frac{2q}{3}}$ moves.

By Corollary 3.3, Maker can occupy a $K_{\frac{2q}{3}}$ on the subgraph induced by $\{w_1, \ldots, w_{q2^{q/3}}\}$ (provided q is large enough); clearly, this takes him at most $(q2^{\frac{q}{3}})^2 \leq q^2 2^{\frac{2q}{3}}$ moves. This forms the second phase. The two phases together allow Maker to achieve a K_q in roughly $2q^2 2^{\frac{2q}{3}}$ moves.

The Tournament Game Finally, for the tournament game Maker can adapt his strategy S as follows. Let T be the goal-tournament of Maker on the vertex set $\{u_1, \ldots, u_q\}$. During the game Maker will maintain so called *candidate sets* $V_1, \ldots, V_q$ such that every $v_i \in V_i$ is still suitable for the part of vertex u_i. In the first round Maker selects a vertex $v_1 \in V_1$ and then responds to each Breaker's move as follows: If Breaker claims an edge connecting v_1 with a vertex in V_i with $i \geq 2$, then Maker claims another, free edge e connecting v_1 with V_i (if there is no such edge e,

Maker just claims an arbitrary edge). Otherwise, he claims any free edge $e = (v_1, v_i)$ with $v_i \in V_i$ for some $i \geq 2$. In either case Maker orients e in such a way that v_1 is the sink of e if and only if u_1 is the sink of (u_1, u_i).

In this way Maker can restrict himself to occupying a copy of $T \backslash \{u_1\}$ in the subgraph induced by $W_2 \cup W_3 \cup \ldots \cup W_q$ where $W_j \subseteq V_j$ is the set of vertices in V_j which are in Maker's graph adjacent to v_1. By Maker's strategy, the size of W_j is at least $\frac{|V_j|}{2}$. This suggests that Maker can succeed if $\frac{N}{q} \geq 2^q$ (at the beginning every candidate set V_j has size $\frac{N}{q}$, and in every round at least half of the vertices of V_j remain candidates), i.e., if $q = (1 - o(1)) \log N$. (Again we did not take into account that some of the edges in the subgraph induced by $W_2 \cup W_3 \ldots W_q$ might already have been claimed by Breaker in the first round.)

3.3 The Biased Game

The next statement is a well known fact in graph theory.

Observation 3.6. *Let G be a graph on n vertices where every vertex has degree at most d. Then G contains an independent set of size at least $\frac{n}{d+1}$.*

This can be seen by considering the following greedy algorithm for building an independent set: Start with an empty set S and then, as long as G contains at least one vertex, iteratively select an arbitrary vertex v, add it to S and remove v and all its neighbors from G. Finally, at most $(d+1)|S|$ vertices were deleted and therefore, $|S| \geq \frac{n}{d+1}$.

In this chapter we will use the following observation several times.

Observation 3.7. *Let $d \geq 0, d_{\mathsf{crit}} \geq 1$ be integers, let G be a graph, and let $S \subseteq V(G)$ such that $d_B(v) \leq d$ for every $v \in S$. Suppose that Breaker claims e additional edges which have at least one endpoint in S. Then there are at least $|S| - \frac{2e}{d_{\mathsf{crit}}}$ vertices $w \in S$ where $d_B(w) \leq d + d_{\mathsf{crit}}$.*

This can be seen as follows. In Breaker's graph, the sum of vertex-degrees in S is increased by at most $2e$. Let $W \subseteq S$ denote the set of vertices $w \in S$ where $d_B(w)$ is increased by more than d_{crit}. If $|W| > \frac{2e}{d_{\mathsf{crit}}}$, then the sum of vertex-degrees in S (in Breaker's graph) is increased by at least $|W| d_{\mathsf{crit}} > 2e$, which leads to a contradiction. Hence, $|W| \leq \frac{2e}{d_{\mathsf{crit}}}$. We have $d_B(v) \leq d + d_{\mathsf{crit}}$ for every $v \in S \backslash W$, as claimed

Before proving Theorem 3.1 we formulate some more auxiliary facts. Observation 3.6 directly implies the next corollary.

Corollary 3.8. *Let G be a graph on n vertices where $d_B(v) \leq d$ for every vertex $v \in V(G)$. Then there is an $S \subseteq V(G)$ with $|S| \geq \frac{n}{d+1}$ such that S is an independent set in Breaker's graph.*

The next proposition shows that Maker can build any clique on a complete graph of sufficiently large size.

Proposition 3.9. *For every integers q, m, b there is an $n = n(q, m, b)$ such that for every $n' \geq n$ Maker has a strategy to build a K_q in the $(m : b)$ game played on $K_{n'}$.*

Proof: It suffices to consider the case where $m = 1$ (Maker can only benefit from larger values). We proceed by induction on q. Clearly, Maker can always build a K_1. Suppose now that $q > 1$ and let $\tilde{n} = n(q-1, 1, b)$.

Let V denote the vertex set of a K_n (n is to be determined later). Maker uses the following strategy. He first selects an arbitrary vertex $v \in V$. In each of his moves he claims an edge incident to v until all such edges are occupied. In this way, at least $\frac{n-1}{b+1}$ vertices are connected to v in Maker's graph. In the meantime Breaker claimed at most $b(n-1)$ edges. Maker ignores all vertices in $V \backslash \{v\}$ with degree at least $4b(b+1)$ in Breaker's graph. Hence he ignores at most $\frac{2b(n-1)}{4b(b+1)} = \frac{n-1}{2(b+1)}$ vertices and therefore has a set W of $\frac{n-1}{b+1} - \frac{n-1}{2(b+1)} = \frac{n-1}{2(b+1)}$ vertices where $d_B(w) < 4b(b+1)$ for every $w \in W$, and all edges in $E(\{v\}, W)$ belong to his graph. By Corollary 3.8, there is a subset $W' \subseteq W$ with $|W'| \geq \frac{|W|}{4b(b+1)}$ such that none of the edges in $E(W')$ belongs to Breaker's graph. Note that $|W'| \geq \frac{\frac{n-1}{2(b+1)}}{4b(b+1)}$. By choosing n appropriately we obtain that $|W'| \geq \tilde{n}$, and thus by induction, Maker can build a K_{q-1} on W', which together with v forms a K_q. □

Proof of Theorem 3.1: Choose $C = C(m, b)$ in such a way that Maker has a strategy to build a K_m in the $(m : b)$ game played on K_C. (Proposition 3.9 guarantees that such a C exists.) Note that since we consider b and m as constants, C is also a constant. Throughout this section we abbreviate $(m : b)$ *game* by *game*.

The next lemma shows how Maker can reduce the task of occupying a K_q to the task of occupying a K_{q-m}. To this end we consider complete graphs where some of the edges are already occupied by either Maker or Breaker.

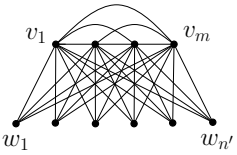


Figure 3.1: The drawn edges represent edges in Maker's graph.

Lemma 3.10. *Let G be a complete graph on n vertices where $d_B(v) \leq d$ for every $v \in V(G)$ and $n \geq C(d+1)$. Let*

$$n' = \frac{n - m - md - 2b\binom{C}{2}}{b+1} \quad . \tag{3.3}$$

If $n' \geq 1$, then Maker can achieve that there are disjoint sets $\{v_1, \dots, v_m\}$, $\{w_1, \dots, w_{n'}\} \subseteq V(G)$ where
(i) all edges in $E(\{v_1, \dots, v_m\}) \cup E(\{v_1, \dots, v_m\}, \{w_1, \dots, w_{n'}\})$ belong to Maker's graph, and
(ii) $d_B(w_i) \leq d + 2(b+1)$ for every i, $1 \leq i \leq n'$.

Figure 3.1 shows an illustration of Lemma 3.10.(i) for $m = 4$ and $n' = 6$. Before proving Lemma 3.10 we first show its consequences. Note that if Maker manages to occupy a $(q-m)$-clique induced by some vertex set $W \subseteq \{w_1, \dots, w_{n'}\}$, then he possesses the q-clique induced by $W \cup \{v_1, \dots, v_m\}$. We will use this fact to analyze Maker's strategy recursively. For integers n, d we let $\mathcal{G}(n,d)$ denote the set of complete graphs G on n vertices where $d_B(v) \leq d$ for every vertex $v \in V(G)$. Lemma 3.10 implies the following.

Corollary 3.11. *Let d, n, q be integers with $n \geq C(d+1)$ and $q \geq m$, let $G \in \mathcal{G}(n,d)$, and let $n' \geq 1$ be defined as in Lemma 3.10. Maker can build a K_q in the game on G if for every $G' \in \mathcal{G}(n', d+2(b+1))$ he can build a K_{q-m} in the game on G'.*

The following corollary is a direct consequence of Corollary 3.8 and the definition of C.

Corollary 3.12. *Let $G \in \mathcal{G}(n,d)$. If $n \geq C(d+1)$, then Maker can build a K_m in the game on G.*

Let $c = m + 2b\binom{C}{2}$ and let $i \geq 2$. By applying Corollary 3.11 $(i-1)$ times and using Corollary 3.12, we obtain that in the original game on

K_N, Maker can occupy a K_{im} if

$$\cfrac{\cfrac{\cfrac{N-(c+0\cdot m)}{b+1}-(c+2(b+1)m)}{b+1}-(c+4(b+1)m)}{\begin{array}{c}\vdots\\ \cfrac{}{b+1}\end{array}}-(c+2(i-2)(b+1)m)\Bigg\}\ b+1$$

is at least $C(d'+1)$ where $d'=2(i-1)(b+1)$.

Hence, Maker can build a K_{im} if

$$\frac{N}{(b+1)^{i-1}}-(c+2(i-2)(b+1)m)\sum_{j=1}^{i-1}\frac{1}{(b+1)^j}\geq C(2(i-1)(b+1)+1). \tag{3.4}$$

Since C, m, b are constants, (3.4) is equivalent to $\frac{N}{(b+1)^{i-1}}\geq c'+c''(i-2)$, for appropriately chosen positive constants c', c''. Let $k=c'+c''$. Hence we have the following.

Corollary 3.13. *If $N\geq ki(b+1)^{i-1}$ for some $i\geq 2$, then Maker can occupy a K_{im} in the game on K_N.*

Let

$$q=\left(\frac{m}{\log(b+1)}-\frac{1}{\log\log N}\right)\cdot\log N.$$

We apply Corollary 3.13 for $i=\frac{q}{m}$ (for simplicity, we assume that q is divisible by m). Using that $q\leq m\log N$, we get

$$\begin{aligned} k\frac{q}{m}(b+1)^{\frac{q}{m}-1} &\leq k\log N(b+1)^{\left(\frac{1}{\log(b+1)}-\frac{1}{m\log\log N}\right)\log N}\\ &= kN\log N(b+1)^{-\frac{\log N}{m\log\log N}}\\ &= kN(b+1)^{\frac{\log\log N}{\log(b+1)}-\frac{\log N}{m\log\log N}}\\ &\quad (\text{since } \log N=(b+1)^{\frac{\log\log N}{\log(b+1)}})\\ &< N \quad (\text{provided } N \text{ is large enough}). \end{aligned}$$

Hence, Maker can occupy a K_q in the game on K_N, which concludes the proof of Theorem 3.1. □

It remains to show Lemma 3.10.

Proof of Lemma 3.10: We assume that no edge of G belongs to Maker's graph. Otherwise, Maker can follow his strategy and, whenever this strategy calls for an edge he already possesses, then he takes an arbitrary free edge: no extra move is disadvantageous for him.

Maker proceeds in two phases.

Phase 1 By assumption, $n \geq C(d+1)$ and thus by Corollary 3.8, there is an $S \subseteq V(G)$ with $|S| = C$ such that none of the edges in $E(S)$ belongs to Breaker's graph. Maker first builds a K_m on some vertex set $\{v_1, \ldots, v_m\} \subseteq S$ (this is possible by the choice of C). Note that in the meantime Breaker occupied at most $b\binom{C}{2}$ edges. From now on Maker will ignore all vertices $v \in V(G)\backslash\{v_1, \ldots, v_m\}$ incident to one of these edges. Moreover, Maker will also ignore those vertices $v \in V(G)$ which are connected to some v_i, $i \in \{1, \ldots, m\}$ in Breaker's graph. In total Maker ignores at most $2b\binom{C}{2} + md$ vertices. Let W denote the set of remaining vertices in $V(G)\backslash\{v_1, \ldots, v_m\}$ and note that

$$|W| \geq n - m - 2b\binom{C}{2} - md. \tag{3.5}$$

Every Breaker's edge incident to a vertex in W has been occupied before Phase 1. So, $d_B(w) \leq d$ for every $w \in W$. Moreover, no edge in $E(\{v_1, \ldots, v_m\}, W)$ belongs to Breaker's graph. Using a similar argument as in the beginning of the proof we can assume, without loss of generality, that no edge in $E(\{v_1, \ldots, v_m\}, W)$ is assigned to Maker.

Phase 2 As long as there are vertices $w \in W$ where (w, v_i) is unclaimed for every $i \in \{1, \ldots, m\}$, as his move Maker selects such a w and occupies the edges $(w, v_1), (w, v_2), \ldots, (w, v_m)$. We refer to such a move as "saving w".

Let r denote the number of vertices Maker saved and, additionally, let $W' = \{w_1, \ldots, w_r\}$ denote the corresponding vertex set. Clearly, $r \geq \frac{|W|}{b+1}$. Moreover, after Phase 2 every $w \in W\backslash W'$ is connected to some v_i with $i \in \{1, \ldots, m\}$ in Breaker's graph, for otherwise, Maker would have saved w. In total, Breaker claimed rb edges during Phase 2, at least $|W| - r$ of which are in $E(W\backslash W', \{v_1, \ldots, v_m\})$. Hence, Breaker claimed at most $rb - (|W| - r) = r(b+1) - |W|$ edges with an endpoint in W'. Figure 3.2 shows an illustration of the two phases for $m = 4$ and $r = 6$.

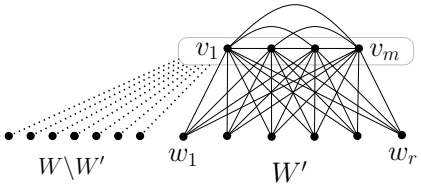


Figure 3.2: The solid edges represent Maker's edges and the dotted edges indicate that every $w \in W \backslash W'$ is adjacent to some v_i in Breaker's graph.

Let W'' denote the set of those vertices $w \in W'$ where $d_B(w) \leq d + 2(b+1)$. By Observation 3.7, $|W''| \geq |W'| - \left(\frac{2(r(b+1)-|W|)}{2(b+1)}\right) = r - \left(r - \frac{|W|}{b+1}\right) = \frac{|W|}{b+1}$. So, $\{v_1, \ldots, v_m\}$, W'' form the required sets. □

3.4 Building a Clique Fast

Proof of Theorem 3.4: Throughout this section we consider the $(1:1)$ game. We will describe a strategy which allows Maker to occupy a K_q fast. Maker proceeds in two phases. The next lemma describes the first phase.

Lemma 3.14. *Let $i, r \geq 1$ be integers and let $N = 9ir2^i$. Then in the game on K_N, Maker can achieve in at most $2N$ moves that for some disjoint $V, W \subseteq V(G)$ with $|V| = i$ and $|W| = r$,*

(i) all edges in $E(V) \cup E(V, W)$ belong to Maker's graph, and

(ii) the subgraph induced by W does not contain any edge of Breaker's graph.

Figure 3.3 shows an illustration of Lemma 3.14 for $i = 4$ and $r = 6$. We postpone the proof of Lemma 3.14 and continue with the proof of Theorem 3.4. For the second phase we apply Corollary 3.3 for the subgraph induced by W. Let $r = 2q'2^{\frac{q'}{2}}$ for some large enough q', and let V, W be the vertex sets from Lemma 3.14. By Corollary 3.3, Maker can build a $K_{q'}$ on the subgraph induced by W, which takes him at most $\binom{r}{2} \leq 2q'^2 2^{q'}$ moves. Hence, altogether Maker can build a $K_{i+q'}$ in

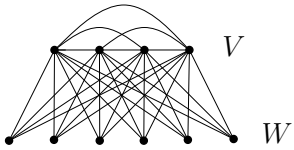


Figure 3.3: The drawn edges belong to Maker's graph. None of the edges spanned by W belongs to Breaker's graph.

$2N + 2q'^2 2^{q'} = 36iq'2^{\frac{q'}{2}+i} + 2q'^2 2^{q'}$ moves. By setting $i = \frac{q}{3}$ and $q' = \frac{2q}{3}$, we obtain Theorem 3.4. □

It remains to show Lemma 3.14. The next lemma describes a main ingredient of Maker's strategy. As in the proof of Theorem 3.1 we consider complete graphs where some of the edges are already occupied by either Maker or Breaker.

Lemma 3.15. *Let G be a complete graph on n vertices where $d_B(w) \leq d$ for every $w \in V(G)$, let $v \in V(G)$, and let*

$$n' = \frac{n - (d+1)}{2}. \tag{3.6}$$

If $n' \geq 1$, then Maker can achieve in at most n moves that for some $\{w_1, \ldots, w_{n'}\} \subseteq V(G)$, (v, w_i) belongs to Maker's graph and $d_B(w_i) \leq d + 4$ for every i, $1 \leq i \leq n'$.

Proof: As in the proof of Lemma 3.10 we can assume, without loss of generality, that no edge of G belongs to Maker's graph. By assumption, $d_B(v) \leq d$, and hence there is a $W \subseteq V(G)\backslash\{v\}$ with $|W| \geq n - (d+1)$ such that none of the edges in $E(\{v\}, W)$ belongs to Breaker's graph. Maker proceeds as follows. Until all edges in $E(\{v\}, W)$ are occupied, as his ith move he claims a free edge $(v, w_i) \in E(\{v\}, W)$. Suppose that he can make r such moves (note that $r \leq n$).

In the meantime Breaker occupied r edges, including (v, w) for every $w \in W\backslash\{w_1, \ldots, w_r\}$, since otherwise, Maker would have claimed (v, w). So, Breaker occupied at most $r - (|W| - r) = 2r - |W|$ edges incident to a vertex in $\{w_1, \ldots, w_r\}$.

By Observation 3.7, there is a $W' \subseteq \{w_1, \ldots, w_r\}$ with $|W'| \geq r - \frac{2(2r-|W|)}{4} = \frac{|W|}{2} \geq \frac{n-(d+1)}{2}$ where $d_B(w') \leq d + 4$ for every $w' \in W'$. So, W' forms the required vertex set. □

Proof of Lemma 3.14: Maker proceeds in i steps. It will turn out that every step j yields appropriate, disjoint vertex sets $V_j, W_j \subseteq V(G)$ where all edges in $E(V_j) \cup E(V_j, W_j)$ belong to Maker's graph.

Let $V_0 := \emptyset$ and $W_0 := V(G)$. (So, obviously, all edges in $E(V_0) \cup E(V_0, W_0) = \emptyset$ belong to Maker's graph.) Let $j \geq 1$ and suppose that Maker has finished step $j-1$. Thus by induction, he occupied all edges in $E(V_{j-1}) \cup E(V_{j-1}, W_{j-1})$. For every $w \in W_{j-1}$, let $d_B^{(j-1)}(w)$ denote the number of Breaker's edges incident to w in the subgraph induced by W_{j-1} (note that if $j = 0$, then $d_B^{(j-1)}(w) = d_B(w)$ for every $w \in V(G)$). Moreover, let d_{max} be the maximum of $d_B^{(j-1)}(w)$ over all vertices $w \in W_{j-1}$ and let v be an arbitrary vertex of W_{j-1}. Applying Lemma 3.15 for the subgraph induced by W_{j-1} gives that Maker can achieve in at most $|W_{j-1}|$ moves that for some $W \subseteq W_{j-1}$ with $|W| = \frac{|W_{j-1}|-(d_{\mathsf{max}}+1)}{2}$, the edge (v, w) belongs to Maker's graph and $d_B^{(j-1)}(w) \leq d_{\mathsf{max}} + 4$ for every $w \in W$. Let $V_j := V_{j-1} \cup \{v\}$ and $W_j := W$. This completes step j. Note that by construction, all edges in $E(V_j) \cup E(V_j, W_j)$ belong to Maker's graph and $d_B^{(j)}(w) \leq d_B^{(j-1)}(w) \leq d_{\mathsf{max}} + 4$, for every $w \in W_j$.

By repeatedly applying Lemma 3.15, we obtain that $|V_i| = i$,

$$|W_i| := \cfrac{\cfrac{\cfrac{\cfrac{N-(0+1)}{2} - (4+1)}{2} - (8+1)}{\vdots}}{\cfrac{}{2}} - (4(i-1)+1)}{2},$$

and that $d_B^{(i)}(w) \leq 4i$ for every $w \in W_i$. Note that V_i and W_i are disjoint and that $|W_i| \geq \frac{N}{2^i} - 4i(\frac{1}{2} + \frac{1}{4} + \ldots + \frac{1}{2^i}) \geq \frac{N}{2^i} - 4i \geq 5ir$ (the last inequality is due to our choice of N). Finally, Maker needed at most $N + \frac{N}{2} + \ldots + \frac{N}{2^{i-1}} \leq 2N$ moves to occupy the edges in $E(V_i) \cup E(V_i, W_i)$.

By Corollary 3.8, there is a $W \subseteq W_i$ with $|W| \geq \frac{|W_i|}{4i+1}$ such that none of the edges in $E(W)$ belongs to Breaker's graph. We have $|W| \geq \frac{5ir}{4i+1} \geq r$, and thus V_i, W form the required sets. □

3.5 Building a Tournament

We will consider a modification of the tournament game which is advantageous for Breaker. The *advanced q-clique game* is played on K_N.

At the beginning Breaker fixes a partition $V_1, V_2, \ldots, V_q$ of the vertex set with $|V_1| = |V_2| = \ldots = |V_q| = \frac{N}{q}$. (For simplicity we assume here that N is divisible by q. The case where N is not divisible by q could be handled by fixing q disjoint vertex sets $V_1, V_2, \ldots, V_q$ with $|V_i| = \lfloor \frac{N}{q} \rfloor$ for every i, $1 \leq i \leq q$.) Maker's goal is to build a K_q on some vertex set $\{v_1, v_2, \ldots, v_q\}$ with $v_i \in V_i$ for every i, $1 \leq i \leq q$; and Breaker's goal is to prevent this.

Observation 3.16. *Let N be an integer and let T be any tournament on q vertices. If Maker has a strategy to win the advanced q-clique game on K_N, then he has a strategy to occupy a copy of T on K_N.*

This can be seen as follows. Let $\{u_1, \ldots, u_q\}$ denote the vertex set of T. Suppose that Maker has a strategy S to win the advanced q-clique game on K_N. To occupy a copy of T, he first fixes an arbitrary partition $V_1, V_2, \ldots, V_q$ of the vertex set of K_N with $|V_i| = \frac{N}{q}$ for every $i \in \{1, \ldots, q\}$, and then, in each of his moves, he applies S together with the following rule for directing the currently claimed edge e: If $e = (v_i, v_j)$ for some $v_i \in V_i$, $v_j \in V_j$ with $i \neq j$, then he orients e in such a way that e is directed from v_i to v_j if and only if the edge between u_i and u_j is directed from u_i to u_j. Otherwise, e is spanned by some set V_i, in which case Maker chooses an arbitrary orientation.

Since S is a winning strategy for the advanced q-clique game, Maker manages to occupy a copy of T in this way.

Proof of Theorem 3.5: By Observation 3.16, it suffices to show that Maker has a strategy to win the advanced q-clique game, abbreviated by q-clique game in the following, on K_N. Let d_{crit} be an integer to be determined later. The next lemma shows how a winning strategy for the $(q-1)$-clique game yields a winning strategy for the q-clique game. As in the proof of Theorem 3.1 we let $\mathcal{G}(n, d)$ denote the set of complete graphs G on n vertices where $d_B(v) \leq d$ for every $v \in V(G)$.

Lemma 3.17. *Let d, n, q be positive integers, let $G \in \mathcal{G}(qn, d)$, and let $V_1, V_2, \ldots, V_q$ be a partition of $V(G)$ with $|V_i| = n$ for every i, $1 \leq i \leq q$. Moreover, let $v_1 \in V_1$ and let*

$$n' = \frac{n-d}{2} - \frac{2nq}{d_{\mathsf{crit}}} \quad .$$

If $n' \geq 1$, then Maker can achieve that for some $W_2 \subseteq V_2, W_3 \subseteq V_3, \ldots, W_q \subseteq V_q$,

(i) $|W_i| = n'$, for every i, $2 \leq i \leq q$,

(ii) *all edges in* $E(\{v_1\}, W_i)$ *belong to Maker's graph, for every* i, $2 \leq i \leq q$, *and*

(iii) $d_B(w) \leq d + d_{\mathsf{crit}}$, *for every* $w \in W_2 \cup W_3 \cup \ldots \cup W_q$.

Before proving Lemma 3.17 we first consider its consequences.

Corollary 3.18. *Let* d, n, q *be positive integers, let* $G \in \mathcal{G}(qn, d)$ *and let* $n' \geq 1$ *be defined as in Lemma 3.17. Maker has a strategy to win the* q*-clique game on* G *if for every* $G' \in \mathcal{G}((q-1)n', d + d_{\mathsf{crit}})$ *he has a strategy to win the* $(q-1)$*-clique game on* G'.

Suppose that $d_{\mathsf{crit}} > 4q$ and let n' be defined as in Lemma 3.17. Then,

$$n' = n\frac{d_{\mathsf{crit}} - 4q}{2d_{\mathsf{crit}}} - \frac{d}{2} = \frac{n}{\frac{2d_{\mathsf{crit}}}{d_{\mathsf{crit}} - 4q}} - \frac{d}{2} = \frac{n}{2 + \frac{8q}{d_{\mathsf{crit}} - 4q}} - \frac{d}{2} \quad . \tag{3.7}$$

Let $s(q) = 2 + \frac{8q}{d_{\mathsf{crit}} - 4q}$. By repeatedly applying Corollary 3.18 and (3.7), we obtain that Maker has a winning strategy in the q-clique game on K_{qn} if $\tilde{n} \geq 1$ where

$$\tilde{n} := \cfrac{\cfrac{\cfrac{\cfrac{n}{s(q)} - 0/2}{s(q-1)} - d_{\mathsf{crit}}/2}{s(q-2)} - 2d_{\mathsf{crit}}/2}{\begin{array}{c}\vdots\\ s(2)\end{array}} - (q-2)d_{\mathsf{crit}}/2 \ .$$

Recall that we assumed that $d_{\mathsf{crit}} > 4q$. Hence, $s(q) \geq 2$ and for every $1 \leq i \leq j \leq q$, $s(i) = \frac{2d_{\mathsf{crit}}}{d_{\mathsf{crit}} - 4i} \leq \frac{2d_{\mathsf{crit}}}{d_{\mathsf{crit}} - 4j} = s(j)$. So, $\tilde{n} \geq \frac{n}{s(q)^{q-1}} - (q-2)\frac{d_{\mathsf{crit}}}{2}\left(1 + \frac{1}{s(2)} + \frac{1}{s(2)^2} + \ldots + \frac{1}{s(2)^{q-2}}\right) \geq \frac{n}{s(q)^q} - qd_{\mathsf{crit}}$.

Corollary 3.19. *Let* N, q *and* $d_{\mathsf{crit}} > 4q$ *be integers and let* $n = \frac{N}{q}$. *If* $\frac{n}{s(q)^q} - qd_{\mathsf{crit}} \geq 1$ *with* $s(q) = 2 + \frac{8q}{d_{\mathsf{crit}} - 4q}$, *then Maker has a winning strategy in the* q*-clique game on* K_N.

We apply Corollary 3.19 for $d_{\mathsf{crit}} = 5q^2$ and $q = (1 - 5\frac{\log\log N}{\log N})\log N$. For large enough N we have $s(q) = 2 + \frac{8q}{5q^2 - 4q} \leq 2 + \frac{2}{q}$. Hence, $s(q)^q \leq (2 + \frac{2}{q})^q = 2^q(1 + \frac{1}{q})^q \leq 2^q e$. Using that $q \leq \log N$ and $2^q = \frac{N}{2^{5\log\log N}}$ we get

$$\frac{n}{s(q)^q} - qd_{\mathsf{crit}} \geq \frac{N}{q2^q e} - 5q^3 \geq \frac{2^{5\log\log N}}{e\log N} - 5\log^3 N \geq \frac{\log^4 N}{e} - 5\log^3 N \geq 1,$$

for large enough N. This concludes our proof of Theorem 3.5 □

Proof of Lemma 3.17: As in the proof of Lemma 3.10 we can assume, without loss of generality, that no edge of G belongs to Maker's graph. Since $d_B(v_1) \leq d$, for every $i \in \{2, \dots, q\}$ there are at most d vertices $v \in V_i$ where (v_1, v) belongs to Breaker's graph. As long as there are unclaimed edges incident to v_1 Maker responds to each Breaker's move as follows. If Breaker claims an edge of the form (v_1, v_i) with $v_i \in V_i$ for some $i \geq 2$ and there are still unclaimed edges in $E(\{v_1\}, V_i)$, then Maker occupies an arbitrary free edge in $E(\{v_1\}, V_i)$. Otherwise, Maker claims any free edge incident to v_1. He stops as soon as all edges incident to v_1 have been claimed. Hence, Maker and Breaker each made at most qn moves. Note that due to his strategy, for every $i \in \{2, \dots, q\}$ Maker occupied at least half of those edges in $E(\{v_1\}, V_i)$ which were initially unclaimed; hence, there is a $V_i' \subseteq V_i$ with $|V_i'| = \frac{n-d}{2}$ such that all edges in $E(\{v_1\}, V_i')$ belong to Maker's graph.

Since Breaker made at most qn moves he claimed at most qn edges; thus, by Observation 3.7, for every $i \in \{2, \dots, q\}$ there is a $W_i \subseteq V_i'$ with $|W_i| \geq |V_i'| - \frac{2qn}{d_{\mathsf{crit}}} = \frac{n-d}{2} - \frac{2qn}{d_{\mathsf{crit}}}$ such that $d_B(w) \leq d + d_{\mathsf{crit}}$ for every $w \in W_i$. □

4

Size Ramsey Number of Bounded Degree Graphs for Games

4.1 Introduction

In this chapter we show that Maker can build a copy of any given graph of bounded maximum degree on a sparse board.

Theorem 4.1. *Let d be a natural number. Then there is a constant $c = c(d)$ with the property that for every graph G on n vertices of maximum degree d there is a graph H on at most cn edges such that Maker has a strategy to occupy a copy of G in the game on H.*

Let $v_1, \ldots, v_n$ denote the vertices of G and let $E(G)$ denote the edge set of G. The graph H we will construct in the proof of Theorem 4.1 has the additional property that for some carefully chosen constant c (depending on d but not on n), we will have $H = G^c$ where G^c denotes the graph obtained by replacing every v_i with a set V_i of size c, and connecting two vertices $u \in V_i$ and $v \in V_j$ with an edge if and only if $(v_i, v_j) \in E(G)$. Figure 4.1 depicts such a graph. Note that H is a

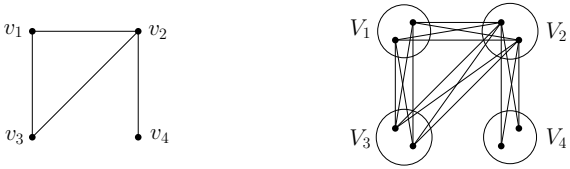


Figure 4.1: A graph G (on the left) and G^2 (on the right).

graph on cn vertices of maximum degree cd, and thus has at most $\frac{1}{2}c^2dn$ edges. Moreover, the strategy we will describe guarantees that by the end of the game Maker's graph contains a copy of G spanned by some $w_1 \in V_1, \ldots, w_n \in V_n$ where every w_i plays the role of v_i.

Previous Work Feldheim and Krivelevich [40] considered a related problem: For a given graph G they investigated the minimum number of moves Maker needs to build a copy of G in the game on a sufficiently large K_N. They derived a strategy for Maker to quickly occupy every given graph of bounded degree. An important aspect of this strategy is that Maker's task is reduced to winning a series of carefully designed discrepancy games. By a result of Alon, Krivelevich, Spencer and Szabó [7], which guarantees that Maker can win each of these subgames, the strategy of Feldheim and Krivelevich gives the following.

Theorem 4.2. *Let d be an integer. Then there are constants $c = c(d), c' = c'(d)$ such that for every graph G on n vertices of maximum degree d and every $N > cn$, Maker has a strategy to occupy a copy of G in the game on K_N in at most $c'n$ rounds.*

We remark that they actually proved this result for the more general class of d-degenerate graphs. The values of the constants are $c = d^{11}2^{2d+9}$ and $c' = d^{11}2^{2d+7}$. Our construction for proving Theorem 4.1 builds on quite a few of their findings and approaches, and couples them with new ingredients.

Universal Graphs For a given family $\mathcal{G}$ of graphs, a graph H is called $\mathcal{G}$-universal if H contains a copy of every $G \in \mathcal{G}$. The construction of sparse universal graphs for several families of graphs occurs in the study of VLSI circuit design (c.f.[26] and [33]). Universal graphs for forests, planar graphs, and related classes have been studied in a series of papers

(see, e.g., [25, 30, 31, 32, 45], and [10, 25, 28, 29, 67]). Alon, Capalbo, Kohayakawa, Rödl, Ruciński and Szemerédi [5] considered the class $\mathcal{G}_{n,d}$ of graphs on n vertices with maximum degree d. They proved that for every d there exists an $\epsilon = \epsilon(d) > 0$ such that for every n there is a $\mathcal{G}_{n,d}$-universal graph with at most $n^{2-\epsilon}$ edges. Further progress was made in a series of publications, yielding several deterministic and randomized constructions of sparse $\mathcal{G}_{n,d}$-universal graphs (see, e.g., [2, 3, 6]). Finally, Alon and Capalbo [4] found the following.

Theorem 4.3. *For every $d \geq 3$ there exists positive constants c_1, c_2 such that for every n there is an (explicitly constructible) $\mathcal{G}_{n,d}$-universal graph H with at most $c_1 n$ vertices and at most $c_2 n^{2-\frac{2}{d}}$ edges.*

Theorem 4.3 is tight since it has been shown in [5] that every $\mathcal{G}_{n,d}$-universal graph contains at least $\Omega(n^{2-\frac{2}{d}})$ edges.

Ramsey Universal Graphs and a Game-Theoretic Variant For a given family $\mathcal{G}$ of graphs, a graph H is called $\mathcal{G}$-Ramsey-universal if any two-coloring of the edges of H contains a monochromatic $\mathcal{G}$-universal graph. Clearly, the lower bound $\Omega(n^{2-\frac{2}{d}})$ (of [5]) on the number of edges in a $\mathcal{G}_{n,d}$-universal graph also serves as a lower bound on the number of edges in a $\mathcal{G}_{n,d}$-Ramsey-universal graph. From the other side, Kohayakawa, Rödl, Schacht and Szemerédi [56] showed that there is a $\mathcal{G}_{n,d}$-Ramsey-universal graph with at most

$$O(n^{2-\frac{1}{d}} \log^{\frac{1}{d}} n) \tag{4.1}$$

edges, which is a generalization of (1.5) on page 14. It is a wide open problem whether this upper bound can be pushed down to $O(n^{2-\frac{2}{d}})$.

As for the ordinary Ramsey property, there is a game-theoretic variant of Ramsey-universality: For a given family $\mathcal{G}$ of graphs we are interested in those graphs H where for *every* $G \in \mathcal{G}$, Maker has a strategy to build a copy of G in the game on H. (In other words, Maker first fixes H and afterwards Breaker chooses G, and then the actual game starts.) We denote the set of these graphs H by $S(\mathcal{G})$ and investigate the smallest number $s = s(\mathcal{G})$ such that there exists a graph H in $S(\mathcal{G})$ with s edges.

Every graph in $S(\mathcal{G})$ is clearly $\mathcal{G}$-universal. Thus, the lower bound $\Omega(n^{2-\frac{2}{d}})$ (of [5]) on the number of edges in a $\mathcal{G}_{n,d}$-universal graph yields that every graph in $S(\mathcal{G}_{n,d})$ has at least $\Omega(n^{2-\frac{2}{d}})$ edges. Hence,

$$s(\mathcal{G}_{n,d}) \geq \Omega(n^{2-\frac{2}{d}}). \tag{4.2}$$

From the other side, a standard strategy-stealing argument yields that if a graph H is G-Ramsey (i.e., if every two-coloring of the edges of H contains a monochromatic copy of G) then Maker can build a copy of G in the game on H. Thus, every $\mathcal{G}_{n,d}$-Ramsey-universal graph is also in $S(\mathcal{G}_{n,d})$. Hence (4.1) implies that some graph in $S(\mathcal{G}_{n,d})$ has at most $O(n^{2-\frac{1}{d}}\log^{\frac{1}{d}} n)$ edges. Together with (4.2) this gives that

$$\Omega(n^{2-\frac{2}{d}}) \leq s(\mathcal{G}_{n,d}) \leq O(n^{2-\frac{1}{d}}\log^{\frac{1}{d}} n).$$

Our construction for proving Theorem 4.1 closes the gap between these two bounds: Let H be the graph from Theorem 4.3 and let $H' = H^c$ for some carefully chosen constant c. By construction, $|E(H')| = c^2|E(H)| = O(n^{2-\frac{2}{d}})$, and, furthermore, H' contains G^c for every $G \in \mathcal{G}_{n,d}$. Thus, by choosing c as in the proof of Theorem 4.1, for every $G \in \mathcal{G}_{n,d}$, Maker has a strategy to occupy a copy of G. Hence, $H' \in S(\mathcal{G}_{n,d})$ and therefore, $s(\mathcal{G}_{n,d}) \leq O(n^{2-\frac{2}{d}})$, which together with (4.2) gives that $s(\mathcal{G}_{n,d}) = \Theta(n^{2-\frac{2}{d}})$.

Notation We first define some game-theoretic notions. Following the standard notation, for a graph property $\mathcal{P}$ of N-vertex graphs and a graph H on the vertex set $V(H) = V(K_N)$ we let $(E(H), \mathcal{P})$ denote the game where Maker's goal is to create a graph which possesses $\mathcal{P}$. In this chapter we investigate the case where $\mathcal{P}$ is the property that the graph contains a copy of a fixed graph G. We call H the *base graph* or the *board*. The base graph along with the sets of Maker's and Breaker's claimed edges is called a *game position*, or just a *position* for short. Adopting the notation of [40], to distinguish between vertices of G and vertices of H we mark the vertices of H with an asterisk.

Throughout this chapter we will assume that Breaker starts the game. Otherwise Maker can start with an arbitrary move, then follow his strategy. If his strategy calls for an edge he already claimed he takes an arbitrary edge (he can only benefit from extra moves). By slightly modifying the standard notation, we let a *round* denote a pair consisting of a Breaker's move and the consecutive Maker's move.

We will also need some graph-terminology. Let G be a graph and let $u, v \in V(G)$. The *neighborhood* $N_G(v)$ of v denotes the set of vertices which are adjacent to v in G. The *distance* $\mathsf{dist}_G(u, v)$ *between* u *and* v is defined as the number of edges in a shortest path in G connecting u and v. For a fixed ordering $v_1, \ldots, v_n$ of the vertices of G we let

$N_G^-(v_i) = N_G(v_i) \cap \{v_1, \ldots, v_{i-1}\}$, and $N_G^+(v_i) = N_G(v_i) \cap \{v_{i+1}, \ldots, v_n\}$. When there is no danger of confusion we sometimes omit the index G.

Let D be a directed acyclic graph and let $u, v \in V(D)$. If $(u, v) \in E(D)$ then v is called an *out-neighbor* of u. If there is a directed path of length at least one from u to v in D then v is called a *descendant* of u. Note that according to this convention, no vertex is considered a descendant of itself.

Organization of this Chapter In Section 4.2 we highlight some features of the approach of Feldheim and Krivelevich, and formulate the basic idea of our proof. In Section 4.3 we devise an appropriate ordering Π of the vertices of G and point out some properties of Π which will be essential for Maker's strategy. In Sections 4.4 and 4.5 we give (slightly modified versions of) some key-concepts and results of Feldheim and Krivelevich, and apply them to our problem. Finally, in Section 4.6 we derive a winning strategy for Maker, using the ingredients presented in the preceding sections together with some new ideas.

4.2 Sketching a Known Result and Our Contribution

Let G be a graph of maximum degree d and suppose that $V(G) = \{v_1, \ldots, v_n\}$. We first point out some important properties of Maker's strategy S (for constructing G fast in the game on K_N) presented by Feldheim and Krivelevich. Maker processes the vertices $v_1, \ldots, v_n$ one by one. For each v_i he chooses a set $W_i \subseteq V(K_N)$ of "fresh vertices" (i.e., vertices where no incident edge has been claimed by Maker or Breaker) where $|W_i|$ is some fixed constant c (depending on d but not on n). Then he starts claiming edges connecting W_i with the already constructed sets $W_1, \ldots, W_{i-1}$, according to an involved strategy $S(i)$, until Maker's subgraph spanned by $E(W_i, W_1 \cup \ldots \cup W_{i-1})$ has some carefully specified properties. Then Maker chooses a set $W_{i+1} \subseteq V(K_N)$ of fresh vertices and continues in this way. Feldheim and Krivelevich derive that the strategies $S(i)$ guarantee that by the end of the game Maker can find vertices $w_1^\star \in W_1$, $w_2^\star \in W_2, \ldots, w_n^\star \in W_n$ such that $\{w_1^\star, \ldots, w_n^\star\}$ span a copy of G, where every $w_i^\star$ plays the role of v_i.

We aim to modify this strategy in such a way that it allows Maker to occupy a copy of G on a sparse board. Recall that G^c denotes the graph obtained by replacing every v_i with a set V_i of size c, and connecting two

vertices $u \in V_i$ and $v \in V_j$ with an edge if and only if $(v_i, v_j) \in E(G)$. The basic idea is to set $H := G^c$, and $W_i := V_i$ for every $i \in \{1, \ldots, n\}$. So, in contrast to the above strategy, the W_i are already determined at the beginning of the game. At first glance it may seem that the only thing Maker has to do is to respond to Breaker claiming an edge in $E(W_i, W_j)$ where $i < j$, by applying a move according to $S(j)$ – the property of the $S(j)$ pointed out above then guarantees that Maker's graph finally contains a copy of G. However, the obstacle is that the $S(j)$ can not be applied in "parallel", since $S(j)$ depends on the outcome of processing $v_1, \ldots, v_{j-1}$.

We observe that in order to apply $S(j)$, Maker, actually, only has to process a certain subset of $\{v_1, \ldots, v_{j-1}\}$. More precisely, for every vertex v_j we specify a subset $P(v_j) \subseteq \{v_1, \ldots, v_{j-1}\}$ with the property that, after processing all vertices in $P(v_j)$, Maker can apply $S(j)$.

The basic idea of our new strategy for Maker is that, whenever Breaker claims an edge connecting W_i and W_j (suppose that $i < j$), Maker first checks whether the vertices in $P(v_j)$ have already been processed. If this is the case he makes a move according to $S(j)$, otherwise he processes some vertex in $P(v_j)$. We will show that for a carefully chosen ordering $v_1, \ldots, v_n$ of the vertices, the sets $P(v_j)$ are all very small, which yields that Breaker can not claim too many edges connecting W_i and W_j before Maker is ready to apply $S(j)$. Thus, Maker's "delay" in each board $E(W_j, W_1 \cup \ldots \cup W_{j-1})$ is small enough so that he can basically still apply $S(j)$, which finally enables him to occupy a copy of G.

4.3 Defining a Suitable Ordering of the Vertices

Let G be a graph. Using the standard notation, $\Delta(G)$ denotes the maximum degree of G and the *chromatic number* $\chi(G)$ denotes the minimum c such that G is c-colorable. We will apply the following well-known bound.

Lemma 4.4. *For every graph G, $\chi(G) \leq \Delta(G) + 1$.*

This can be seen as follows: Let $G = (V, E)$, let $V = \{v_1, \ldots, v_n\}$ and let $C = \{1, \ldots, \Delta(G) + 1\}$ be a set of colors. By coloring the vertices in the order $v_1, \ldots, v_n$, assigning to v_i the smallest color of C not already used on a neighbor of v_i, we obtain a proper coloring. □

Figure 4.2: A graph G (on the left) and $D_2(G)$ (on the right).

For a given graph $G = (V, E)$ we let $D_2(G) = (V', E')$ denote the graph where $V' = V$ and E' consists of all pairs (u, v) of vertices where $u \neq v$ and u, v have distance at most two in G. Figure 4.2 shows an example. Note that, with $\Delta := \Delta(G)$, the maximum degree of $D_2(G)$ is at most $\Delta + \Delta(\Delta - 1) = \Delta^2$. Moreover, every independent set of $D_2(G)$ corresponds to a subset $S \subseteq V$ where every two vertices in S have distance at least three in G. The next corollary is a direct consequence of Lemma 4.4.

Corollary 4.5. *Let $G = (V, E)$ be a graph and let $\Delta = \Delta(G)$. Then $\chi(D_2(G)) \leq \Delta^2+1$. In particular, we can color V with colors $\{1, \ldots, \Delta^2+1\}$ such that every two vertices of the same color have distance at least 3 in G.*

Coloring the Vertices From now on we let G be a fixed graph on n vertices of maximum degree d and we let $l : V(G) \to \{1, \ldots, d^2 + 1\}$ be the coloring of Corollary 4.5. Figure 4.4(a) shows an example. We also fix an ordering $v_1, \ldots, v_n$ of the vertices in $V(G)$ such that $l(v_1) \leq l(v_2) \leq \ldots \leq l(v_n)$. Finally, we assume without loss of generality that $d \geq 2$. (Note that if the maximum degree is at most 1 then we can just consider a supergraph $G' \supseteq G$ with maximum degree 2.)

Let $1 \leq i < j \leq n$. We say that an index r is *neighborwise sandwiched between* i and j if $i < r < j$ and $v_r \in N_G(v_i)$. For every $(v_i, v_j) \in E(G)$ where $i < j$ we let $S_{i,j}$ denote the set of indices r which are neighborwise sandwiched between i and j. Figure 4.3 shows an illustration. Moreover,

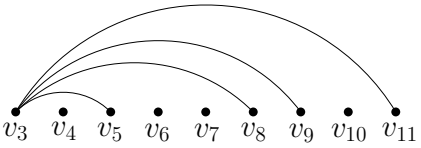


Figure 4.3: An example for $i = 3$ and $j = 11$. We have $S_{3,11} = \{5, 8, 9\}$.

for every index j we let $\mathcal{S}_j$ denote the union of all sets $S_i := \{i\} \cup S_{i,j}$ where $v_i \in N_G^-(v_j)$. Note that all vertices in $\mathcal{S}_j$ have distance at most two from v_j, thus

$$|\mathcal{S}_j| \leq d + d(d-1) = d^2. \tag{4.3}$$

It will turn out that for every j the set of indices $\mathcal{S}_j$ plays an important role in Maker's strategy. Intuitively, it will be the case that he can only process a vertex v_j as soon as he "completed" all vertices v_i where $i \in \mathcal{S}_j$ (more details will be given in the sequel).

For further discussion we aim to express the relation "$i \in \mathcal{S}_j$" by a graph: Let D denote the directed graph on the vertex set $v_1, \ldots, v_n$ where there is an arc from v_j to v_i if and only if $i \in \mathcal{S}_j$. Figures 4.4(b) and 4.4(c) depict an example. For every arc $(v_k, v_i) \in E(D)$ we have $l(v_k) \geq l(v_i)$ and $l(v_k) \neq l(v_i)$ (because $\mathsf{dist}(v_i, v_k) \leq 2$), thus $l(v_k) > l(v_i)$, and therefore D is acyclic.

Observation 4.6. *For every $j \in \{1, \ldots, n\}$ we have that the number of descendants of v_j in D is at most $(d^2)^{d^2+1}$.*

This can be seen as follows. By (4.3) and the construction of D, every vertex has at most d^2 out-neighbors in D. Since $l(v_k) > l(v_i)$ for every arc (v_k, v_i) in D, the vertices of every directed path in D have distinct colors. Thus, every directed path in D with start vertex v_j has at most $l(v_j) \leq d^2 + 1$ vertices. So the number of descendants of v_j is at most $\sum_{i=1}^{l(v_j)-1} (d^2)^i \leq (d^2)^{l(v_j)} \leq (d^2)^{d^2+1}$. □

In terms of the intuition formulated above, Observation 4.6 yields that in order to be able to process a vertex v_j, Maker only has to complete a constant number of vertices. A precise analysis will be given in Section 4.6 when we devise Maker's overall strategy.

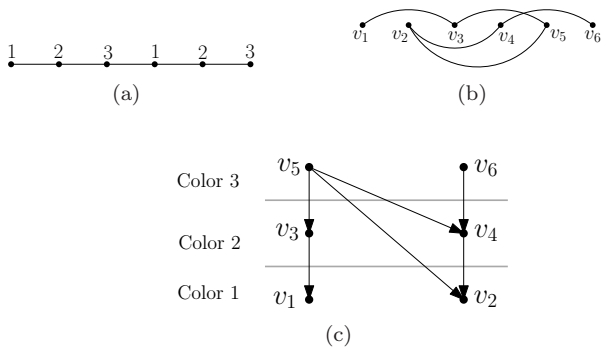


Figure 4.4: A coloring l of a path of length 5 such that vertices with the same color have distance at least 3 (on the top left), and an ordering of the vertices according to l (on the top right). The figure on the bottom shows the corresponding graph D.

Construction of the Board Recall that we fixed an ordering of the vertices $v_1, \ldots, v_n$ of G according to the coloring l of Corollary 4.5. Let

$$c_d = d^5 2^{d+4}, \quad \text{and} \tag{4.4}$$

$$c = dc_d^2 d^{2d^2+2} + c_d + 2. \tag{4.5}$$

We set

$$H := G^c. \tag{4.6}$$

Recall that, by definition of G^c, H is the graph resulting from replacing every v_i with a set V_i of size c, and connecting two vertices $u \in V_i$ and $v \in V_j$ with an edge if and only if $(v_i, v_j) \in E(G)$.

4.4 Candidates and Candidate Schemes

We introduce the concepts of a *candidate vertex* and a *candidate scheme* by stating adapted versions of Definitions 2.1–2.3 in [40]. Recall that, according to (4.6), $V(H) = V_1 \cup \ldots \cup V_n$, and that a *position* is the board graph H along with the sets of Maker's and Breaker's claimed edges. The intuition behind the $B_k \subseteq V_k$ used in the next definitions is that during the game, as a part of his strategy, Maker will define for every $k \in \{1, \ldots, n\}$ an appropriate subset $B_k \subseteq V_k$ with $|B_k| = c_d$.

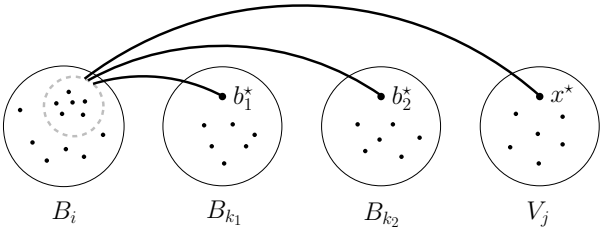


Figure 4.5: An illustration of Definition 4.7 for $t = 3$: Many vertices in B_i are connected to all three vertices $b_1^\star, b_2^\star, x^\star$.

Definition 4.7. *(Vertex candidate with respect to a specific edge) Let $H^\star$ be a position, let $(v_i, v_j) \in E(G)$ where $i < j$, and suppose that $S_{i,j} = \{k_1, \dots, k_{t-1}\}$. Moreover, for every $k \in \{i\} \cup \{k_1, \dots, k_{t-1}\}$, let $B_k \subseteq V_k$ be a non-empty set. A vertex $x^\star \in V_j$ is called a* candidate with respect to the edge (v_i, v_j) and the family $\mathcal{B} = \{B_i, B_{k_1}, \dots, B_{k_{t-1}}\}$ *if for every choice of vertices $b_1^\star \in B_{k_1}, b_2^\star \in B_{k_2}, \dots, b_{t-1}^\star \in B_{k_{t-1}}$ we have*

$$\frac{|\{b^\star \in B_i : \text{Maker claimed } (b^\star, b_1^\star), \dots, (b^\star, b_{t-1}^\star), (b^\star, x^\star) \text{ in } H^\star\}|}{|B_i|} \geq \frac{1}{t2^t} \ .$$

Figure 4.5 illustrates Definition 4.7. Note that if no index is neighborwise sandwiched between i and j (i.e., if $t = 1$) then a vertex $x^\star \in V_j$ is a candidate with respect to (v_i, v_j) and $\mathcal{B} = \{B_i\}$ if and only if $x^\star$ is connected to at least half of the vertices of B_i in Maker's graph.

We will now specify those vertices $x^\star \in V_j$ which are candidates for every edge $(v_i, v_j) \in E(G)$ with $i < j$.

Definition 4.8. *(Vertex candidate) Let $H^\star$ be a position, let $1 \leq j \leq n$ and for every $k \in \mathcal{S}_j$ let $B_k \subseteq V_k$. Moreover, let $\mathcal{B} = \{B_k : k \in \mathcal{S}_j\}$. A vertex $x^\star \in V_j$ is called a* candidate with respect to the family $\mathcal{B}$ *if for every $v_i \in N_G^-(v_j)$, $x^\star$ is a candidate with respect to (v_i, v_j) and $\{B_i\} \cup \{B_k : k \in S_{i,j}\}$.*

Note that if $N_G^-(v_j) = \emptyset$ then every $x^\star \in V_j$ is a candidate with respect to the empty set. Finally, we define a *candidate scheme* which, as we will see, guarantees that Maker's graph contains a copy of G.

Definition 4.9. *(Candidate Scheme) Let $H^\star$ be a position and for every $1 \leq j \leq n$, let $B_j \subseteq V_j$ with $|B_j| \geq d2^d$. We say that $(B_1, B_2, \dots, B_n)$*

form a candidate scheme *if for every* $1 \leq j \leq n$ *and every* $x^\star \in B_j$, $x^\star$ *is a candidate with respect to* $\{B_k : k \in \mathcal{S}_j\}$.

The next lemma, which is a slight adaptation of Lemma 2.1 in [40], shows that a candidate scheme is sufficient for Maker's win.

Lemma 4.10. *(Feldheim, Krivelevich) Let* $H^\star$ *be a position and let* $(B_1, B_2, \ldots, B_n)$ *be a candidate scheme. Then Maker's graph contains a copy of* G.

For completeness we reproduce the proof given in [40].

Proof of Lemma 4.10: Our goal is to construct an embedding $\Phi : \{v_1, \ldots, v_n\} \to V(H)$ such that $\Phi(v_i) \in B_i$ for every $1 \leq i \leq n$, and $\Phi(v_1), \ldots, \Phi(v_n)$ span a copy of G in Maker's graph. We proceed inductively, starting from $\Phi(v_n)$ and moving down to $\Phi(v_1)$. First, we choose an arbitrary $x^\star \in B_n$ and set $\Phi(v_n) := x^\star$. Let $i \leq n-1$ and suppose that we have already defined $\Phi(v_{i+1}), \ldots, \Phi(v_n)$ in such a way that they span a copy of $G[\{v_{i+1}, \ldots, v_n\}]$ in Maker's graph. If $N_G^+(v_i) = \emptyset$ then we choose an arbitrary vertex $x^\star \in B_i$ as $\Phi(v_i)$; clearly, $\Phi(v_i), \ldots, \Phi(v_n)$ span a copy of $G[\{v_i, \ldots, v_n\}]$ in Maker's graph. Otherwise, let $\{v_{k_1}, \ldots, v_{k_t}\} = N_G^+(v_i)$ and note that by assumption, $t \leq d$. Since $\Phi(v_{k_t})$ is a candidate with respect to the edge (v_i, v_{k_t}) and $\{B_i, B_{k_1}, \ldots, B_{k_{t-1}}\}$ we get that

$$|\{b^\star \in B_i : \text{Maker claimed } (b^\star, \Phi(v_{k_1})), \ldots, (b^\star, \Phi(v_{k_t})) \text{ in } H^\star\}| \geq \frac{|B_i|}{t2^t},$$

which is at least $\frac{|B_i|}{d2^d} \geq 1$. We choose one of these vertices as $\Phi(v_i)$. This gives that $\Phi(v_i), \ldots, \Phi(v_n)$ span a copy of $G[\{v_i, \ldots, v_n\}]$ in Maker's graph. □

4.5 Two Suitable Subgames

Our goal is to devise a strategy which allows Maker to obtain a candidate scheme. To this end we first analyze two appropriate subgames. We will need the concept of untouched vertices: Let $H^\star$ be a position. A vertex $x^\star \in V_j$ is *touched* if Maker or Breaker claimed an edge of the form $(y^\star, x^\star)$ where $y^\star \in V_1 \cup V_2 \cup \ldots \cup V_{j-1}$. Otherwise $x^\star$ is called *untouched*.

In the first subgame Maker's goal is to achieve the situation described in Definition 4.7.

Definition 4.11. *Let $H^\star$ be a position, let $(v_i, v_j) \in E(G)$ where $i < j$, and suppose that $S_{i,j} = \{k_1, \dots, k_{t-1}\}$. Moreover, for every $k \in \{i\} \cup \{k_1, \dots, k_{t-1}\}$, let $B_k \subseteq V_k$ be a non-empty set. For every vertex $x^\star \in V_j$ which is untouched in $H^\star$, the game $G_{B_i,x^\star}$* with respect to *$\{B_{k_1}, \dots, B_{k_{t-1}}\}$ is defined as follows. The board consists of the set of edges $E_H(B_i, \{x^\star\})$, and Maker's goal is to achieve that $x^\star$ becomes a candidate with respect to (v_i, v_j) and $\{B_i, B_{k_1}, \dots, B_{k_{t-1}}\}$.*

Note that for some positions $H^\star$, the game is hopeless for Maker; e.g. for every $H^\star$ where for some $1 \le r \le t-1$ and some $y^\star \in B_{k_r}$, all edges in $E_H(B_i, \{y^\star\})$ belong to Breaker's graph. The strategy for Maker, which we will describe, however, guarantees that the B_k are convenient.

Maker's goal in the second subgame is that several $x^\star \in V_j$ become a candidate with respect to *all* edges (v_i, v_j) where $i < j$.

Definition 4.12. *Let $H^\star$ be a position, let $1 \le j \le n$, and for every $k \in \mathcal{S}_j$, let $B_k \subseteq V_k$ be a non-empty set. For every set $B_j \subseteq V_j$ where all vertices of B_j are untouched in $H^\star$, the game G_{B_j}* with respect to *$\{B_k : k \in \mathcal{S}_j\}$ is defined as follows. The board consists of the union of the edge-sets $E_H(B_i, B_j)$ where $v_i \in N_G^-(v_j)$, and Maker's goal is to achieve that* every *$x^\star \in B_j$ becomes a candidate with respect to $\{B_k : k \in \mathcal{S}_j\}$.*

We will show that Maker has a strategy to determine the B_j in such a way that he succeeds in every G_{B_j}, which finally allows him to obtain a candidate scheme.

We first express the board size of the game G_{B_j} in terms of the size of the B_i where $v_i \in N_G^-(v_j)$.

Observation 4.13. *Let $H^\star$ be a position, let $1 \le j \le n$, and for every $k \in \mathcal{S}_j \cup \{j\}$ let $B_k \subseteq V_k$ be as in Definition 4.12. The board size of the game G_{B_j} (with respect to $\{B_k : k \in \mathcal{S}_j\}$) is $\sum_{v_i \in N_G^-(v_j)} |B_i||B_j|$.*

We now use an adaptation of Lemma 2.2 in [40] to show that for convenient $B_{k_1}, \dots, B_{k_{t-1}}$, Maker has a strategy to win the game $G_{B_i,x^\star}$ described in Definition 4.11.

Lemma 4.14. *Let $H^\star$ be a given position, let $(v_i, v_j) \in E(G)$ where $i < j$, and suppose that $S_{i,j} = \{k_1, \dots, k_{t-1}\}$. Moreover, for every $k \in \{i\} \cup \{k_1, \dots, k_{t-1}\}$, let $B_k \subseteq V_k$ where $|B_k| = c_d$. Suppose that for every $1 \le r \le t-1$ we have that every $y^\star \in B_{k_r}$ is a candidate with respect to (v_i, v_{k_r}) and $\{B_i\} \cup \{B_{k_1}, \dots, B_{k_{r-1}}\}$. Then, for every $x^\star \in V_j$ which is untouched in $H^\star$, Maker has a strategy to win $G_{B_i,x^\star}$ with respect to $\{B_{k_1}, \dots, B_{k_{t-1}}\}$.*

Lemma 4.14 can be shown along similar lines as Lemma 2.2 in [40]. For completeness we reproduce the proof here.

We first need some notation. Let $F = (V, E)$ be a hypergraph, i.e., the hyperedge set E is a subset of the power set 2^V. We consider the game where Maker and Breaker alternately claim an unclaimed vertex of V until all vertices are claimed. We will use the following result by Alon, Krivelevich, Spencer and Szabó [7], extending a previous result by Székely [75].

Theorem 4.15. *(Alon, Krivelevich, Spencer, Szabó) Let F be a hypergraph with X hyperedges, whose smallest hyperedge contains at least x vertices. Then Maker has a strategy to claim at least $\frac{x}{2} - \sqrt{\frac{x \ln(2X)}{2}}$ vertices of each hyperedge.*

Proof of Lemma 4.14: If $t = 1$ then Maker can win easily: In each of his moves he claims a free edge connecting $x^\star$ with a vertex in B_i; this guarantees that by the end of the game half of the edges in $E(B_i, \{x^\star\})$ belong to Maker's graph, as claimed. Suppose now that $t \geq 2$ and let F denote the hypergraph where $V(F) = E(B_i, \{x^\star\})$ and $E(F)$ is obtained by adding (to the initially empty set), for every choice of vertices $b_1^\star \in B_{k_1}, b_2^\star \in B_{k_2}, \ldots, b_{t-1}^\star \in B_{k_{t-1}}$, the hyperedge

$$e_{b_1^\star,\ldots,b_{t-1}^\star} := \{(b^\star, x^\star) \in E(B_i, \{x^\star\}) : \text{ Maker claimed } (b^\star, b_1^\star), \ldots, (b^\star, b_{t-1}^\star)\}.$$

By interpreting $G_{B_i,x^\star}$ as a game on F, we get that Lemma 4.14 is equivalent to the statement that Maker has a strategy to claim at least $\frac{|B_i|}{t2^t} = \frac{c_d}{t2^t}$ vertices of each hyperedge. Our goal is to show the latter. By (4.4) we have that

$$|E(F)| \leq \prod_{r=1}^{t-1} |B_{k_r}| = {c_d}^{t-1} = \left(d^5 2^{d+4}\right)^{t-1}.$$

Let $b_1^\star \in B_{k_1}, \ldots, b_{t-1}^\star \in B_{k_{t-1}}$ be any choice of vertices. We get that $|e_{b_1^\star,\ldots,b_{t-1}^\star}| = |\{b^\star \in B_i : \text{ Maker claimed } (b^\star, b_1^\star), \ldots, (b^\star, b_{t-1}^\star)\}|$. Since (by assumption) $b_{t-1}^\star$ is a candidate with respect to $(v_i, v_{k_{t-1}})$ and $\{B_i, B_{k_1}, \ldots, B_{k_{t-2}}\}$, we obtain that

$$|e_{b_1^\star,\ldots,b_{t-1}^\star}| \geq \frac{|B_i|}{(t-1)2^{t-1}} = \frac{c_d}{(t-1)2^{t-1}} = \frac{d^5 2^{d+4}}{(t-1)2^{t-1}} \quad .$$

By Theorem 4.15, Maker has a strategy to claim at least

$$\frac{d^5 2^{d-t+4}}{t-1} - \sqrt{\frac{d^5 2^{d-t+4}}{t-1} \ln(2(d^5 2^{d+4})^{t-1})} \tag{4.7}$$

vertices of each hyperedge. We need some auxiliary calculations. It can easily be seen that for every integer d,

$$d^5 2^4 > 5d^4 \ln d + (d+4)d^4 \ln 2 + d^3 \ln 2.$$

Hence, for every integers $1 < t \leq d$ we have that

$$\begin{aligned} \frac{d^5 2^{d-t+4}}{(t-1)t^2} &> \frac{1}{(t-1)t^2}\left(5t^4 \ln d + (d+4)t^4 \ln 2 + t^3 \ln 2\right) \\ &> 5(t-1)\ln d + (d+4)(t-1)\ln 2 + \ln 2 \\ &= \ln(2(d^5 2^{d+4})^{t-1}). \end{aligned}$$

Multiplying with $\frac{d^5 2^{d-t+4}}{(t-1)}$ and taking the root gives that

$$\frac{d^5 2^{d-t+4}}{(t-1)t} > \sqrt{\frac{d^5 2^{d-t+4}}{t-1} \ln(2(d^5 2^{d+4})^{t-1})},$$

which, together with the fact that $\frac{1}{t} = 1 - \frac{t-1}{t}$, directly implies that the expression in (4.7) is at least $\frac{d^5 2^{d-t+4}}{t} = \frac{c_d}{t 2^t}$, which concludes the proof. □

The next corollary is a consequence of Lemma 4.14.

Corollary 4.16. *Let $H^\star$ be a position, let $1 \leq j \leq n$, and for every $k \in \mathcal{S}_j$, let $B_k \subseteq V_k$ where $|B_k| = c_d$. Suppose that for every $v_i \in N_G^-(v_j)$ and every $q \in S_{i,j}$ we have that every $y^\star \in B_q$ is a candidate with respect to (v_i, v_q) and $\{B_i\} \cup \{B_r : r \in S_{i,q}\}$. Then, for every $B_j \subseteq V_j$ which is untouched in $H^\star$, Maker has a strategy to win G_{B_j} with respect to $\{B_k : k \in \mathcal{S}_j\}$.*

Proof: We first note that for every $v_i \in N_G^-(v_j)$ the conditions of Lemma 4.14 are satisfied. Indeed, let $S_{i,j} = \{k_1, \ldots, k_{t-1}\}$. By assumption, for every $k_r \in \{k_1, \ldots, k_{t-1}\}$ we have that all vertices in B_{k_r} are candidates with respect to (v_i, v_{k_r}) and $\{B_i\} \cup \{B_s : s \in S_{i,k_r}\} = \{B_i\} \cup \{B_{k_1}, \ldots, B_{k_{r-1}}\}$.

Consider the following strategy for Maker. Suppose that Breaker claims an edge $(b^\star, x^\star)$ with $b^\star \in B_i$ and $x^\star \in B_j$ where $i < j$. Then Maker responds in the game $G_{B_i, x^\star}$ (with respect to $\{B_r : r \in S_{i,j}\}$).

Since the boards of the games $G_{B_i,x^\star}$ are pairwise disjoint, Maker can treat each game $G_{B_i,x^\star}$ separately. Thus Lemma 4.14 yields a winning strategy for Maker in G_{B_j}. □

The next observation shows that the game G_{B_j} is finished after a reasonably small number of rounds.

Observation 4.17. *Let $H^\star$ be a position, let $1 \leq j \leq n$, and for every $k \in \mathcal{S}_j \cup \{j\}$ let $B_k \subseteq V_k$ be such that the conditions of Corollary 4.16 are satisfied. Suppose additionally that $|B_j| = c_d$. By Observation 4.13 the board size of the game G_{B_j} (with respect to $\{B_k : k \in \mathcal{S}_j\}$) is $\sum_{v_i \in N_G^-(v_j)} |B_i||B_j| \leq dc_d^2$, hence G_{B_j} lasts at most dc_d^2 rounds.*

Corollary 4.16 Gives a Strategy for Winning Fast We point out that Theorem 4.2 is a consequence of Corollary 4.16. To this end we first note that in the setting of Theorem 4.2, H is a complete graph on N vertices, so we do not define $V_1, \ldots, V_n$. Suppose that Maker proceeds as follows. He processes the vertices $v_1, \ldots, v_n$ one by one, maintaining the invariant that after processing v_{j-1} he has vertex sets $B_1, \ldots, B_{j-1}$ of size c_d such that for every $i \in \{1, \ldots, j-1\}$, every vertex of B_i is a candidate with respect to $\{B_k : k \in \mathcal{S}_i\}$. (Note that for $j = 1$ the invariant is clearly fulfilled.) After processing v_{j-1}, Maker selects a set B_j of c_d untouched vertices (the existence of such a B_j can be guaranteed by choosing a large enough N) and applies Corollary 4.16, which gives him a strategy to transform all vertices of B_j into candidates with respect to $\{B_k : k \in \mathcal{S}_j\}$. Thus, the invariant is still fulfilled after processing v_j. By induction, after processing v_n, $(B_1, \ldots, B_n)$ form a candidate scheme, which by Lemma 4.10 guarantees that Maker's graph contains a copy of G. Due to Observation 4.17, processing $v_1, \ldots, v_n$ takes at most $dc_d^2 n$ rounds, hence Maker needs only a linear number of rounds.

4.6 Obtaining a Candidate Scheme via the Subgames

Equipped with Corollary 4.16 we can now describe a strategy for Maker to obtain a candidate scheme (which due to Lemma 4.10 guarantees that Maker's graph contains a copy of the goal-graph G). We first need some more notation. Recall that D denotes the directed graph where $V(D) = \{v_1, \ldots, v_n\}$ and $E(D) = \{(v_j, v_i) : \ i \in \mathcal{S}_j\}$. For every vertex v_j we let

$N_{\mathsf{out}}(v_j)$ denote the set of out-neighbors of v_j and we let $\mathsf{Desc}(v_j)$ denote the set of descendants of v_j.

During the game Maker will determine for each $j \in \{1, \dots, n\}$ a subset $B_j \subseteq V_j$ with $|B_j| = c_d$ (which is untouched at the time of its determination). For a vertex v_j where for every $k \in \mathcal{S}_j \cup \{j\}$ the set B_k has already been determined, we say that v_j is *completed* if Maker won the game G_{B_j}, i.e., if every $x^\star \in B_j$ is a candidate with respect to $\{B_k : k \in \mathcal{S}_j\}$. Conversely, we say that v_j is *spoiled* if all edges of the board of G_{B_j} have been claimed and at least one $x^\star \in B_j$ is not a candidate with respect to $\{B_k : k \in \mathcal{S}_j\}$.

Note that, as soon as $v_1, \dots, v_n$ are all completed, $(B_1, \dots, B_n)$ form a candidate scheme, and thus Maker won.

Furthermore, we call a vertex v_j *ready* if every vertex v_k where $k \in \mathcal{S}_j$ is completed. In other words, v_j is ready if all vertices in $N_{\mathsf{out}}(v_j)$ are completed. In the following, for every ready vertex v_j we abbreviate by "G_{B_j}" the game G_{B_j} with respect to $\{B_k : k \in \mathcal{S}_j\}$ (note that since v_j is ready the B_k are all determined). Maker's strategy will have the property that every completed vertex is also ready.

Maker's Strategy At the very beginning of the game, for every v_j where $\mathcal{S}_j = \emptyset$, Maker chooses B_j to be an arbitrary subset of V_j where $|B_j| = c_d$. (Thus, by definition, v_j is completed.) By slightly abusing notation, we refer to this as round zero. For every vertex v_i which became ready in round zero, Maker also determines an arbitrary $B_i \subseteq V_i$ of size c_d. Maker will proceed in such a way that after each round r the following two invariants are maintained.

(I1) For every vertex v_j which is not ready yet, at least $c_d + 2$ vertices of V_j are untouched.

(I2) Suppose that a vertex v_j became ready in round r (i.e., v_j was not ready after round $r-1$). Then, after round r Maker selects a subset $S \subseteq V_j$ of c_d untouched vertices and sets $B_j := S$.

At the very beginning of the game (i.e., after round 0) the invariants (I1) and (I2) clearly hold. Suppose that $r-1$ rounds have been played. By induction we can assume that (I1) and (I2) are fulfilled so far.

Assume that in round r Breaker claims the edge $(x^\star, y^\star)$, and suppose that $x^\star \in V_i$, $y^\star \in V_j$ and $i < j$. If some vertex v_k is spoiled at this point, Maker immediately forfeits. Otherwise we distinguish three cases.

Case 1: v_j is ready and $(x^\star, y^\star) \in E_H(B_i, B_j)$. Then Maker responds in the game G_{B_j}. (If all edges of the board of G_{B_j} are occupied, Maker claims any edge in some game G_{B_k}; no extra move is disadvantageous for him.)

Case 2: v_j is ready and $(x^\star, y^\star) \notin E_H(B_i, B_j)$. Then Maker claims any edge in some game G_{B_k}.

Case 3: v_j is not ready. Then there is at least one vertex $v_k \in \mathsf{Desc}(v_j)$ such that v_k is ready but not completed. (This can be seen as follows. By definition of readiness, $N_{\mathsf{out}}(v_j) \subseteq \mathsf{Desc}(v_j)$ contains at least one vertex which is not completed. Let v_k be a non-completed vertex in $\mathsf{Desc}(v_j)$ where k is minimal. Every $v_l \in N_{\mathsf{out}}(v_k)$ has the property that $l < k$, hence by minimality all vertices in $N_{\mathsf{out}}(v_k)$ are completed, thus v_k is ready.) By our choice of v_k and the assumption that Maker did not forfeit, v_k is neither completed nor spoiled, implying that not all edges of the board of G_{B_k} are occupied. Maker claims any free edge in the board of G_{B_k}. (Figure 4.6 depicts one such $v_k \in \mathsf{Desc}(v_j)$.)

In any case, if a vertex v_k became ready in round r then Maker chooses a subset $S \subseteq V_k$ of $c_d + 2$ vertices which were all untouched after round $r-1$ (such an S exists due to (I1)). Note that at most two vertices $z_1^\star, z_2^\star \in S$ became touched in round r. Maker sets $B_k := S \backslash \{z_1^\star, z_2^\star\}$.

Checking the Invariants Assuming that Maker did not forfeit, the invariant (I2) is clearly fulfilled after round r. To prove (I1), we fix a vertex v_j which is non-ready after round r. Note that due to his strategy Maker did not claim any edge incident to a vertex in V_j so far. We observe that every time a vertex $y^\star \in V_j$ was touched by Breaker, Maker selected some (ready) vertex $v_k \in \mathsf{Desc}(v_j)$ and occupied an edge in the board of G_{B_k} (here it is crucial that a vertex $y^\star \in V_j$ is only called touched if some player claimed an edge in $E(\{y^\star\}, V_1 \cup \ldots \cup V_{j-1})$). We fix such a vertex v_k and set $f(y^\star) := v_k$. Then the number of touched vertices in V_j (after r rounds) is bounded by the number of preimages of $\mathsf{Desc}(v_j)$ under f. We have

$$|f^{-1}(\mathsf{Desc}(v_j))| = |\cup_{v_k \in \mathsf{Desc}(v_j)} f^{-1}(\{v_k\})|.$$

By Observation 4.13, the board size of every G_{B_k} is at most dc_d^2, and thus,

$$|f^{-1}(\{v_k\})| \leq dc_d^2.$$

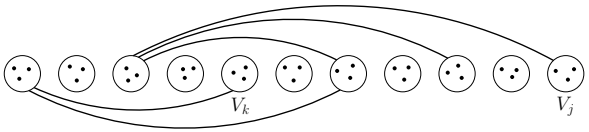


Figure 4.6: An example for an index k with $v_k \in \mathsf{Desc}(v_j)$. (A thick line indicates that the two corresponding vertex sets form a complete bipartite graph.)

Hence, by Observation 4.6 we obtain that

$$|f^{-1}(\mathsf{Desc}(v_j))| \leq dc_d^2|\mathsf{Desc}(v_j)| \leq dc_d^2 d^{2d^2+2},$$

which together with (4.5) and (4.6) implies that at least $c_d + 2$ vertices in V_j remain untouched. Hence (I1) is also fulfilled after round r.

We finally show that Maker did not forfeit in round r. Let v_j be a vertex which was ready after round $r-1$ (note that a non-ready vertex v_k can not be spoiled since B_k is not determined yet). By assumption we have that every v_i where $i \in \mathcal{S}_j$ is completed. Assume that v_j became ready in round $r' \leq r-1$. It can be checked that (due to Maker's strategy and the definition of readiness) after round r' the conditions of Corollary 4.16 were satisfied. Note that Maker's strategy has the property that from this point on, whenever Breaker claims an edge of the board of G_{B_j}, Maker will respond in G_{B_j}. In this way Maker can treat each G_{B_i} separately, following the strategy of Corollary 4.16. This guarantees that Maker will eventually win G_{B_j}. Hence v_j can not become spoiled. Thus, Maker did not forfeit in round r. By induction we get the following.

Observation 4.18. *Throughout the game, no vertex becomes spoiled. Hence Maker never forfeits.*

Proof of Maker's Win We first note that Maker can always make a move according to his strategy unless all edges of the boards of the G_{B_k} are occupied. We now show by induction that every vertex v_i eventually becomes completed. For $i = 1$ the claim is clearly true (note that v_1 is already completed at the very beginning of the game). Let $i \geq 2$ and let r be the smallest integer such that after round r, $v_1, \ldots, v_{i-1}$ are all completed (such an r exists by induction). Hence, after round r the vertex v_i is ready. As long as v_i is neither completed nor spoiled, there

are still free edges in the board of G_{B_i}, guaranteeing that Maker can make a move according to his strategy. Since the board H is finite, v_i eventually becomes either completed or spoiled. Observation 4.18 rules out the latter, implying that v_i finally becomes completed.

Hence, at some point, $v_1, \ldots, v_n$ are all completed, which yields that $(B_1, \ldots, B_n)$ form a candidate scheme. By Lemma 4.10 this guarantees that Maker's graph contains a copy of G. This concludes the proof of Theorem 4.1. □

5

Special Trees With Implications to SAT and Games

5.1 Introduction

5.1.1 Trees

We first consider a special class of trees, which connect both to SAT and games. Throughout this chapter, by a *binary tree* we mean a rooted tree where every node has either two or no children. We say that a leaf v of a binary tree is *l-close* to a node w if w is an ancestor of v, at distance at most l from v. For positive integers k and d, we call a binary tree T a *(k,d)-tree* if (i) every leaf has depth at least k and (ii) for every node u of T there are at most d leaves that are k-close to u. Clearly, every binary tree with all leaves at depth at least k is a $(k,2^k)$-tree. The following lemma will be the main ingredient in proving some new results on games and SAT.

Lemma 5.1. *A* $\left(k, \left(\frac{2}{e} + O(1/\sqrt{k})\right)\frac{2^k}{k}\right)$*-tree exists.*

We will use the following observation a few times.

Observation 5.2. *Lemma 5.1 gives that a* $\left(k-1, \left(\frac{1}{e}+O(1/\sqrt{k})\right)\frac{2^k}{k}\right)$*-tree exists.*

5.1.2 Games

Let $\mathcal{F}$ be a k-uniform hypergraph. The *degree* $d(v)$ of a vertex v is the number of hyperedges containing v and the *maximum degree* $\Delta(\mathcal{F})$ of a hypergraph $\mathcal{F}$ is the maximum degree of its vertices. The *neighborhood* $N(e)$ of a hyperedge e is the set of hyperedges of $\mathcal{F}$ which intersect e, excluding e itself, and the *maximum neighborhood size* of $\mathcal{F}$ is the maximum of $|N(e)|$ where e runs over all hyperedges of $\mathcal{F}$.

The famous Erdős-Selfridge Theorem [42] states that for each k-uniform hypergraph $\mathcal{F}$ with less than 2^{k-1} hyperedges Breaker has a winning strategy. This upper bound on the number of hyperedges is best possible as the following example shows. Let T be a full binary tree with k levels and let $\mathcal{G}$ be the hypergraph whose hyperedges are exactly the sets $\{v_0, \dots v_{k-1}\}$ such that $v_0, \dots, v_{k-1}$ is a path from the root to a leaf. Note that the number of hyperedges of $\mathcal{G}$ is 2^{k-1}. By first claiming the root and then pairing every vertex with its sibling (i.e. the vertex having the same parent) Maker can finally occupy all vertices on some path from the root to a leaf, which by construction contains a hyperedge. Hence Maker has a winning strategy on $\mathcal{G}$.

Note that the maximum degree of $\mathcal{G}$ is 2^{k-1}, thus equally large as the number of hyperedges of $\mathcal{G}$. This provides some evidence that in order to be a Maker's win a hypergraph must have largely overlapping hyperedges. Moreover, Beck [21] conjectured that the main criterion for whether a hypergraph is a Breaker's win is not the cardinality of the hyperedge set but rather the maximum neighborhood size, i.e., the actual reason why each hypergraph $\mathcal{H}$ with less than 2^{k-1} edges is a Breaker's win is that the maximum neighborhood size of $\mathcal{H}$ is smaller than $2^{k-1}-1$.

Neighborhood Conjecture. *(Open Problem 9.1(a), [21]) Assume that $\mathcal{F}$ is a k-uniform hypergraph, and its maximum neighborhood size is smaller than $2^{k-1}-1$. Is it true that Breaker has a winning strategy on $\mathcal{F}$?*

The hypergraph $\mathcal{G}$ considered above yields that if the Neighborhood Conjecture is true then it is also best possible. The best known result in the direction of the Neighborhood Conjecture is that if the maximum *degree* of a k-uniform hypergraph is at most $\lfloor \frac{k}{2} \rfloor$ then Breaker has a

winning strategy. This can be seen as follows. Hall's Theorem asserts that one can assign to every hyperedge two of its vertices such that every vertex is assigned to at most one hyperedge. By pairing the two vertices assigned to the same hyperedge Breaker can occupy at least one vertex in every hyperedge, thus he has a winning strategy.

Further motivation for the Neighborhood Conjecture is the well-known Erdős-Lovász 2-coloring Theorem – a direct consequence of the Lovász Local Lemma – which states that every k-uniform hypergraph with maximum neighborhood size at most $\frac{2^{k-1}}{e} - 1$ has a proper 2-coloring. An interesting feature of this theorem is that the size of the hypergraph does not matter. By another application of the Local Lemma we prove (in Theorem 5.7) that every k-uniform hypergraph with maximum degree at most $\frac{2^{k-2}}{ek}$ has a so called proper *halving* 2-coloring, i.e., a proper 2-coloring where the number of red vertices and the number of blue vertices differ by at most 1. This guarantees the *existence* of a course of the game such that at the end Breaker owns at least one vertex of each hyperedge and thus is the winner. Moreover, the existence of a proper halving 2-coloring is a necessary condition for Breaker having a winning strategy. Indeed, assume for a contradiction that Breaker has a winning strategy on a hypergraph $\mathcal{F}$ which does not admit a proper halving 2-coloring. Suppose further that during the game Maker colors his vertices red and Breaker colors his vertices blue. Note that by the end of the game the vertices are colored in such a way that the number of red vertices and the number of blue vertices differ by at most 1. By assumption Breaker has a winning strategy, thus every hyperedge contains at least one blue vertex. Since $\mathcal{F}$ does not admit a proper halving 2-coloring there must be at least one hyperedge containing only blue vertices. Since Maker starts the game he can now "steal" Breaker's strategy by starting with an arbitrary first vertex and then following Breaker's strategy (if this strategy calls for a vertex he occupied before he takes an arbitrary free vertex: no extra move is disadvantageous for him). This enables him to occupy all vertices of some hyperedge.

We now describe a connection between (k, d)-trees and games, which will allow us to disprove the Neighborhood Conjecture, in this strongest of its forms.

Connection to Trees Let T be a binary tree where every leaf has depth at least $k-1$. By $\mathcal{H}_T = \mathcal{H}_T(k)$ we denote the k-uniform hypergraph whose hyperedges are the vertex sets of paths that start at a leaf and go up $k-1$ levels. Note that the set of non-root nodes of T can be divided

into pairs of siblings. So by first claiming the root of T and then pairing every node with its sibling Maker can finally occupy all vertices on some path from the root to a leaf, which by assumption contains a hyperedge. Hence we have the following.

Observation 5.3. *Let T be a binary tree where every leaf has depth at least $k-1$. Then Maker has a winning pairing strategy on $\mathcal{H}_T$.*

The next observation is a direct consequence of Observation 5.3 and the defining property of (k,d)-trees.

Observation 5.4. *Let T be a $(k-1,d)$-tree. Then*

(i) Maker has a winning pairing strategy on $\mathcal{H}_T$, and

(ii) every vertex of $\mathcal{H}_T$ occurs in at most d hyperedges.

Observation 5.2 and Observation 5.4 imply the next corollary, which disproves the Neighborhood Conjecture. Note that the maximum neighborhood size of a k-uniform hypergraph $\mathcal{H}$ is upper bounded by $k \cdot \Delta(\mathcal{H})$.

Corollary 5.5. *For every k there is a k-uniform hypergraph $\mathcal{H}$ with maximum degree $\left(1+O(1/\sqrt{k})\right)\frac{2^k}{ek}$ where Maker has a winning pairing strategy. In particular, $\mathcal{H}$ has maximum neighborhood size at most $\left(1+O(1/\sqrt{k})\right)\frac{2^k}{e}$.*

We will point out in Section 5.2 that we can actually push down the second bound of Corollary 5.5 by a factor of 2. The construction of $\mathcal{H}$ relies on the proof of Lemma 5.1, which is rather involved and long. So we will additionally give a second counterexample for the Neighborhood Conjecture which is slightly weaker (in the sense that the hypergraph has more hyperedges) but can be obtained by a simpler construction.

In his book [21] Beck also poses several weaker versions of the Neighborhood Conjecture. The last one is as follows.

Open Problem 5.6. *(Open Problem 9.1(f), [21]) How about if we just want a proper halving 2-coloring?*

Recall that a proper halving 2-coloring is a proper 2-coloring where the number of red vertices and the number of blue vertices differ by at most one. It is already known [21] that the answer to Open Problem 5.6 is positive if the maximum degree is at most $\left(\frac{3}{2}-o(1)\right)^k$. According to Beck [21] the real question is whether or not $\frac{3}{2}$ can be replaced by 2. We prove that the answer is yes.

Theorem 5.7. *Let k be a large enough integer. For every k-uniform hypergraph $\mathcal{F}$ with maximum degree at most $\frac{2^{k-2}}{ek}$ there is a proper halving 2-coloring.*

5.1.3 SAT

The satisfiability of Boolean formulas, often abbreviated by SAT, is widely considered the 'mother' of all NP-complete problems. Following the standard notation, a k-CNF formula is the conjunction of clauses that are the disjunction of exactly k distinct literals; and a (k, s)-CNF formula is a k-CNF formula where every variable appears in at most s clauses. A pair of clauses sharing at least one variable is called an *intersecting pair*.

The largest integer $f(k)$ for which every $(k, f(k))$-CNF formula is satisfiable, is a value with many interesting properties, which has been widely studied. The best known lower bound on $f(k)$, a consequence of the Lovász Local Lemma, is due to Kratochvíl, Savický, and Tuza [58]:

$$f(k) \geq \left\lfloor \frac{2^k}{ek} \right\rfloor. \tag{5.1}$$

The previously best known upper bound is due to Hoory and Szeider [53] who showed that $f(k) \leq O\left(\log k \cdot \frac{2^k}{k}\right)$. We determine $f(k)$ up to lower order terms.

Theorem 5.8.

$$f(k) = \left(\frac{2}{e} + O\left(\frac{1}{\sqrt{k}}\right)\right)\frac{2^k}{k}.$$

So (5.1) can be strengthen by a factor of two and this bound is tight. To prove the upper bound of Theorem 5.8 we will use Lemma 5.1 and show that we can associate with every (k, d)-tree an unsatisfiable (k, d)-CNF formula.

The lower bound is achieved via the lopsided version of the Local Lemma. The key of the proof is to assign the random values of the variables counter-intuitively: each variable is *more* probable to satisfy those clauses where it appears as a literal with its *less* frequent sign. The lower bound can also be derived from a theorem of Berman, Karpinski and Scott [24] tailored to give good lower bounds on $f(k)$ for small values of k. In [24] the asymptotic behavior of the bound is not calculated, since the authors do not believe in its optimality. In Section 5.4 we

reproduce a simple argument giving these asymptotics, because the proof of [24] contains a couple of inaccuracies, so the unusual choice of the probabilities is not apparent.

We now briefly give the intuition of where the factor two improvement is coming from and how to achieve it. The lopsided version of the Local Lemma [43] allows for a more restricted definition of "intersecting" clauses in a CNF formula. Namely, one can consider two clauses intersect only if they contain a common variable with *different sign* and this still allows the same conclusion as in the original Local Lemma. If all variables in a (k, s)-CNF formula are *balanced*, that is they appear an equal number of times with either sign, then each clause intersects only at most $ks/2$ other clauses in this restricted sense, instead of the at most $k(s-1)$ other clauses it may intersect in the original sense and the factor two improvement is immediate. To handle the unbalanced case we consider a distribution on assignments where the variables are assigned true or false values with some bias. It would be natural to favor the assignment that satisfies more clauses, but the opposite turns out to be the distribution that works. This is because the clauses with some variables receiving the *less frequent sign* are those that intersect more than average other clauses, so those are the ones whose satisfiability should be boosted with the bias put on the assignments.

Bounded Neighborhood Size For a clause C of a CNF formula the *neighborhood* $\Gamma(C)$ of C denotes the set of other clauses (excluding C itself) intersecting C. Analogously to $f(k)$, we let $l(k)$ denote the largest integer r such that every k-CNF formula $\mathcal{F}$ for which $|\Gamma(C)| \leq r$, for every clause C of $\mathcal{F}$, is satisfiable. An immediate consequence of the Local Lemma states that if $|\Gamma(C)| \leq 2^k/e - 1$ for every clause C of a k-CNF formula $\mathcal{F}$ then $\mathcal{F}$ is satisfiable. Thus,

$$l(k) \geq \left\lfloor \frac{2^k}{e} \right\rfloor - 1. \tag{5.2}$$

From the other side, the complete formula consisting of all 2^k clauses of size k over k variables is clearly unsatisfiable, implying that $l(k) < 2^k - 1$. Our construction for proving the upper bound of Theorem 5.8 also shows that (5.2) is tight (up to lower order terms).

Theorem 5.9.

$$l(k) = \left(\frac{1}{e} + O\left(\frac{1}{\sqrt{k}}\right)\right) 2^k.$$

One may wonder how the lower bound on $l(k)$ implied by the (original) Local Lemma is tight and no improvement can be achieved using the lopsided version. This is no surprise when we see that in the formulas we construct every pair of clauses that intersects in the original sense also does in the more restricted sense.

The Class MU(1)

The function $f(k)$ is not known to be computable. In order to still be able to upper bound its value, one tries to restrict to a smaller/simpler class of formulas. When looking for unsatisfiable (k, s)-CNF formulas it is naturally enough to consider *minimal unsatisfiable formulas*, i.e., unsatisfiable CNF formulas that become satisfiable if we delete any one of their clauses. The set of minimal unsatisfiable CNF formulas is denoted by MU. As observed by Tarsi (c.f. [1]), all formulas in MU have more clauses than variables, but some have only one more. The class of these MU formulas, having one more clauses than variables is denoted by MU(1). This class has been widely studied (see, e.g., [1], [38], [55], [60], [74]). Hoory and Szeider [54] considered the function $f_1(k)$, denoting the largest integer such that no $(k, f_1(k))$-CNF formula is in MU(1), and showed that $f_1(k)$ is computable. Their computer search determined the values of $f_1(k)$ for small k: $f_1(5) = 7$, $f_1(6) = 11$, $f_1(7) = 17$, $f_1(8) = 29$, $f_1(9) = 51$. Via the trivial inequality $f(k) \leq f_1(k)$, these are the best known upper bounds on $f(k)$ in this range. In contrast, even the value of $f(5)$ is not known.

It is an interesting open problem whether $f(k) = f_1(k)$ for every k. While the derivation of the previous bound of $f(k) = O(\log k \cdot \frac{2^k}{k})$ by Hoory and Szeider did not go via an MU(1) formula, our upper bound construction from Theorem 5.8 does reside in the class MU(1) and hence shows that $f(k)$ and $f_1(k)$ are equal asymptotically: $f(k) = (1 + o(1))f_1(k)$.

Scheder [71] showed that for almost disjoint k-CNF formulas (i.e. k-CNF formulas where any two clauses have at most one variable in common) the two functions are not the same. That is, if $\tilde{f}(k)$ denotes the maximum s such that every almost disjoint (k, s)-CNF formula is satisfiable, for k large enough every unsatisfiable almost disjoint $(k, \tilde{f}(k) + 1)$-CNF formula is outside of MU(1).

Connection to Trees.

For every binary tree T with all leaves having depth at least k we can construct a k-CNF formula as follows. For every non-leaf node $w \in V(T)$ we create a variable x_w and label one of its children with the literal x_w and the other with the negated version $\bar{x}_w$. Note that the root does not receive a literal. With every leaf $v \in V(T)$ we associate a clause C_v by walking along a path of length $k-1$ from v towards the root and taking the disjunction of all labels encountered on this path (i.e., the labels of all nodes to which v is $(k-1)$-close). Finally, we let $\mathcal{F}(T)$ denote the conjunction of all such clauses C_v. We will show in Section 5.3 that $\mathcal{F}(T)$ has the following properties.

Lemma 5.10. *Let $k \geq 1$ and $d \geq 1$ be integers and let T be a (k,d)-tree. Then $\mathcal{F} = \mathcal{F}(T)$ is an unsatisfiable (k,d)-CNF formula. If T is minimal with respect to the number of leaves then $\mathcal{F}$ belongs to* MU(1). *In particular, $f(k), f_1(k) \leq d-1$.*

Lemma 5.10 establishes a direct connection between (k,d)-trees and (k,d)-CNF formulas, which together with Lemma 5.1 immediately gives the upper bound of Theorem 5.8. However, it does not imply (at least not in an obvious way) the upper bound of Theorem 5.9: The main obstacle is that we cannot improve the immediate bound $|\Gamma(C)| \leq k(d-1)$ for a clause C of $\mathcal{F}(T)$ (in order to derive Theorem 5.9 we should have an upper bound for $|\Gamma(C)|$ which is around $\frac{kd}{2}$).

So we also need a second construction: For every $(k-1,d)$-tree T we let $\widehat{T}$ denote the binary tree obtained by attaching the roots of two copies of T as the children of a new root. Every leaf of $\widehat{T}$ has depth at least k and the number of leaves at distance at most k from a given non-leaf vertex w is equal to the number of leaves which are $(k-1)$-close to any child of w, thus at most $2d$. Hence $\widehat{T}$ is a $(k,2d)$-tree. We set

$$\mathcal{G}(T) = \mathcal{F}(\widehat{T}).$$

In Section 5.3 we will show that $\mathcal{G}(T)$ has the following properties.

Lemma 5.11. *Let $k \geq 2$ and $d \geq 1$ be integers and let T be a $(k-1,d)$-tree. Then $\mathcal{G} = \mathcal{G}(T)$ is an unsatisfiable $(k,2d)$-CNF formula with the following properties.*

(a) *Every literal occurs in at most d clauses of $\mathcal{G}$.*
(b) *For every two distinct clauses C, D having a variable in common there is a variable that appears in C and D with opposite signs.*

(c) *If T is minimal with respect to the number of leaves then $\mathcal{G}$ belongs to* MU(1).
(d) $|\Gamma(C)| \leq kd$ *for all clauses C of $\mathcal{G}$.*

In particular, $f(k), f_1(k) \leq 2d-1$ *and* $l(k) \leq kd-1$.

Note that Lemma 5.11 together with Observation 5.2 directly implies the upper bound of Theorem 5.9. We remark that Lemma 5.11 (coupled with Observation 5.2) also implies the upper bound of Theorem 5.8.

Implications on Trees Let $f_{\mathsf{tree}}(k)$ be the largest integer d such that no (k,d)-tree exists. Lemma 5.10 gives that

$$f(k) \leq f_1(k) \leq f_{\mathsf{tree}}(k). \tag{5.3}$$

Lemma 5.1 implies that $f_{\mathsf{tree}}(k) \leq \left(\frac{2}{e} + O(1/\sqrt{k})\right)\frac{2^k}{k}$. The lower bound of Theorem 5.8, coupled with (5.3), yields that this upper bound on $f_{\mathsf{tree}}(k)$ is tight (up to lower order terms). Hence we obtain the following.

Observation 5.12. $f_{\mathsf{tree}}(k) = \left(\frac{2}{e} + O(1/\sqrt{k})\right)\frac{2^k}{k} = (1+o(1))f(k)$.

5.1.4 Structure of this Chapter

In Section 5.2 we first consider Corollary 5.5, which disproves the Neighborhood Conjecture in the strongest of its forms, and point out that it can be improved. Additionally, we present a second, slightly weaker counterexample (in the sense that the hypergraph has more hyperedges) for the Neighborhood Conjecture which can be obtained by a much simpler construction. Finally, we show Theorem 5.7.

In Section 5.3 we establish a connection between the trees we study and SAT by proving Lemmas 5.10 and 5.11. In Section 5.4 we show the lower bound of Theorem 5.8.

The proof of Lemma 5.1 (which together with Lemmas 5.10 and 5.11 directly implies the upper bounds of Theorems 5.8 and 5.9) contains several technically involved arguments. However, a weaker statement, still settling the asymptotics of $f(k)$, can be shown with less effort: In Section 5.5 we present a short (and explicit) proof of the existence of $(k, \lfloor \frac{2^{k+3}}{k} \rfloor)$-trees for $k \geq 1$.

In Section 5.6 we give an informal proof of Lemma 5.1 in order to sketch the main ideas and intuitions behind our approach. To this end

we define a suitable continuous setting for the construction of the appropriate binary trees, which allows us to study the problem via a differential equation. The solution of this differential equation corresponds to our construction of the binary trees, which then can be given completely discretely. This is the subject of Section 5.7.

In Section 5.8 we finally give an outlook and state some open problems.

Notation Let T be a binary tree. Following the standard notation, the *level* (or, alternatively, the *depth*) of a vertex v denotes the distance from v to the root (so, in particular, the root has depth zero). A *path* of T is a sequence of vertices $v_1, v_2, \ldots, v_j$ of T where v_k is a child of v_{k-1} for every $k = 2, \ldots, j$. Depending on the context we consider a hyperedge e of a hypergraph $\mathcal{H}_T$ either as a set or as a path in T. So we will sometimes speak of the start or end node of a hyperedge.

5.2 Results on Games

Improving Corollary 5.5 Let $k \geq 3$ and let T be a $(k-2, d)$-tree. Recall that $\widehat{T}$ denotes the $(k-1, 2d)$-tree obtained by attaching the roots of two copies of T as the children of a new root. Clearly, for every node v there are at most d leaves $(k-2)$-close to v. As defined above, $\mathcal{H}_{\widehat{T}}$ denotes the k-uniform hypergraph obtained by associating with every leaf v_0 of $\widehat{T}$ a hyperedge e_{v_0} consisting of all vertices $v_0, v_1, \ldots, v_{k-1}$ on the path that starts at v_0 and goes up $k-1$ levels. Let e_{w_0} be any hyperedge intersecting e_{v_0} and let i be the smallest number such that v_i occurs both in e_{v_0} and e_{w_0}. By construction e_{w_0} contains the child v_i' of v_i which does *not* belong to e_{v_0}. Note that w_0 is $(k-2)$-close to v_i'.

For every $i \in \{1, \ldots, k-1\}$ let v_i' denote the child of v_i which does not belong to e_{v_0}. Since e_{w_0} was chosen arbitrarily, we get that every hyperedge intersecting e_{v_0} contains some v_i', hence $|N(e_{v_0})|$ is at most the number of leaves which are $(k-2)$-close to some v_i'. By construction, we have

$$|N(e_{v_0})| \leq (k-1)d. \tag{5.4}$$

By Lemma 5.1 there exists a $(k-2, d)$-tree for $d = \left(\frac{2}{e} + O(1/\sqrt{k})\right) \frac{2^{k-2}}{k-2}$. Together with (5.4) this gives that

$$|N(e_{v_0})| \leq \left(1 + O(1/\sqrt{k})\right) \frac{2^{k-1}}{e}.$$

Thus, the maximum neighborhood size of $\mathcal{H}_{\widehat{T}}$ is actually bounded by $\left(1 + O(1/\sqrt{k})\right) \frac{2^{k-1}}{e}$, improving the second bound in Corollary 5.5 by a factor of 2.

A Second Counterexample for the Neighborhood Conjecture Corollary 5.5 relies on Lemma 5.1, whose proof is rather long. However, a slightly weaker statement, still disproving the Neighborhood Conjecture, can be shown with less effort.

Proposition 5.13. *For every $k \geq 3$ there is a binary tree T where every leaf has depth at least $k-1$ such that $\mathcal{H}_T$ has maximum neighborhood size $2^{k-2} + 2^{k-3}$.*

Note that Observation 5.3 guarantees that Maker has a winning strategy on $\mathcal{H}_T$.

Proof of Proposition 5.13. Let T' be a full binary tree with $k-1$ levels. For each leaf u of T' we proceed as follows: We add two children v, w to u and let v be a leaf. Then we attach a full binary tree S with $k-2$ levels to w (such that w is the root of S). For each leaf u' of S we add two children v', w' to u' and let v' be a leaf. Note that the hyperedge ending at v' starts at u. Finally, we attach a full binary tree S' with $k-1$ levels to w' (such that w' is the root of S'), see Figure 5.1. Let T denote the resulting tree.

Clearly, every leaf of T has depth at least $k-1$. It remains to show that the maximum neighborhood size of $\mathcal{H}_T$ is at most $2^{k-2} + 2^{k-3}$.

Claim. *Every hyperedge e of $\mathcal{H}_T$ intersects at most $2^{k-2} + 2^{k-3}$ other hyperedges.*

In order to prove this claim, we fix six vertices u, u', v, v', w, w' according to the above description, i.e., u is a node on level $k-2$ whose children are v and w, u' is a descendant of w on level $2k-4$ whose children are v' and w'. Let e be a hyperedge of $\mathcal{H}_T$. Note that the start node of e is either the root r of T, a node on the same level as u or a node on the same level as u'. We now distinguish these cases.

(a) The start node of e is r (recall that r denotes the root). By symmetry we assume that e ends at v. According to the construction of T the hyperedge e intersects the $2^{k-2}-1$ other hyperedges starting at r and the 2^{k-3} hyperedges starting at u. So altogether e intersects $2^{k-2} + 2^{k-3} - 1$

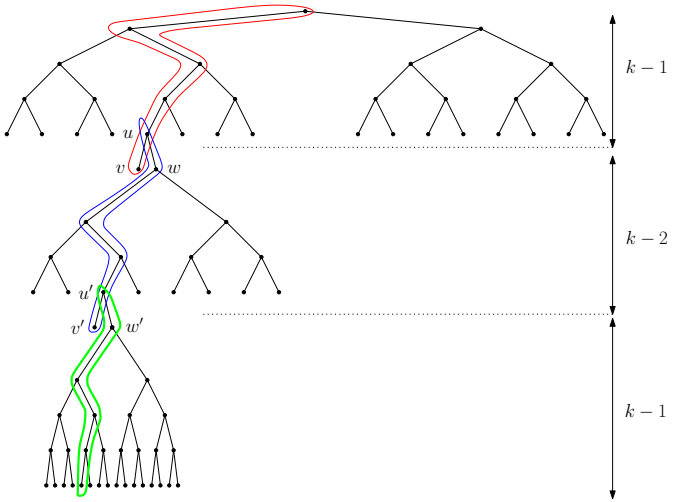


Figure 5.1: An illustration of $\mathcal{H}_T$. The marked paths represent exemplary hyperedges.

other hyperedges.

(b) The start node of e is on the same level as u. By symmetry we suppose that e starts at u and ends at v'. The hyperedges intersecting e can be divided into the following three categories.

- The hyperedge starting at r and ending at v,
- the $2^{k-3}-1$ hyperedges different from e starting at u, and
- the 2^{k-2} hyperedges starting at u',

implying that e intersects $2^{k-2}+2^{k-3}$ other hyperedges in total.

(c) The start node of e is on the same level as u'. By symmetry we assume that e starts at u'. Then e intersects the $2^{k-2}-1$ other hyperedges starting at u' and the hyperedge starting at u and ending at v', thus 2^{k-2} other hyperedges altogether. □

Establishing a Proper Halving 2-Coloring *Proof of Theorem 5.7:* Let $\mathcal{F}=(V,E)$. We can assume without loss of generality that $|V|$ is even.

(Otherwise we let V' be the vertex set obtained by adding a new dummy vertex x to V and we consider the hypergraph $\mathcal{F}' = (V', E)$ instead of $\mathcal{F}$. Since every proper halving 2-coloring of $\mathcal{F}'$ is also a proper halving 2-coloring of $\mathcal{F}$ it suffices to show that $\mathcal{F}'$ has as proper halving 2-coloring). We will show the following stronger claim.

Proposition 5.14. *Let k be a large enough integer and let $\mathcal{F} = (V, E)$ be a k-uniform hypergraph with $2r$ vertices and maximum degree at most $\frac{2^{k-2}}{ek}$. Then for every partition $(v_1, v_1'), \ldots, (v_r, v_r')$ of V into pairs, there is a proper 2-coloring such that v_i and v_i' have different colors for every i, $i = 1, \ldots, r$.*

Before starting with the proof we need some notation. First, let $P = (v_1, v_1'), \ldots, (v_r, v_r')$ be a partition of V into pairs. By a *(proper) P-2-coloring* we denote a (proper) 2-coloring of $\mathcal{F}$ such that v_i and v_i' have different colors for every i, $i = 1, \ldots, r$. Moreover, for every vertex $x \in V$ we denote by $f(x)$ the vertex which is paired with x in P (i.e., $f(v_i) = v_i'$ and $f(v_i') = v_i$).

Proof of Proposition 5.14: We fix a partition $P = (v_1, v_1'), \ldots, (v_r, v_r')$ of V into pairs. Our goal is to show that there is a proper P-2-coloring. For each hyperedge $e = (w_1, \ldots, w_k)$ we add e and $e' = (f(w_1), \ldots, f(w_k))$ and denote the resulting hypergraph by $\mathcal{F}'$. Note that $\Delta(\mathcal{F}') \leq 2 \cdot \Delta(\mathcal{F}) \leq \frac{2^{k-1}}{ek}$. We now transform $\mathcal{F}'$ into a SAT instance. For every hyperedge $e = (u_1, \ldots, u_k)$ of $\mathcal{F}'$ we form a clause $C_e = (u_1 \vee u_2 \ldots \vee u_k)$ and set $\mathcal{H} := \wedge_{e \in E(\mathcal{F}')} C_e$ with $E(\mathcal{F}')$ denoting the hyperedge set of $\mathcal{F}'$. Then we replace (in $\mathcal{H}$) v_i and v_i' with x_i and $\bar{x}_i$, respectively, for every i, $i = 1, \ldots, r$. Note that every variable x_i occurs in at most $2\Delta(\mathcal{F}') \leq \frac{2^k}{ek}$ clauses of $\mathcal{H}$. Due to Theorem 5.8 there is a satisfying assignment α of $\mathcal{H}$. Note that by construction, every clause C of $\mathcal{H}$ contains two literals x, y where $\alpha(x) = 1$ and $\alpha(y) = 0$. For every variable x_i where $i \in \{1, \ldots, r\}$ we proceed as follows. If $\alpha(x_i) = 1$ then we color v_i red and v_i' blue, otherwise we do it the other way round. Clearly, this yields a proper P-2-coloring of $\mathcal{F}'$. □

We note that the above proof implies that Theorem 5.7 remains true if we replace $\frac{2^{k-2}}{ek}$ with $(1 + O(1/\sqrt{k}))\frac{2^{k-1}}{ek}$.

5.3 Proof of Lemmas 5.10 and 5.11

In the proof of Lemmas 5.10 and 5.11 we will use a powerful characterization of MU(1)-formulas, established by Davydov, Davydova, and Kleine

Büning [38]. (Here a clause $C = (x_1 \vee x_2 \vee \ldots \vee x_k)$ is represented as the set $\{x_1, \ldots, x_k\}$ of its literals, and a CNF formula $\mathcal{F} = C_1 \wedge C_2 \wedge \ldots \wedge C_n$ is represented as the set $\{C_1, \ldots, C_n\}$ of its clauses. "$\mathsf{vbl}(\mathcal{F})$" denotes the set of variables which occur in $\mathcal{F}$.)

Lemma 5.15. (Davydov, Davydova, and Kleine Büning [38]) *$\mathcal{F} \in$ MU(1) if and only if either $\mathcal{F} = \{\emptyset\}$ or $\mathcal{F}$ is the disjoint union of formulas $\mathcal{F}'_1, \mathcal{F}'_2$ such that for a variable x we have*

- $\mathsf{vbl}(\mathcal{F}'_1) \cap \mathsf{vbl}(\mathcal{F}'_2) = \{x\}$ *and* $\{x, \bar{x}\} \subseteq \bigcup_{C \in \mathcal{F}} C$*;*
- $\mathcal{F}_1 := \{C \backslash \{x\} : C \in \mathcal{F}'_1\} \in$ MU(1)*;*
- $\mathcal{F}_2 := \{C \backslash \{\bar{x}\} : C \in \mathcal{F}'_2\} \in$ MU(1)*.*

Let T be a binary tree and suppose that, as in our construction of $\mathcal{F}(T)$, every vertex different from the root is labeled with a literal such that the two children of any non-leaf vertex are labeled with complementary literals. A CNF formula $\mathcal{G}$ is called a *T-formula* if $\mathcal{G}$ can be obtained by associating with every leaf v of T a clause C_v that is the disjunction of *some* literals along the path from v to the root, and taking the conjunction of all the C_v. Note that according to our construction, $\mathcal{F}(T)$ is a T-formula and $\mathcal{G}(T)$ is a $\widehat{T}$-formula.

The following characterization of MU(1)-formulas is an immediate (and known) consequence of Lemma 5.15.

Corollary 5.16. *A CNF formula $\mathcal{G}$ is in* MU(1) *if and only if $\mathcal{G}$ is a T-formula for some binary tree T where every literal associated to a vertex of T* does *appear in $\mathcal{G}$.*

Proof of Lemma 5.10: Let T be a (k, d)-tree and suppose that every non-leaf vertex w is assigned a variable x_w such that one child of w is labeled with the literal x_w and the other child is labeled with the literal $\bar{x}_w$. Recall that with every leaf $v \in V(T)$ we associate the clause C_v which is the disjunction of the first k labels encountered on the path from v to the root (including the label of v), and that $\mathcal{F}(T)$ denotes the conjunction of all such clauses C_v.

$\mathcal{F}$ is unsatisfiable, for if an assignment α over the variables of T is given, it defines a path from the root to a leaf, say v, by always proceeding to the unique child whose label is mapped to false by α; thus the clause C_v associated with v is violated by α. Moreover, by construction, for every fixed non-leaf node w the *variable* x_w occurs in exactly those clauses C_v where v is a leaf at distance at most k from w. Thus the

defining property of (k,d)-trees guarantees that every variable occurs in at most d clauses.

It remains to show the second claim of Lemma 5.10. Let T be a (k,d)-tree which is minimal with respect to the number of leaves. Due to the minimality of T, every non-root node w of T has at least one leaf descendant at distance at most $k-1$, since otherwise the subtree of T rooted at w would be a (k,d)-tree with fewer leaves than T. Thus, every literal associated to a node of T does appear in $\mathcal{F}$, and therefore Corollary 5.16 yields that $\mathcal{F}$ belongs to MU(1). □

Proof of Lemma 5.11: Recall that $\widehat{T}$ is the binary tree obtained by attaching the roots of two copies of T as the children of a new root, and $\mathcal{G}(T) = \mathcal{F}(\widehat{T})$. Thus, by Lemma 5.10, $\mathcal{G}$ is an unsatisfiable $(k,2d)$-CNF formula. Moreover, the defining property of $(k-1,d)$-trees asserts that no literal appears in more than d clauses (hence (a)). We now settle (b). Let C_u, C_v be two clauses sharing at least one variable and let w denote the lowest common ancestor of u and v (i.e. the node of maximum depth that appears on both paths from u and v, respectively, to the root). Then one child of w occurs in C_u whereas the other child occurs in C_v. Since siblings have complementary literals (b) is shown.

Next we prove (c). Note that $\mathcal{G}$ is a $\widehat{T}$-formula. Due to the minimality of T, every non-root node w of T has at least one leaf descendant at distance at most $k-2$, since otherwise the subtree of T rooted at w would be a $(k-1,d)$-tree with fewer leaves than T. Hence every node of T (including the root) has at least one leaf descendant at distance at most $k-1$, and thus every literal associated to a vertex of $\widehat{T}$ does appear in $\mathcal{G}$. Thus, by Corollary 5.16 $\mathcal{G}$ belongs to MU(1).

(d) follows from (a) and (b): Indeed, if we define $\mathrm{occ}(u)$ as the number of clauses of $\mathcal{G}$ containing a literal u, and if we abbreviate "the clause C contains the literal u" by "$u \in C$", then (b) allows us to write $|\Gamma(C)|$ as $\sum_{u\in C} \mathrm{occ}(\overline{u})$, which is at most kd for every clause C of $\mathcal{G}$. □

5.4 Proof of the Lower Bound of Theorem 5.8

For our proof we use the Lopsided Local Lemma of Erdős and Spencer [43].

Lemma 5.17. *(Lopsided Local Lemma) Let $\{A_C\}_{C\in I}$ be a finite set of events in some probability space. Let $\Gamma(C)$ be a subset of I for each $C \in I$ such that for every subset $J \subseteq I \setminus (\Gamma(C) \cup \{C\})$ we have*

$$Pr(A_C | \wedge_{D\in J} \bar{A}_D) \leq Pr(A_C).$$

Suppose there are real numbers $0 < x_C < 1$ *for* $C \in I$ *such that for every* $C \in I$ *we have*

$$Pr(A_C) \leq x_C \prod_{D \in \Gamma(C)} (1 - x_D).$$

Then

$$Pr(\wedge_{C \in I} \bar{A}_C) > 0.$$

Let $\mathcal{F}$ be a (k, s)-CNF formula with $s = \left\lfloor \frac{2^{k+1}}{e(k+1)} \right\rfloor$.

We set the values of the variables randomly and independently, but not according to the uniform distribution. This seems reasonable to do as the number of appearances of a variable x_i in $\mathcal{F}$ as a non-negated literal could be significantly different from the number of clauses where x_i appears negated. It is even possible that a variable x_i appears negated in only a few, maybe even in a single clause, in which case one tends to think that it is reasonable to set this variable to true with much larger probability than setting it to false. In fact it is exactly the opposite we will do. The *more* a variable appears in the clauses of $\mathcal{F}$ as non-negated, *the less likely* we will set it to true. The intuition behind this is explained in the introduction of this chapter.

For a literal v we denote by d_v the number of occurrences of v in $\mathcal{F}$. We set a variable x to true with probability $P_x = \frac{1}{2} + \frac{2d_{\bar{x}} - s}{2sk}$. This makes the negated version $\bar{x}$ satisfied with probability $P_{\bar{x}} = \frac{1}{2} - \frac{2d_{\bar{x}} - s}{2sk} \geq \frac{1}{2} + \frac{2d_x - s}{2sk}$ as we have $d_x + d_{\bar{x}} \leq s$. So any literal v is satisfied with probability at least $\frac{1}{2} + \frac{2d_{\bar{v}} - s}{2sk}$.

For each clause C in $\mathcal{F}$, we define the "bad event" A_C to be that C is not satisfied. Moreover, for every C in $\mathcal{F}$ we define $\Gamma(C)$ to be the family of clauses D in $\mathcal{F}$ that have at least one such variable in common with C whose sign is different in C and D. Finally we set the value of each x_C to be $x = \frac{e}{2^k}$.

We need to check that for every subset $J \subseteq I \setminus (\Gamma(C) \cup \{C\})$ we have

$$Pr(A_C | \wedge_{D \in J} \bar{A}_D) \leq Pr(A_C).$$

This can be seen as follows. If C is not satisfied then all literals in C are set to false. Thus, $Pr(\wedge_{D \in J} \bar{A}_D | A_C) \leq Pr(\wedge_{D \in J} \bar{A}_D)$. Therefore,

$$\begin{aligned} Pr(A_C | \wedge_{D \in J} \bar{A}_D) &= \frac{Pr(A_C \bigwedge (\wedge_{D \in J} \bar{A}_D))}{Pr(\wedge_{D \in J} \bar{A}_D)} \\ &= \frac{Pr(\wedge_{D \in J} \bar{A}_D | A_C) \cdot Pr(A_C)}{Pr\left(\wedge_{D \in J} \bar{A}_D\right)} \leq Pr(A_C). \end{aligned}$$

We need to check also the other condition of the lemma. Let C be an arbitrary clause and let us denote the literals it contains by $v_1, \ldots, v_k$. For C not to be satisfied we must not set any of the independent literals in C to true and therefore we have

$$\begin{aligned} Pr(A_C) &= \prod_{i=1}^{k}(1 - P_{v_i}) \\ &\leq \prod_{i=1}^{k}\left(\frac{1}{2} - \frac{2d_{\bar{v}_i} - s}{2sk}\right) \\ &\leq \frac{1}{2^k}\prod_{i=1}^{k}\left(\left(1 + \frac{1}{k}\right)\left(1 - \frac{ed_{\bar{v}_i}}{2^k}\right)\right) \\ &\leq \frac{\left(1 + \frac{1}{k}\right)^k}{2^k}\prod_{i=1}^{k}(1 - x)^{d_{\bar{v}_i}} \\ &< \frac{e}{2^k}(1 - x)^{|\Gamma(C)|} \\ &= x\prod_{D \in \Gamma(C)}(1 - x). \end{aligned}$$

The inequality in the fourth line holds due to the well-known inequality that $1 - ax \leq (1 - x)^a$ for every $0 < x < 1$ and $a \geq 1$. As the conditions of the Lopsided Local Lemma are satisfied, its conclusion must also hold. It states that the random evaluation of the variables we consider satisfies the (k, s)-CNF $\mathcal{F}$ with positive probability. Thus $\mathcal{F}$ must be satisfiable and we have $f(k) \geq s = \left\lfloor \frac{2^{k+1}}{e(k+1)} \right\rfloor$.

5.5 Proof of a Weaker Version of Lemma 5.1

We need some notation first. Let T be a binary tree (not necessarily with all leaves having depth at least k) and let v be a vertex of T. In the following we denote by the *degree* $d(v)$ of v the number of leaf descendants which have distance at most k from v. As a first step towards proving Lemma 5.1 we show the following weaker bound, which already settles the asymptotics of $f(k)$.

Proposition 5.18. *A $(k, \lfloor \frac{2^{k+3}}{k} \rfloor)$-tree exists for every $k \geq 1$.*

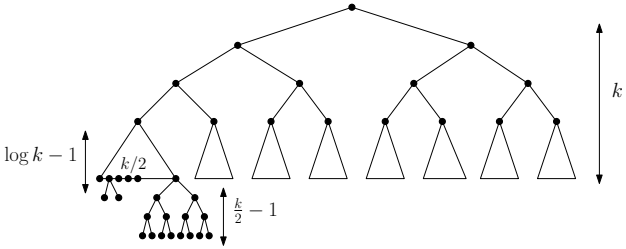


Figure 5.2: The construction of T where k is a power of 2.

Proof: Note that the full binary tree of height k is a $(k, 2^k)$-tree. We have $\lfloor \frac{2^{k+3}}{k} \rfloor \geq 2^k$ for $k \leq 8$ so we can assume that $k > 8$. Let $r = 2^{k+2-\lfloor \log k \rfloor}$ and note that $r \leq \lfloor \frac{2^{k+3}}{k} \rfloor$. Let T' be a full binary tree of height k. We subdivide its leaves into $s = 2^{k-\lfloor \log k \rfloor + 1}$ intervals $I_1, \ldots, I_s$ of length $2^{\lfloor \log k \rfloor - 1}$ each (i.e., an interval I_j contains all leaf descendants of some vertex on level $k - (\lfloor \log k \rfloor - 1)$). For every such interval $I_j = \{v_0, \ldots, v_{2^{\lfloor \log k \rfloor - 1} - 1}\}$ and every $v_i \in I_j$ we attach a full binary subtree of height i to v_i. Let T denote the resulting tree. Figure 5.2 shows an illustration for the case where k is a power of 2. It suffices to prove the following.

Proposition 5.19. *Let v be a vertex of T. Then $d(v) \leq r$.*

Proof: We apply induction on the depth i of v. For $i = 0$ the claim is clearly true. Indeed, the degree of the root is $\frac{2^k}{2^{\lfloor \log k \rfloor - 1}} = 2^{k+1-\lfloor \log k \rfloor} = \frac{r}{2}$. Now suppose that v has depth $i \in \{1, \ldots, 2^{\lfloor \log k \rfloor - 1} - 1\}$. Note that the set of descendants of v on level k can be subdivided into $\frac{2^{k-i}}{2^{\lfloor \log k \rfloor - 1}}$ intervals, i.e., at least one interval for the values of i we consider. Let v' denote the parent of v. By construction the number of leaf descendants which have distance at most $k-1$ from v equals $\frac{d(v')}{2}$. Moreover, every interval $\{v_0, \ldots, v_{2^{\lfloor \log k \rfloor - 1} - 1}\}$ gives rise to 2^i leaves on level $k+i$, implying that the number of leaf descendants of v which have distance exactly k from v equals $\frac{2^{k-i}}{2^{\lfloor \log k \rfloor - 1}} \cdot 2^i = 2^{k+1-\lfloor \log k \rfloor} = \frac{r}{2}$. So altogether, $d(v) \leq \frac{d(v')}{2} + \frac{r}{2} \leq r$. It remains to consider the case where v has depth at least $2^{\lfloor \log k \rfloor - 1}$. By construction no leaf of T has depth larger than $k + 2^{\lfloor \log k \rfloor - 1} - 1$, implying that the degree of v is at most the degree of its parent. □

5.6 Informal Proof of Lemma 5.1 via a Continuous Construction

In this section we give most of the formal definitions and sketch the main informal ideas behind the construction of the tree of Lemma 5.1. Even though the final construction can be formulated without mentioning the underlying continuous context, we feel that an informal description greatly helps in motivating it.

5.6.1 Vectors and Constructibility

Given a binary tree T, we assign a *leaf-vector* $\vec{d}_w = (x_0, x_1, \ldots, x_k)$ to each node $w \in V(T)$, where x_i denotes the number of leaf descendants of w having distance i from w. E.g., for a leaf w we have $\vec{d}_w = (1, 0, \ldots, 0)$, and for the root w of a full binary tree of height $l \leq k$ we have $\vec{d}_w = (0, 0, \ldots, 0, 2^l, 0, \ldots, 0)$. For every vector $\vec{x} = (x_0, \ldots, x_k)$ we set $|\vec{x}| = \sum_{i=0}^{k} x_i$. By definition, for every node w of a (k, d)-tree we have $|\vec{d}_w| \leq d$.

For some vector $\vec{x}$ with $|\vec{x}| \leq d$ we define a $(k, d, \vec{x})$-*tree* as a binary tree where (i) the root has a leaf-vector $\vec{y}$ which is coordinate-wise dominated by $\vec{x}$ (i.e., $y_i \leq x_i$ for every i), and (ii) each vertex has at most d leaf descendants that are k-close. E.g., a tree consisting of a parent with two children is a $(k, d, \vec{x})$-tree, for every $\vec{x}$ with $x_1 \geq 2$ and $\sum_{i=0}^{k} x_i \leq d$. A vector $\vec{x}$ is (k, d)-*constructible* (or *constructible* if k and d are clear from the context), if a $(k, d, \vec{x})$-tree exists. E.g., $(1, 0, \ldots, 0)$, or more generally $(\underbrace{0, 0, \ldots, 0}_{l}, 2^l, 0, \ldots, 0)$ are constructible as long as $2^l \leq d$. Observe that the constructibility of the vector $(0, \ldots, 0, r)$ for some $r \leq d$ readily implies the existence of a (k, d)-tree.

If $|\vec{x}| \leq d$ then $\vec{x}$ is a (k, d)-*vector*. For a vector $\vec{x} = (x_0, \ldots, x_k)$ the *weight* $w(\vec{x})$ is $\sum_{i=0}^{k} x_i/2^i$. The next lemma gives a useful sufficient condition for the constructibility of a vector.

Lemma 5.20. *Let $\vec{x}$ be a (k, d)-vector. If $w(\vec{x}) \geq 1$ then $\vec{x}$ is constructible.*

We note that Lemma 5.20 is a direct consequence of Kraft's inequality. For completeness we give a proof here.

Proof of Lemma 5.20: We build a binary tree starting with the root and adding the levels one by one. Among the vertices on level i we select a set of x_i vertices and let them be leaves. We construct the next level

by adding two children to the remaining $2^i(1-\sum_{j=0}^{i}\frac{x_j}{2^j})$ vertices on level i. At the first level i where $\sum_{j=0}^{i}\frac{x_j}{2^j} \geq 1$ we mark all vertices as leaves and stop the construction of the tree. The total number of leaves is at most $\sum_{i=0}^{k} x_i \leq d$ and the number of leaves at distance i from the root is at most x_i, so the constructed tree is clearly a $(k, d, \vec{x})$-tree. □

In our new terminology, the proof of Proposition 5.18 establishes the constructibility of the vector $\vec{v} = (0, \ldots, 0, 1, 2, \ldots, 2^s)$, provided $s = k + 1 - \lfloor \log k \rfloor$ and $d \geq 2^{s+1}$. This is an immediate consequence of Lemma 5.20. By considering a full binary tree T with all leaves at depth s and attaching a $(k, d, \vec{v})$-tree to every leaf l of T (such that l is the root) one can obtain the constructibility of $(0, \ldots, 0, 2^s)$, which directly implies the existence of a (k, d)-tree for k large enough and $d = 2^{s+1} = 2^{k+2-\lfloor \log k \rfloor} \leq \lfloor \frac{2^{k+3}}{k} \rfloor$.

A non-leaf vertex v of a binary tree "distributes" its k-close leaf descendants between its children w' and w''. That is, if $\vec{d}_{w'} = (x'_0, x'_1, \ldots, x'_k)$ and $\vec{d}_{w''} = (x''_0, x''_1, \ldots, x''_k)$, then we have

$$\vec{d}_v = (0, x'_0 + x''_0, x'_1 + x''_1, \ldots, x'_{k-1} + x''_{k-1}) \tag{5.5}$$

We will consider two fundamentally different kinds of way a parent vertex v with leaf-vector $\vec{d}_v = (x_0, x_1, \ldots, x_k)$ can distribute its k-close leaf descendants between its children. Here we assume that d is a large power of 2.

Even Distributions First, a distribution is *even* if $x'_i = x''_i = \frac{x_{i+1}}{2}$ for every $i \in \{0, \ldots, k-1\}$. (We assume for the moment that the coordinates of $\vec{d}_v$ are even.)

Observation 5.21. *Let m be an integer and let $\vec{x} = (x_0, x_1, \ldots, x_k)$ be a (k, d)-vector where x_i is divisible by 2^m, for every $i \geq m$. Then*

$$\vec{x}^{(m)} = \left(\frac{x_m}{2^m}, \frac{x_{m+1}}{2^m}, \ldots, \frac{x_k}{2^m}, \frac{d}{2^m}, \frac{d}{2^{m-1}}, \ldots, \frac{d}{2} \right)$$

is a (k, d)-vector. Moreover, if $\vec{x}^{(m)}$ is constructible then $\vec{x}$ is constructible.

Proof: The first statement is immediate. For the second statement, attach a copy of a $(k, d, \vec{x}^{(m)})$-tree to each leaf of a full binary tree of height m. This gives a $(k, d, \vec{x})$-tree. □

The next corollary is a direct consequence of Observation 5.21 (for $m = \log d - 1$).

Corollary 5.22. *Let $d \leq 2^{k+1}$ be a power of 2. If $(0,\ldots,0,1,2,\ldots,\frac{d}{4},\frac{d}{2})$ is constructible then $(0,\ldots,0,\frac{d}{2})$ is constructible and thus a (k,d)-tree exists.*

Piecewise Distribution Recall that we assumed v to be a non-leaf node of a binary tree with children w' and w'', and that we denote by x_i, x'_i and x''_i the ith coordinate of $\vec{d}_v, \vec{d}_{w'}$ and $\vec{d}_{w''}$, respectively. The corresponding distribution is *piecewise* if there is some threshold index t such that for $i \leq t-1$, $x'_i = x_{i+1}$ and $x''_i = 0$, whereas for $t \leq i \leq k-1$, $x'_i = 0$ and $x''_i = x_{i+1}$. The next observation is the analogon of Observation 5.21.

Observation 5.23. *Let $\vec{x} = (x_0,\ldots,x_k)$ be a (k,d)-vector and let $0 \leq t \leq k$. Then $\vec{x}' = (x_1,\ldots,x_t,0,\ldots,0)$ and $\vec{x}'' = (0,\ldots,0,x_{t+1},\ldots,x_k,0)$. are (k,d)-vectors. Furthermore, the constructibility of $\vec{x}'$ and the constructibility of $\vec{x}''$ imply the constructibility of $\vec{x}$.*

To build our (k,d)-tree we will construct a $(k,d,\vec{y})$-tree for the vector $\vec{y} = (0,\ldots,0,1,2,\ldots,\frac{d}{4},\frac{d}{2})$ (note that by Corollary 5.22 this readily gives a (k,d)-tree) from the root down to the leaves. Note that as d is of the order $2^k/k$ the number of 0s at the beginning is about $\log k$.

The Cut Subroutine The typical subroutine we use to define the leaf-vectors of the children of some node is the one we call *cut at t*. This involves first a piecewise distribution at t, such that Lemma 5.20 is applicable for the first child w', that is $\sum_{i=0}^{t-1} x_{i+1}/2^i \geq 1$. Then the even distribution is applied $m = \log x_{t+1}$ times to the second child w'' (c.f. Observation 5.21) in order to produce a leaf-vector

$$(0,\ldots,0,1,\frac{x_{t+2}}{2^m},\ldots,\frac{x_k}{2^m},0,\frac{d}{2^m},\ldots,\frac{d}{2})$$

(for simplicity we assume for a moment that all x_is are large enough powers of 2). In fact we produce 2^m copies of this vector, but since (by Observation 5.21) its constructibility implies the constructibility of $\vec{x}'' := (0,\ldots,0,x_{t+1},\ldots,x_k,0)$, and thus (by the assumption that Lemma 5.20 is applicable for $\vec{x}' := (x_1,\ldots,x_t,0,\ldots,0)$) also the constructibility of $\vec{x} = (x_0,\ldots,x_k)$, we concentrate on producing one.

Deep Cuts We will need a generalization of the above cut subroutine, which we will call an *l-deep cut at t*. The idea is to choose a very small t, and distribute the possibly very little gain in the weight of $\vec{x}''$ more

evenly on all the coordinates x''_i. In order to be able to cut at a very small t and still being able to use Lemma 5.20 we need to multiply the weight of $\vec{x}'$ by a large number (we will use 2^l) to compensate for the shortness of the cut. We do that by actually distributing the vector $\vec{x}$ into 2^l vectors averaging $\vec{x}$, out of one will (roughly) be the first t entries of $\vec{x}$ shifted by l positions and all the others will (roughly) be the last $k-t$ entries multiplied by $1/(2^l-1)$ and shifted by l positions. This can be achieved by a full binary tree of height l connecting the corresponding trees.

For a more precise description let $\vec{x} = (x_0, \ldots, x_k)$ be a (k,d)-vector and let $l < t$. Moreover, let $\vec{x}' = (x_l, \ldots, x_t, 0, \ldots, 0)$ and let $\vec{x}'' = (0, \ldots, 0, \frac{x_{t+1}}{2^l-1}, \ldots, \frac{x_k}{2^l-1}, 0, \frac{d}{2^{l-1}}, \ldots, \frac{d}{2})$ (we omit floor signs for the sake of convenience). We note that $\vec{x}'$ and $\vec{x}''$ are (k,d)-vectors and that the constructibility of $\vec{x}'$ and $\vec{x}''$ readily imply the constructibility of $\vec{x}$. The latter can be seen by taking a full binary tree of height l, attaching a $(k,d,\vec{x}')$-tree to one leaf, and attaching a $(k,d,\vec{x}'')$-tree to each of the remaining 2^l-1 leaves. Figure 5.3 shows an illustration. Finally, by evaluating Observation 5.21 for $m = \log\left(\frac{x_{t+1}}{2^l-1}\right)$ (for simplicity we assume for a moment that all the $\frac{x_i}{2^l-1}$ are large enough powers of 2) we obtain that the constructibility of $\vec{x}''$ directly implies the constructibility of $\vec{x}''' = (0, \ldots, 0, 1, \frac{x_{t+2}}{2^m(2^l-1)}, \ldots, \frac{x_k}{2^m(2^l-1)}, 0, \frac{d}{2^{m+l-1}} \ldots, \frac{d}{2})$. Putting everything together we get that the constructibility of $\vec{x}'$ and the constructibility of $\vec{x}'''$ yield the constructibility of $\vec{x}$. We will refer to the two vectors $\vec{x}'$ and $\vec{x}'''$ as the outcome of applying to $\vec{x}$ an l-deep cut at t.

In the next subsection we give an indication how, using only the simple cut-subroutine, we can produce a (k,d)-tree with $d \leq \frac{2}{3} \cdot \frac{2^{k+1}}{k}$. Then we use l-deep cuts to push the bound on d to the limit.

5.6.2 Passing to Continuous

The goal of this subsection is to give the motivation behind the formal proof of the next section. Recall that our goal is to obtain a (k,d)-tree for

$$d = \frac{1}{T}\frac{2^{k+1}}{k} \tag{5.6}$$

where T should be as large as possible. By the lower bound in Observation 5.12 we know that we can not achieve $T > e$, our goal is to have $T \geq e - \epsilon$ for every $\epsilon > 0$ and k large enough.

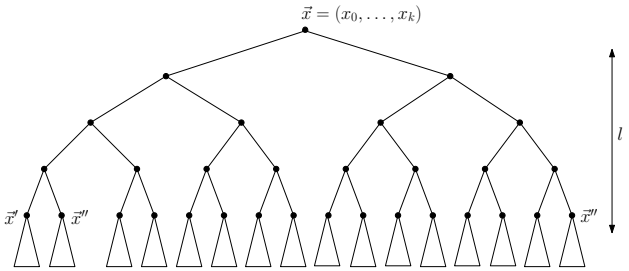


Figure 5.3: Attaching a $(k, d, \vec{x}')$-tree and $2^l - 1$ $(k, d, \vec{x}'')$-trees to the leaves of a full binary tree of height l gives a $(k, d, \vec{x})$-tree.

After fixing a target constant T, it will be helpful to consider the leaf-vectors in a normalized form, which then will enable us to interpret them as continuous functions on the $[0, 1]$ interval. For a given desired value d and a leaf-vector $\vec{x} = (x_0, \ldots, x_k)$ the *normalized vector*, $\vec{y} = \vec{y}(\vec{x})$ is defined as $(y_0, \ldots, y_k)$ with $y_j = \frac{1}{2^j} \frac{2^{k+1}}{d} x_j$, i.e., $\vec{y} = (2^{k+1} \frac{x_0}{d}, 2^k \frac{x_1}{d}, \ldots, 4\frac{x_{k-1}}{d}, 2\frac{x_k}{d})$. We say that a normalized vector $\vec{y}(\vec{x})$ is constructible if $\vec{x}$ is constructible. The next observation follows directly from Lemma 5.20 and (5.6).

Observation 5.24. *Let $\vec{y} = (y_0, y_1, \ldots, y_k)$ be a normalized vector. If $\sum_{i=0}^{k} y_i \geq Tk$ then $\vec{y}$ is constructible.*

By Corollary 5.22 we immediately get the following.

Observation 5.25. *Let d be a power of 2. If the normalized vector $(0, \ldots, 0, 1, \ldots, 1)$ where the last $\log d$ entries are ones is constructible then a (k, d)-tree exists.*

The relationship between the normalized vectors of a parent and its children is also described by an equation similar to (5.5). The normalized vector of the parent is

$$\left(0, \frac{y'_0 + y''_0}{2}, \frac{y'_1 + y''_1}{2}, \ldots, \frac{y'_{k-1} + y''_{k-1}}{2}\right) \tag{5.7}$$

We use this normalizing operation to help us see the leaf-vectors more and more as functions, defined on the $[0, 1]$ interval, possibly in a continuous manner. Indeed, if k is large enough then normalized vectors can be represented as step-functions.

For example, the normalizing operation transforms the leaf-vector $\vec{x} = (0, \ldots, 0, 1, 2, \ldots, 2^s)$ with $2^s = \frac{d}{2}$ into the normalized vector $\vec{y}(\vec{x}) = (0, \ldots, 0, 1, \ldots, 1)$. If $s \approx k - \log k$, then only $o(k)$ entries are 0 and most entries are 1. We want to disregard this small error and consider this normalized vector as the constant 1 function defined on the interval $[0, 1]$. Hence in the continuous setting we can reformulate Observation 5.25 as follows. (The $\star$ denotes that this is not a formal statement.)

Lemma* 5.26. *The constructibility of the constant* 1 *function* f *implies the existence of a* (k, d)*-tree.*

In general we will talk about real functions on the interval $[0, 1]$ and we will routinely ignore the little $o(1)$ sized pieces – by choosing k large enough the normalized vectors are approximated well by the real functions and the $o(1)$ errors will be insignificant.

The following is a reformulation of Observation 5.24 into the continuous setting. The underlying intuition is that for a given normalized vector y we set $f(x) = y_{\lfloor kx \rfloor}$.

Lemma* 5.27. *If* $\int_{x=0}^{1} f(x)\ dx \geq T$ *then* f *is constructible.*

We say that f is *easily constructible* if Lemma* 5.27 applies.

Target T = 1.5

First we take a look at how the cut subroutine of the previous subsection can be pushed to give an easily constructible function with target $T = 1.5$.

In the continuous the subroutine will be called *cut at* v which, given a function f on $[0, 1]$ and a value v between 0 and 1, creates the following two functions (corresponding to the two leaf-vectors in the previous subsection):

$$\begin{aligned} f_{\text{left}}(x) &= \begin{cases} 2f(x) & x \in [0, v) \\ 0 & x \in [v, 1] \end{cases} \\ f_{\text{right}}(x) &= \begin{cases} 2f(x+v) & x \in [0, 1-v) \\ 1 & x \in [1-v, 1] \end{cases} \end{aligned} \tag{5.8}$$

As in the cut subroutine of the previous subsection, where the child w' is expected to use Lemma 5.20, here we want that the function f_{left} is able to use Lemma* 5.27 and hence is easily constructible. We will refer

to f_{left} and f_{right} as the left child function and the right child function, respectively, of f.

Starting with the constant 1 function f and performing a cut at $1-\delta$ with any small constant δ, we obtain that the integral of f_{left} is at least $2-2\delta \geq 1.5$, hence Lemma* 5.27 applies. The other child, f_{right}, on the other hand is significantly improved compared to his parent: its value on the interval $[0,\delta)$ is not 1, but 2. Hence we are able to perform a cut at $1-2\delta$ for f_{right}, obtaining a large enough value for the integral of the left child function. Suppose that we repeatedly cut the right child functions at $v_i = 1-i\delta$ and let f_i denote the ith right child function obtained in this way. By induction we get that, as long as $i\delta < \frac{1}{2}$,

$$f_i(x) = \begin{cases} 2 & x \in [0, i\delta) \\ 1 & x \in [i\delta, 1] \end{cases} \tag{5.9}$$

By our construction the left child function of f_i (obtained by cutting f_i at $v_{i+1} = 1-(i+1)\delta$) is as follows.

$$f_{i,\text{left}}(x) = \begin{cases} 4 & x \in [0, i\delta) \\ 2 & x \in [i\delta, 1-(i+1)\delta) \\ 0 & x \in [1-(i+1)\delta, 1] \end{cases} \tag{5.10}$$

Figure 5.4 depicts f_i and its children. By (5.10) the integral of $f_{i,\text{left}}$ is at least $2-2\delta \geq 1.5$ and therefore the functions $f_{i,\text{left}}$ can always immediately use Lemma* 5.27. Note that by our definition, $f_{i,\text{right}} = f_{i+1}$. (5.9) gives that as soon $i\delta$ reaches $1/2$, the integral of f_i equals 1.5 and thus Lemma* 5.27 applies to it. So the constant 1 function f is constructible, which together with Lemma* 5.26 implies that a (k,d)-tree exists for $d = \frac{1}{T}\frac{2^{k+1}}{k} = \frac{2}{3}\frac{2^{k+1}}{k}$.

We point out that a careful analysis yields that by continuing cutting the right child functions at $v_i = 1-i\delta$ we obtain that as soon as $i\delta$ reaches $\frac{2}{3}$, the integral of the current right child function gets larger than 2. Furthermore, it can be shown that the integral of every left child function occurring in this process is at least $2-8\delta$. Thus, for every ϵ we can also achieve the target $T = 2-\epsilon$. We will not give more details, since below we will strive for a better target anyway.

Target Close to e

We now illustrate how the l-deep-cut subroutine of the previous subsection can be applied to achieve the target $T = e-\epsilon$. In the continuous

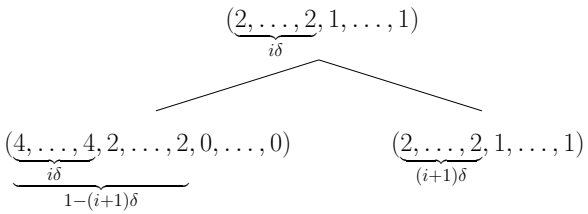


Figure 5.4: An illustration of f_i, its left child function $f_{i,\text{left}}$ and its right child function $f_{i,\text{right}} = f_{i+1}$. For better readability we use the vector-representation ignoring the little $o(1)$ sized pieces.

the subroutine will be called *l-deep cut at v* which, given a function f on $[0,1]$ and a value v between 0 and 1, creates the following two functions (where f, f_{left} and f_{right} correspond to the vectors $\vec{x}$, $\vec{x}'$ and $\vec{x}'''$, respectively, of the previous subsection).

$$f_{\text{left}}(x) = \begin{cases} 2^l f(x) & x \in [0, v) \\ 0 & x \in [v, 1] \end{cases}$$

$$f_{\text{right}}(x) = \begin{cases} \frac{2^l}{2^l-1} f(x+v) & x \in [0, 1-v) \\ 1 & x \in [1-v, 1] \end{cases}$$

We can cut very close to 0 and this is what we will do. On the other hand, the improvement in the value of the right child function is then hardly visible, but is very evenly distributed and is tailored for continuous analysis.

We define a function that approximates well how the right child functions develop if we repeatedly apply an l_i-deep cut at v_i to the current right child function f_i where v_i is very close to 0 (the values of the l_i and the v_i will be determined later). For every t which is of the form $t = \sum_{j=1}^{i} v_j$ for some i and for every $x \in [0,1]$ we define $F(t,x)$ as $f_{i+1}(x)$ (recall that f_{i+1} is the $(i+1)$th right child function we consider). Since the v_i will become infinitesimally small we can regard $F(t,x)$ as a continuous two-variable function, and we will refer to t as the time that has elapsed since we started our process. We will have the initial conditions $F(0,x) = 1$ (we start with the constant 1 function) and $F(t,1) = 1$ (by definition the right child function f of an l-deep cut has the property that $f(1) = 1$).

Let $F(t,x)$ be our current right child function (defined on the $[0,1]$ interval with variable x) for some fixed t. What happens if we cut at some infinitesimally small $\delta \in [0,1]$ with a cut of sufficiently large depth? In order to have the left child function use Lemma* 5.27 immediately the integral of the function on the interval $[0,\delta]$, times 2^l should be at least T: approximating this integral with $\delta F(t,0)$ we should have $l = \log(T/(\delta F(t,0)))$.

After this cut the right child function $f_{\text{right}}(x)$ is 1 on the infinitesimally small interval $[1-\delta,1]$. On the long interval our right child function is defined as

$$f_{\text{right}}(x) = \frac{2^l}{2^l-1} F(t, x+\delta).$$

By construction, we have $f_{\text{right}}(x) = F(t+\delta, x)$. Hence by our choice of l and by the approximation $\frac{2^l}{2^l-1} \approx 1 + \frac{1}{2^l}$ we get that

$$F(t+\delta, x-\delta) \approx \left(1 + \frac{1}{2^l}\right) F(t,x) = (1 + \frac{\delta F(t,0)}{T}) F(t,x).$$

This gives us an equation on the derivative of $F(t,x)$ (in direction $(1,-1)$) which describes how fast the function values tend to change.

$$\frac{F'(t,x)}{F(t,x)} = \frac{1}{F(t,x)} \frac{F(t+\delta, x-\delta) - F(t,x)}{\delta} = \frac{F(t,0)}{T}.$$

Integrating on the segment from $(s-1,1)$ to $(s,0)$ we obtain

$$\int (\ln F)' = \frac{1}{T} \int_{s-1}^{s} F(t,0) dt.$$

The left hand side evaluates to $\ln F(s,0)$ by the initial condition. Assuming that the function $F(t,0)$ is monotonically increasing gives that the right hand side is at least $\frac{F(s-1,0)}{T}$, which implies that

$$F(s,0) \geq e^{\frac{F(s-1,0)}{T}}. \tag{5.11}$$

Our goal is to show that $F(s,0)$ tends to infinity for $T = e - \epsilon$. To this end we need the following observation.

Observation 5.28. *By elementary calculus we get that* $\frac{x}{\ln x} \geq e$ *for every* $x > 1$.

Estimating $F(s,0)$ as a sequence (say on the integers), and setting $a_0 := F(0,0)$ and $a_s := e^{\frac{a_{s-1}}{T}}$, we obtain that $a_s \leq F(s,0)$. The initial condition gives that $a_0 = 1$. Note that by Observation 5.28 and our choice of T we have that the sequence (a_s) is monotonically increasing. If (a_s) does not diverge then for $\epsilon' := \frac{\epsilon}{2} \ln a_1$ we can find an index s such that $a_s - a_{s-1} \leq \epsilon'$. Thus, by definition of a_s and by Observation 5.28 we get that

$$T = \frac{a_{s-1}}{\ln a_s} \geq \frac{a_s - \epsilon'}{\ln a_s} \geq e - \frac{\epsilon}{2},$$

contradicting our choice of T.

Hence (a_s) diverges and therefore the function $F(s,0)$ tends to infinity (for $T = e - \epsilon$), showing that the right child function can also use Lemma* 5.27 after a while.

The above continuous heuristic is the underlying idea of the construction described in Section 5.7. It provides a good approximation to what happens in the discrete case, even though a large number of small errors must be introduced and the handling of all of them is quite technical. Crucially the number of these errors depends only on the given ϵ and hence one can plan for them in advance.

5.7 Formal Proof of Lemma 5.1

To simplify notation we will omit vector arrows throughout this section.

Let us fix the positive integers k, d and l. We will not show the dependence on these parameters in the next definitions to simplify the notation, although d', E, C_r and C_r^* depend on them. We let $d' = d(1 - 1/(2^l - 1))$. For a (k,d)-vector $x = (x_0, \ldots, x_k)$ we define $E(x) = (\lfloor x_1/2 \rfloor, \lfloor x_2/2 \rfloor, \ldots, \lfloor x_k/2 \rfloor, \lfloor d'/2 \rfloor)$. We denote by $E^m(x)$ the vector obtained from x by m applications of the operation E. For $l \leq r \leq k$ we define $C_r(x)$ to be the $(k+1)$-tuple starting with $r + 1 - l$ zero entries followed by $\lfloor x_j/(2^l - 1) \rfloor$ for $j = r+1, \ldots, k$, followed by $\lfloor d'/2^{l-j} \rfloor$ for $j = 0, 1, \ldots, l-1$ and let $C_r^*(x)$ be the $(k+1)$-tuple starting with x_j for $j = l, l+1, \ldots, r$ followed by $k - r + l$ zeros.

Note that for the following lemma to hold we could use d instead of d' in the definition of E and also in most places in the definition of C. The one place where we cannot do this is the entry $\lfloor d'/2^l \rfloor$ of $C_r(x)$ right after $\lfloor x_k/(2^l - 1) \rfloor$. If we used a higher value there, then one of the children of the root of the tree constructed in the proof below would have more than d leaves among its descendants in distance at most k.

We use d' everywhere to be consistent and provide for the monotonicity as used in the subsequent proof of Lemma 5.1. The first part of the next lemma is a reformulation of Observation 5.21 whereas the second part deals with the l-deep-cut subroutine.

Lemma 5.29. *Let k, d and l be positive integers and x a (k,d)-vector.*

(a) $E(x)$ is a (k,d)-vector. If $E(x)$ is constructible, then so is x.
(b) For $l \le r \le k$ both $C_r(x)$ and $C_r^*(x)$ are (k,d)-vectors. If both of these vectors are constructible and $|C_r^*(x)| < d/2^l$, then x is also constructible.

Proof: (a) We have $|E(x)| \le |x|/2 + d'/2 < d$, so $E(x)$ is a (k,d)-vector. If there exists a $(k,d,E(x))$-tree, take two disjoint copies of such a tree and connect them with a new root vertex, whose children are the roots of these trees. The new binary tree so obtained is a (k,d,x)-tree.

(b) The sum of the first $k+1-l$ entries of $C_r(x)$ is at most $|x|/(2^l-1) \le d/(2^l-1)$, the remaining fixed terms sum to less than $d' = d(1-1/(2^l-1))$, so $|C_r(x)| \le d$. We trivially have $|C_r^*(x)| \le |x| \le d$, so both $C_r(x)$ and $C_r^*(x)$ are (k,d)-vectors.

Let T be a $(k,d,C_r(x))$-tree and T^* a $(k,d,C_r^*(x))$-tree. Consider a full binary tree of height l and attach T^* to one of the 2^l leaves of this tree and attach a separate copy of T to all remaining 2^l-1 leaves. This way we obtain a finite binary tree T'. To check condition (i) of the definition (of a (k,d,x)-tree) notice that no leaf of T' is in distance less than l from the root, leaves in distance $l \le j \le r$ are all in T^* and leaves in distance $r < j \le k$ are all in the 2^l-1 copies of T. Condition (ii) we have to check only for vertices in distance $1 \le j \le l$ from the root. There are two types of vertices in distance j. One of them has 2^{l-j} copies of T below it, the other has one less and also T^*. In the first case we can bound the number of leaf descendants in distance at most k by

$$\begin{aligned} & 2^{l-j}\left(\frac{x_{r+1}+\ldots+x_k}{2^l-1}+\frac{d'}{2^l}+\ldots+\frac{d'}{2^{l-j+1}}\right) \\ \le\ & \frac{2^{l-j}}{2^l-1}\cdot d + d'\left(1-\frac{1}{2^j}\right) \\ \le\ & \frac{2^{l-j}}{2^l-1}\cdot d + d\left(\frac{2^l-2}{2^l-1}\right)\left(1-\frac{1}{2^j}\right) \end{aligned}$$

$$
\begin{aligned}
&= \frac{d}{2^l-1}\left(2^{l-j}+2^l\left(1-\frac{1}{2^j}\right)-\underbrace{2\left(1-\frac{1}{2^j}\right)}_{\geq 1}\right)\\
&\leq \frac{d}{2^l-1}(2^l-1)\leq d. \qquad (5.12)
\end{aligned}
$$

We point out that here it is crucial that we use the specific value of $d' < d$.

In the second case, the number of leaf descendants in distance at most k is bounded by

$$
\begin{aligned}
&(2^{l-j}-1)\left(\frac{x_{r+1}+\ldots+x_k}{2^l-1}+\frac{d'}{2^l}+\ldots+\frac{d'}{2^{l-j+1}}\right)+|C_r^*(x)|\\
\leq\ & \frac{2^{l-j}-1}{2^l-1}\cdot d+d'\left(1-\frac{1}{2^j}\right)+\frac{d}{2^l-1}\\
\leq\ & \frac{2^{l-j}}{2^l-1}\cdot d+d'\left(1-\frac{1}{2^j}\right),
\end{aligned}
$$

which by (5.12) is at most d. □

Armed with the last lemma we are ready to prove Lemma 5.1.

Proof of Lemma 5.1

We will show that (k,d)-trees exist for large enough k and with $d = \lfloor 2^{k+1}/(ek) + 100 \cdot 2^{k+1}/k^{3/2} \rfloor$.

We set $l = \lfloor \log k/2 \rfloor$ and $s = 2l$. Thus $2^l \approx \sqrt{k}$. We define the (k,d)-vectors $x^{(t)} = (x_0^{(t)}, \ldots, x_k^{(t)})$ recursively. We start with $x^{(0)} = E^{k-s}(z)$, where z denotes the all zero (k,d)-vector. For $t \geq 0$ we define $x^{(t+1)} = E^{r_t-s-l}(C_{r_t}(x^{(t)}))$, where r_t is the is the smallest index in the range $s+l \leq r_t \leq k$ with $\sum_{j=0}^{r_t} x_j^{(t)}/2^j \geq 2^{-l}$. At this point we may consider the sequence of the (k,d)-vectors $x^{(t)}$ end whenever the weight of one of them falls below 2^{-l} and thus the definition of r_t does not make sense. But we will establish below that this never happens and the sequence is infinite.

Notice first, that we have $x_j^{(t)} = 0$ for all t and $0 \leq j \leq s$, while the entries $x_j^{(t)}$ for $s < j \leq k$ are all obtained from d' by repeated application of dividing by an integer (namely by $2^l - 1$ or by a power of 2) and taking lower integer part. Using the simple observation that

$\lfloor \lfloor a \rfloor / j \rfloor = \lfloor a/j \rfloor$ if a is real and j is a positive integer we can ignore all roundings but the last. This way we can write each of these entries in the form $\left\lfloor \frac{d'}{2^i(2^l-1)^j} \right\rfloor = \left\lfloor \frac{d'}{2^{i+lj}} \left(1 + \frac{1}{2^l-1}\right)^j \right\rfloor$ for some non-negative integers i and j. Using the values $\alpha = 1 + 1/(2^l - 1)$ and $q_t = r_t - s$ (this is the amount of "left shift" between x^t and x^{t+1}) we can give the exponents explicitly.

$$x_j^{(t)} = \left\lfloor \frac{d'}{2^{k+1-j}} \alpha^{c(t,j)} \right\rfloor \tag{5.13}$$

for all $s < j \le k$ and all t, where $c(t, j)$ is the largest integer $0 \le c \le t$ satisfying $\sum_{i=t-c}^{t-1} q_i \le k - j$. We define $c(t, j) = 0$ if $q_{t-1} > k - j$.

The formal inductive proof of this formula is a straight forward calculation. What really happens here (ignoring the rounding) is that each entry enters at the right end of the vector as $d'/2$ and is divided by 2 every time it moves one place to the left (application of E) but when it moves l places to the left through an application of C_{r_t} it is divided by $2^l - 1$ instead of 2^l so it gains a factor of α. The exponent $c(t, j)$ counts how many such factors are accumulated. If the "ancestor" of the entry $x_j^{(t)}$ was first introduced in $x^{(t')}$, then $c(t, j) = t - t'$.

We claim next that $c(t, j)$ and $x_j^{(t)}$ increases monotonously in t for each fixed $s < j \le k$, while q_t decreases monotonously with t. We prove these statements by induction on t. We have $c(0, j) = 0$ for all j, so $c(1, j) \ge c(0, j)$. If $c(t + 1, j) \ge c(t, j)$ for all j, then all entries of $x^{(t+1)}$ dominate the corresponding entries of $x^{(t)}$ by (5.13). If $x_j^{(t+1)} \ge x_j^{(t)}$ for all j, then we have $r_{t+1} \le r_t$ by the definition of these numbers, so we also have $q_{t+1} \le q_t$. Finally, if q_i is decreasing for $i \le t + 1$, then by the definition of $c(i, j)$ we have $c(i + 1, j) \ge c(i, j)$ for $i \le t + 1$.

The monotonicity just established also implies that the weight of $x^{(t)}$ is also increasing, so if the weight of $x^{(0)}$ is at least 2^{-l}, then so is the weight of all the other $x^{(t)}$, and thus the sequence is infinite. The weight of $x^{(0)}$ is

$$\begin{aligned} & \sum_{j=s+1}^{k} \frac{\left\lfloor \frac{d'}{2^{k+1-j}} \right\rfloor}{2^j} \\ > & \sum_{j=s+1}^{k} \frac{\frac{d'}{2^{k+1-j}} - 1}{2^j} \end{aligned}$$

$$\begin{aligned} &> (k-s)\frac{d'}{2^{k+1}} - 2^{-s} \\ &> (k-s)\frac{1-\frac{1}{2^l-1}}{ek} - 2^{-s}, \end{aligned}$$

where the last inequality follows from $d > 2^{k+1}/(ek)$ and the last term tends to e^{-1} as k tends to infinity, so it is larger than 2^{-l} for large enough k.

We have just established that the sequence $x^{(t)}$ of (k,d)-vectors is infinite and coordinate-wise increasing. Since $|x^{(t)}| \leq d$ and it must strictly increase before the sequence stabilizes, the sequence must stabilize in at most d steps. So $x^{(t)} = x = (x_0, \ldots, x_k)$ for $t \geq d$. This implies that q_t also stabilizes with $q_t = q$ for $t \geq d$. (5.13) as applied to $t > d + k$ simplifies to

$$x_j = \left\lfloor \frac{d'}{2^{k+1-j}} \alpha^{\lfloor (k-j)/q \rfloor} \right\rfloor. \tag{5.14}$$

Recall that $q = r_{t_0} - s$, and r_{t_0} is defined as the smallest index in the range $s + l \leq r \leq k$ with $\sum_{j=0}^{r} x_j/2^j \geq 2^{-l}$. Thus we have $q \geq l$. We claim that equality holds. Assume for contradiction that $q > l$ Then by the minimality of r_{t_0} we must have

$$\begin{aligned} 2^{-l} &> \sum_{j=0}^{s+q-1} \frac{x_j}{2^j} \\ &= \sum_{j=s+1}^{s+q-1} \frac{\left\lfloor \frac{d'}{2^{k+1-j}} \alpha^{\lfloor (k-j)/q \rfloor} \right\rfloor}{2^j} \\ &> \sum_{j=s+1}^{s+q-1} \frac{\frac{d'}{2^{k+1-j}} \alpha^{(k-j)/q-1} - 1}{2^j} \\ &> (q-1)\frac{d'}{2^{k+1}} \alpha^{k/q-4} - 2^{-2l}. \end{aligned}$$

In the last inequality we used $s = 2l$ and that $\frac{j}{q} \leq \frac{s+q}{q} = 1 + \frac{s}{q} \leq 1 + \frac{2l}{l} = 3$. This inequality simplifies to

$$2^{-l}(1 + 2^{-l})\alpha^4 \frac{2^{k+1}}{d'} > (q-1)\alpha^{k/q}. \tag{5.15}$$

Simple calculus gives that the right hand side takes its minimum for $q \geq 2$ between $c - 1$ and $c - 2$ for $c = k \ln \alpha$ and this minimum is more

than $(c-3)e$. Using $\alpha = 1 + 1/(2^l - 1) > e^{2^{-l}}$ we have $c \geq k/2^l$. So (5.15) yields

$$2^{-l}(1+2^{-l})\alpha^4 \frac{2^{k+1}}{d'} > \frac{ke}{2^l} - 3e.$$

Thus,

$$(1+2^{-l})\alpha^4 \frac{2^{k+1}}{d'} - ke + 3e2^l > 0.$$

Substituting our choice for d, d', α and l (as functions of k) and assuming that k is large enough we get that the left hand side is at most

$$\begin{aligned}
& \left(1+\frac{2}{\sqrt{k}}\right)\left(1+\frac{2}{\sqrt{k}-2}\right)^4 \frac{2^{k+1}}{d\left(1-\frac{2}{\sqrt{k}-2}\right)} - ek + 3e\sqrt{k} \\
\leq\ & \left(1+\frac{2}{\sqrt{k}}\right)\left(1+\frac{4}{\sqrt{k}}\right)^4 \frac{2^{k+1}}{d\left(1-\frac{4}{\sqrt{k}}\right)} - ek + 3e\sqrt{k} \\
\leq\ & \left(1+\frac{19}{\sqrt{k}}\right) \frac{ek}{\left(1-\frac{4}{\sqrt{k}}\right)\left(1+\frac{99e}{\sqrt{k}}\right)} - ek + 3e\sqrt{k} \\
\leq\ & \left(1-\frac{70}{\sqrt{k}}\right) ek - ek + 3e\sqrt{k} = -70e\sqrt{k} + 3e\sqrt{k} < 0,
\end{aligned}$$

which contradicts our assumption. Hence $q = l$ as claimed.

We finish the proof of the theorem by establishing that the (k,d)-vectors $x^{(t)}$ are constructible. We prove this statement by downward induction for t. We start with $t = d$, where $x^{(d)} = x$. Using (5.14) and the fact that $q = l$ we get that for large k,

$$\begin{aligned}
w(x) &\geq \frac{x_{s+1}}{2^{s+1}} \geq \frac{d'}{2^{k+1}}\alpha^{\frac{k-s-1}{l}-1} - 1 \\
&\geq \frac{d'}{2^{k+1}}\alpha^{\frac{k}{2l}} - 1 \\
&\geq \frac{d/2}{2^{k+1}}\left(1+\frac{1}{\sqrt{k}}\right)^{\frac{k}{\log k}} - 1 \\
&\geq \frac{1}{2ek} e^{\frac{k}{2\sqrt{k}\log k}} - 1 > 1.
\end{aligned}$$

The first inequality on the last line holds due to the well-known fact that $1 + \frac{1}{y} \geq e^{\frac{1}{2y}}$ for y large enough. So by Lemma 5.20 x is constructible.

Now assume that $x^{(t+1)}$ is constructible. Recall that we have $x^{(t+1)} = E^{r_t-s-l}(C_{r_t}(x^{(t)}))$, so by (the repeated use of) Lemma 5.29 part (a) $C_{r_t}(x^{(t)})$ is constructible. By part (b) of the same lemma $x^{(t)}$ is also constructible (and thus the inductive step is complete) if we can (i) show that $C^*_{r_t}(x^{(t)})$ is constructible and (ii) establish that $|C^*_{r_t}(x^{(t)})| \leq d/2^l$. For (i) we use the definition of r_t: $\sum_{j=0}^{r_t} x_j^{(t)}/2^j \geq 2^{-l}$. But the weight of $C^*_{r_t}(x^{(t)})$ is $\sum_{j=l}^{r_t} x_j^{(t)}/2^{j-l}$, so the contribution of each term with $j \geq l$ is multiplied by 2^l, while the missing terms $j < l$ contributed zero anyway as $l < s$. This shows that the weight of $C^*_{r_t}$ is at least 1 and therefore Lemma 5.20 proves (i). For (ii) we use monotonicity to see $r_t \leq r_0$ and (5.13) to see that $r_0 \leq 5k/2^l$ for large enough k. Then we use monotonicity again to see $|C^*_{r_t}(x^{(t)})| = \sum_{j=s+1}^{r_t} x_j^{(t)} \leq \sum_{j=s+1}^{r_t} x_j$. Finally, by (5.14) and the fact that $q = l$ we get that for large k,

$$\begin{aligned}
|C^*_{r_t}(x^{(t)})| &\leq d' \sum_{j=s+1}^{r_t} \frac{1}{2^{k+1-j}} \cdot \alpha^{\frac{k-j}{l}} \\
&\leq d \sum_{j=s+1}^{r_t} \frac{\alpha^{k-j}}{2^{k-j}} \\
&\leq d \sum_{j=s+1}^{r_t} \left(\frac{\alpha}{2}\right)^{k-j} \\
&\leq d \left(\frac{3}{4}\right)^{k-r_t} \cdot 4 \quad \left(\text{since } \frac{\alpha}{2} \leq \frac{3}{4}\right) \\
&\leq d \left(\frac{3}{4}\right)^{\frac{k}{2}} \cdot 4 \quad \left(\text{since } r_t \leq \frac{5k}{2^l} \leq \frac{k}{2}\right) \\
&< \frac{d}{\sqrt{k}} \leq \frac{d}{2^l}.
\end{aligned}$$

This finishes the proof of (ii) and hence the inductive proof that $x^{(t)}$ is constructible for every t.

As $x^{(0)} = E^{k-s}(z)$ is constructible Lemma 5.29 (a) implies that z is also constructible, so there exists (k, d, z)-tree T. As z is the all zero vector, T must also be a (k, d)-tree. □

5.8 Outlook and Open Problems

5.8.1 MU(1) Formulas and Trees

We first summarize some results of the previous sections. Let T be a binary tree where every vertex different from the root is labeled with a literal such that the two children of any non-leaf vertex are labeled with complementary literals. A CNF formula $\mathcal{G}$ is called a *T-formula* if $\mathcal{G}$ can be obtained by associating with every leaf v of T a clause C_v that is the disjunction of *some* literals along the path from v to the root, and taking the conjunction of all those C_v. By Corollary 5.16, a CNF formula $\mathcal{G}$ is in MU(1) if and only if $\mathcal{G}$ is a T-formula for some binary tree T where all literals associated to vertices of T *do* appear in $\mathcal{G}$.

In our construction for the upper bound on $f(k)$ (c.f. Theorem 5.8 on page 77) we restrict to T-formulas of the form $\mathcal{F}(T)$, where T is some (k,d)-tree and with every leaf v of T we associate the clause which is the disjunction of the first k labels on the path from v to the root. By Lemma 5.10, $\mathcal{F}(T)$ is an unsatisfiable (k,d)-CNF formula. We call a (k,d)-tree T *minimal* if none of its subtrees is a (k,d)-tree. Note that every non-root vertex w of a minimal (k,d)-tree T has a leaf descendant at distance at most $k-1$ (otherwise the subtree rooted at w is a (k,d)-tree, contradicting the minimality of T); thus, by construction, every literal of T does appear in $\mathcal{F}(T)$, and therefore, due to the above characterization of MU(1) formulas, $\mathcal{F}(T)$ belongs to MU(1). Hence,

$$f(k) \leq f_1(k) \leq f_{\mathsf{tree}}(k), \tag{5.16}$$

where $f_1(k)$ denotes the largest integer s such that no (k,s)-CNF formula is in MU(1), and $f_{\mathsf{tree}}(k)$ denotes the largest integer d such that no (k,d)-tree exists. Finally, Observation 5.12 gives that the values $f(k)$, $f_1(k)$ and $f_{\mathsf{tree}}(k)$ are equal up to lower order terms:

$$f_{\mathsf{tree}}(k) = \left(\frac{2}{e} + O(1/\sqrt{k})\right)\frac{2^k}{k} = (1+o(1))f(k). \tag{5.17}$$

5.8.2 On the Size of Unsatisfiable Formulas

By the *size* of a rooted tree we mean the number of its leaves and by the *size* of a CNF formula we mean the number of its clauses. With this notation the size of a minimal (k,d)-tree T and the size of the corresponding (k,d)-CNF formula $\mathcal{F}(T)$ in MU(1) are the same. If a (k,d)-tree T is not minimal then by picking a minimal subtree T' of T

and considering $\mathcal{F}(T')$ we can readily find a *corresponding* (k,d)-CNF formula in MU(1) whose size is at most the size of T.

By a *minimum* (k,d)-tree we denote a (k,d)-tree which has the smallest size of all (k,d)-trees. A *minimum* (k,d)-CNF formula is then defined analogously. In the proof of Lemma 5.1 we constructed (k,d)-trees for $d \approx \frac{2^{k+1}}{ek}$. Their size and therefore the size of the corresponding (k,d)-CNF formula in MU(1) is at most 2^h, where h is the *height* of the tree: the largest root-leaf distance. In fact, the size of the trees we constructed is very close to this upper bound. Therefore it makes sense to take a closer look at the height.

Recall that we associated with a vertex v of a (k,d)-tree the (k,d)-vector $(x_0,\ldots,x_k)$, where x_j is the number of leaf descendants of v of distance j from v. A minimum (k,d)-tree has no two vertices along the same branch with identical vectors, so the height of a minimum (k,d)-tree is limited by the number of (k,d)-vectors, less than $(d+1)^{k+1}$. For $d = f(k)+1$ this is $2^{\Theta(k^2)}$. The same bound for minimum (k,d)-CNF formulas in MU(1) is implicit in [54]. There is numerical evidence that the size of the minimum $(k, f_{\mathsf{tree}}(k)+1)$-tree and the minimum $(k, f_1(k)+1)$-CNF formula in MU(1) might indeed be doubly exponential in k (consider the size of the minimum $(7,18)$-tree and the minimum $(7,18)$-CNF formula in MU(1) mentioned below).

A closer analysis of the proof of Lemma 5.1 shows that the height of the (k,d)-tree constructed in it is at most kd. While this is better than the general upper bound above it still allows for trees with sizes that are doubly exponential in k.

This height can, however, be substantially decreased if we allow the error term in d to slightly grow. If we allow $d = (1+\epsilon)\frac{2^{k+1}}{ek}$ for a fixed $\epsilon > 0$, then a more careful analysis shows that the height of the tree created becomes $O(k)$. This bounds the size of the tree and the corresponding formula by a *polynomial in* d.

Let us define $f_1(k,d)$ for $d > f_1(k)$ to be the size of the minimum (k,d)-CNF formula in MU(1) and let $f_{\mathsf{tree}}(k,d)$ stand for the size of the minimum (k,d)-tree, assuming $d > f_{\mathsf{tree}}(k)$. While $f_1(k, f_1(k)+1)$ and similarly $f_{\mathsf{tree}}(k, f_{\mathsf{tree}}(k)+1)$ are probably doubly exponential in k, for slightly larger values of d, $f_1(k,d)$ and $f_{\mathsf{tree}}(k,d)$ are polynomial in d (and thus simply exponential in k).

5.8.3 Extremal Values and Algorithms

Finally, we mention the question whether $f_1(k) = f_{\mathsf{tree}}(k)$ for all k. In other words, we ask whether we lose by selecting the k vertices farthest from the root when making an MU(1) k-CNF formula from a binary tree. As mentioned above, $f(k) = f_1(k)$ is also open, but $f_1(k) = f_{\mathsf{tree}}(k)$ seems to be a simpler question as both functions are computable. Computing their values up to $k = 8$ we found these values agreed. To gain more insight we computed the corresponding size functions too and found that $f_1(k, d) = f_{\mathsf{tree}}(k, d)$ for $k \leq 7$ and all $d > f_1(k)$ with just a single exception. We have $f_1(7) = 17$ and $f_1(7, 18) = 10,197,246,480,846$, while $f_{\mathsf{tree}}(7, 18) = 10,262,519,933,858$. Does this indicate that all other equalities are coincidences and f_1 and f_{tree} will eventually diverge?

A related algorithmic question is whether the somewhat simpler structure of (k, d)-trees can be used to find an algorithm computing $f_{\mathsf{tree}}(k)$ substantially faster than the algorithm of Hoory and Szeider [54] for computing $f_1(k)$. Such an algorithm would give useful estimates for $f_1(k)$ and also $f(k)$. At present we use a similar (and similarly slow) algorithm for either function.

Bibliography

[1] R. Aharoni and N. Linial. Minimal non-two-colorable hypergraphs and minimal unsatisfiable formulas. *J. Combin. Theory Ser. A* **43**, (1986), 196–204.

[2] N. Alon and V. Asodi. Sparse universal graphs. *Journal of Computational and Applied Mathematics.* **142**, (2002), 1–11.

[3] N. Alon and M. Capalbo. Sparse universal graphs for bounded degree graphs. *Random Structures and Algorithms* **31**, (2007), 123–133.

[4] N. Alon and M. Capalbo. Optimal universal graphs with deterministic embedding. *Proc. of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)* (2008), 373–378.

[5] N. Alon, M. Capalbo, Y. Kohayakawa, V. Rödl, A. Ruciński and E. Szemerédi. Universality and tolerance. *Proceedings of the 41st IEEE FOCS* (2000), 14–21.

[6] N. Alon, M. Capalbo, Y. Kohayakawa, V. Rödl, A. Ruciński and E. Szemerédi. Near-optimum universal graphs for graphs with bounded degrees. *RANDOM-APPROX* (2001), 170–180.

[7] N. Alon, M. Krivelevich, J. Spencer and T. Szabó. Discrepancy Games. *Electronic Journal of Combinatorics*, **12**(1), (2005), R51.

[8] N. Alon. A parallel algorithmic version of the Local Lemma. *Random Structures and Algorithms* **2**(4), (1991), 367–378

[9] N. Alon and J.H. Spencer. The Probabilistic Method. *J. John Wiley & Sons* (2002).

[10] L. Babai, F. R. K. Chung, P. Erdős, R. L. Graham and J. Spencer. On graphs which contain all sparse graphs. *Ann. Discrete Math.* **12**, (1982), 21–26.

[11] J. Balogh, R. Martin and A. Pluhár. The diameter game. *Random Structures and Algorithms* **35**, (2009) 369–389.

[12] J. Beck. Van der Waerden and Ramsey Type Games. *Combinatorica* **1**, (1981), 103–116.

[13] J. Beck. Remarks on positional games. *Acta Math. Acad. Sci. Hungar.* **40**, (1982), 65–71.

[14] J. Beck. On size Ramsey number of paths, trees and cycles I. *J. Graph Theory* **7**, (1983), 115–130.

[15] J. Beck. Random graphs and positional games on the complete graph. *Annals of Discrete Math.* **28**, (1985), 7–13.

[16] J. Beck. An algorithmic approach to the Lovász local lemma I. *Random Structures and Algorithms* **2** (1991), 343–365.

[17] J. Beck. Achievement games and the probabilistic method. In *Combinatorics, Paul Erdős is Eighty, Keszthely* Bolyai Soc. Math Studies Budapest, (1993), Vol 1, 51–78.

[18] J. Beck. Deterministic graph games and a probabilistic intuition. *Combin., Prob. Computing* **3**, (1994), 13–26.

[19] J. Beck. Foundations of Positional Games. *Random Structures and Algorithms* **9**, (1996), 15–47.

[20] J. Beck. Ramsey Games. *Discrete Math.* **249**, (2002), 3–30.

[21] J. Beck. Combinatorial Games: Tic-Tac-Toe Theory. Cambridge University Press, 2008.

[22] M. Bednarska and T. Łuczak. Biased positional games and the phase transition. *Random Structures and Algorithms* **18**, (2001), 141–152.

[23] M. Bednarska and O. Pikhurko. Biased positional games on matroids. *European J. Combin.* **26** (2005), 271–285.

[24] P. Berman, M. Karpinski, and A. D. Scott. Approximation hardness and satisfiability of bounded occurrence instances of SAT. *Electronic Colloquium on Computational Complexity (ECCC)*, **10** (022), 2003.

[25] S. N. Bhatt, F. Chung, F. T. Leighton and A. Rosenberg. Universal graphs for bounded-degree trees and planar graphs. *SIAM J. Disc. Math.* **2**, (1989), 145–155.

[26] S. N. Bhatt and C. E. Leiserson. How to assemble tree machines. *Advances in Computing Research*, F. Preparata, ed., 1984.

[27] B. Bollobás. Random Graphs. Cambridge University Press, 2001.

[28] M. Capalbo. A small universal graph for bounded-degree planar graphs. *Proc. 10th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)* (1999), 150–154.

[29] M. Capalbo and S. R. Kosaraju. Small universal graphs. *Proc. 31st Ann. ACM Symp. on Theory of Computing (STOC)* (1999), 741–749.

[30] F. R. K. Chung and R. L. Graham. On graphs which contain all small trees. *J. Combin. Theory Ser. B.* **24**, (1978), 14–23.

[31] F. R. K. Chung and R. L. Graham. On universal graphs. *Ann. New York Acad. Sci.* **319**, (1979), 136–140.

[32] F. R. K. Chung and R. L. Graham. On universal graphs for spanning trees. *Proc. London Math. Soc.* **27**, (1983), 203–211.

[33] F. R. K. Chung, A. L. Rosenberg and L. Snyder. Perfect storage representations for families of data structures. *SIAM J. Alg. Disc. Methods.* **4**, (1983), 548–565.

[34] V. Chvátal and P. Erdős. Biased positional games. *Annals of Discrete Math.* **2**, (1978), 221–228.

[35] F. Chung and R. Graham. *Erdős on graphs. His Legacy of Unsolved Problems.* A K Peters, Ltd., Wellesley, MA, (1998).

[36] V. Chvátal, V. Rödl, E. Szemerédi, and W.T. Trotter. The Ramsey number of a graph with bounded maximum degree. *J. Combin. Theory Ser. B* **34**, (1983), 239–243.

[37] S.A. Cook. The complexity of theorem-proving procedures. *Proc. 3rd Ann. ACM Symp. on Theory of Computing (STOC)* (1971), 151–158.

[38] G. Davydov, I. Davydova, and H. Kleine Büning. An efficient algorithm for the minimal unsatisfiability problem for a subclass of CNF. *Artif. Intell.* **23**, (1998), 229–245.

[39] O. Dubois. On the r, s-SAT satisfiability problem and a conjecture of Tovey. *Discrete Appl. Math.* **26** (1990), 51–60.

[40] O. N. Feldheim and M. Krivelevich. Winning fast in sparse graph construction games. *Combinatorics, Probability and Computing* **17**(6), (2008) 781–791.

[41] P. Erdős, R.J. Faudree, C.C. Rousseau, and R.H. Schelp. The size Ramsey number. *Period. Math. Hungar.* **9**, (1978), 145–161.

[42] P. Erdős and J.L. Selfridge. On a combinatorial game. *J. Combinatorial Theory Ser. A* **14**, (1973), 298–301.

[43] P. Erdős and J.Spencer. Lopsided Lovász local lemma and Latin transversals. *Discrete Appl. Math.* **30**, (1991), 151–154.

[44] U. Feige. A threshold of $\ln n$ for approximating set cover. *J. ACM* **45**(4), (1998), 634–652.

[45] J. Friedman and N. Pippenger. Expanding graphs contain all small trees. *Combinatorica* **7**, (1987), 71–76.

[46] H. Gebauer. Disproof of the Neighborhood Conjecture with Implications to SAT. *Proc. 17th Annual European Symposium on Algorithms (ESA)* (2009), LNCS 5757, 764–775.

[47] H. Gebauer. Maker Can Construct a Sparse Graph on a Small Board. *Eprint, arXiv:1011.2178v2* (2010)

[48] H. Gebauer. R. A. Moser, D. Scheder and E. Welzl. The Lovász Local Lemma and Satisfiability. *Efficient Algorithms*, (2009), 30–54.

[49] H. Gebauer and T. Szabó. Asymptotic random graph intuition for the biased connectivity game. *Random Structures and Algorithms* **35**, (2009), 431–443.

[50] H. Gebauer, T. Szabó and G. Tardos. The Local Lemma is Tight for SAT. *Proc. 22nd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)* (2011), 664–674.

[51] P.E. Haxell, Y. Kohayakawa and T. Łuczak. The induced size-Ramsey number of cycles. *Combin. Prob. Comp.* **4** (1995), 217–239.

[52] D. Hefetz, M. Krivelevich, M. Stojaković and T. Szabó. Planarity, colorability, and minor games. *SIAM J. Discrete Math.* **22**, (2008), no. 1, 194–212.

[53] S. Hoory and S. Szeider. A note on unsatisfiable k-CNF formulas with few occurrences per variable. *SIAM J. Discrete Math* **20** (2), (2006), 523–528.

[54] S. Hoory and S. Szeider. Computing unsatisfiable k-SAT instances with few occurrences per variable. *Theoretical Computer Science* **337**(1–3) (2005), 347–359.

[55] H. Kleine Büning and X. Zhao. On the structure of some classes of minimal unsatisfiable formulas. *Discr. Appl. Math.* **130**(2), (2003), 185–207.

[56] Y. Kohayakawa, V. Rödl, M. Schacht and E. Szemerédi. Sparse partition universal graphs for graphs of bounded degree. *Advances in Mathematics* **226**(6), (2011), 5041–5065.

[57] L. G. Kraft. A device for quantizing, grouping, coding amplitude modulated pulses. *M.S. thesis, Electrical Engineering Department, MIT* (1949).

[58] J. Kratochvíl, P. Savický and Z. Tuza. One more occurrence of variables makes satisfiability jump from trivial to NP-complete. *SIAM J. Comput.* **22** (1), (1993), 203–210.

[59] M. Krivelevich and T. Szabó. Biased positional games and small hypergraphs with large covers. *Electronic Journal of Combinatorics*, **15(1)** (2008), R70.

[60] O. Kullmann. An application of matroid theory to the SAT problem. *Fifteenth Annual IEEE Conference on Computational Complexity* (2000), 116-124.

[61] A. Lehman. A solution of the Shannon switching game. *J. Soc. Indust. Appl. Math* **12** (1964), 687–725.

[62] R. A. Moser, G. Tardos. A constructive proof of the general lovász local lemma. *J. ACM* **57**(2), (2010).

[63] R. A. Moser. A constructive proof of the Lovász Local Lemma. *Eprint, arXiv:0810.4812v2* (2008).

[64] C.H. Papadimitriou and M. Yannakakis. Optimization, approximation, and complexity classes. *J. Comput. System Sci.* **43** (3), (1991), 425–440.

[65] A. Pekeč. Ramsey Games. A winning strategy for the Ramsey graph game. *Combin. Probab. Comput.* **5**, (1996), 267–276.

[66] J. Radhakrishnan, A. Srinivasan. Improved bounds and algorithms for hypergraph 2-coloring. *Random Structures Algorithms* **16** (3), (2000), 4–32.

[67] V. Rödl. A note on universal graphs. *Ars Combin.* **11** (1981), 225–229.

[68] V. Rödl and E. Szemerédi. On size Ramsey numbers of graphs with bounded degree. *Combinatorica* **20** (2000), 257–262.

[69] S. Roman. Coding and Information Theory. *Springer, New York* (1992).

[70] P. Savický and J. Sgall. DNF tautologies with a limited number of occurrences of every variable. *Theoret. Comput. Sci.* **238** (1–2), (2000), 495–498.

[71] D. Scheder. Unsatisfiable Linear CNF Formulas Are Large and Complex. *27th International Symposium on Theoretical Aspects of Computer Science (STACS)* (2010), 621–631.

[72] A. Srinivasan. Improved algorithmic versions of the Lovász Local Lemma. *Proc. 19th Ann. ACM-SIAM Symposium on Discrete Algorithms (SODA)* (2008), 611–620.

[73] T. Szabó and M. Stojaković. Positional games on random graphs. *Random Structures and Algorithms* **26**, (2005), no. 1–2, 204–223.

[74] S. Szeider. Homomorphisms of conjunctive normal forms. *Discr. Appl. Math.* **130**(2), (2003), 351–365.

[75] L. A. Székely. On two concepts of discrepancy in a class of combinatorial games. *Infinite and Finite Sets, Colloquia Mathematica Societatis János Bolyai*, **37**, (1984) 679–683.

[76] C.A. Tovey. A simplified NP-complete satisfiability problem. *Discr. Appl. Math.* **8** (1), (1984), 85–89.

Curriculum Vitae

Heidi Gebauer

born August 21, 1981 in Zürich, Switzerland

1994 - 2001	Highschool Kantonsschule Wiedikon, Zürich
2001 - 2007	Studies of Computer Science ETH Zürich, Switzerland Major: Theoretical Computer Science Degree: Master of Science
since April 2007	Ph.D. student at ETH Zürich, Switzerland