Threshold Phenomena in Random Constraint Satisfaction Problems

by

Harold Scott Connamacher

A thesis submitted in conformity with the requirements
for the degree of Doctor of Philosophy
Graduate Department of Computer Science
University of Toronto

Abstract

Threshold Phenomena in Random Constraint Satisfaction Problems

Harold Scott Connamacher

Doctor of Philosophy

Graduate Department of Computer Science

University of Toronto

2008

Despite much work over the previous decade, the Satisfiability Threshold Conjecture remains open. Random k-SAT, for constant $k \geq 3$, is just one family of a large number of constraint satisfaction problems that are conjectured to have exact satisfiability thresholds, but for which the existence and location of these thresholds has yet to be proven. Of those problems for which we are able to prove an exact satisfiability threshold, each seems to be fundamentally different than random 3-SAT.

This thesis defines a new family of constraint satisfaction problems with constant size constraints and domains and which contains problems that are NP-complete and a.s. have exponential resolution complexity. All four of these properties hold for k-SAT, $k \geq 3$, and the exact satisfiability threshold is not known for any constraint satisfaction problem that has all of these properties. For each problem in the family defined in this thesis, we determine a value c such that c is an exact satisfiability threshold if a certain multi-variable function has a unique maximum at a given point in a bounded domain. We also give numerical evidence that this latter condition holds.

In addition to studying the satisfiability threshold, this thesis finds exact thresholds for the efficient behavior of DPLL using the unit clause heuristic and a variation of the generalized unit clause heuristic, and this thesis proves an analog of a conjecture on the satisfiability of $(2 + p)$-SAT.

Besides having similar properties as k-SAT, this new family of constraint satisfaction problems is interesting to study in its own right because it generalizes the XOR-SAT problem and it has close ties to quasigroups.

Acknowledgements

First and foremost, I wish to acknowledge and thank my supervisor, Michael Molloy, for his help and support. The many hours he spent guiding me through the subject, listening to my ideas and providing constructive advice, and helping me revise my papers, talks, and this thesis are invaluable. Through it all, he taught me a great deal about mathematics, computer science, and how to do research. I am very fortunate to have been his student.

I would like to thank Derek Corneil for being an informal mentor to me during my time as a graduate student. I really appreciate the many times I sought his advice and was warmly received.

I owe a debt of gratitude to the other members of my supervisory committee: Fahiem Bacchus, Stephen Cook, and Toniann Pitassi. Their insightful comments and recommendations helped me tremendously in preparing this thesis.

I would like to thank Nicholas Wormald for his very careful reading of the thesis and excellent suggestions for improving it.

I wish to acknowledge many people who helped me throughout my work on this thesis, from explaining details of various papers and mathematical techniques to providing suggestions for ways to attack the various problems: Christina Christara, Abraham Flaxman, Travis Gagie, Hamed Hatami, Stephanie Horn, and Glenn Lilly. In particular, Hamed Hatami was especially valuable in helping me prove some of the lemmas in this thesis and simplify the matrices of Section 4.5.

Finally, I wish to thank my friends and fellow graduate students from the graph theory group at the University of Toronto: Anna Bretscher, Babak Farzad, Richard Krueger, Lap Chi Lau, Yiannis Papoutsakis, Natasa Przulj, Mohammad Salavatipour, and Frank Van Bussel. The shared work we did in reading papers and the discussions we had made my life as a graduate student more enjoyable.

To Celeste, Nadine, and Charlotte.

Contents

List of Tables

List of Figures

Chapter 1

Introduction and Definitions

The study of threshold phenomena in random constraint satisfaction problems has grown out of three different disciplines: mathematics, statistical physics, and computer science. In mathematics, Erdös and Rényi began the study of random graphs in the 1950's. A graph is a finite set of vertices and a set of edges, each edge connecting a pair of vertices. Erdös and Rényi explored two different models of random graphs. The first is to fix integers n and m and choose a graph uniformly at random from all possible graphs with n vertices and m edges. The second is to fix an integer n and a probability p and each of the $\binom{n}{2}$ possible edges is included in the graph with probability p. In [ER60] they considered the model where the number of edges is fixed, and they studied many properties such as being connected (studied in [ER59]), containing a cycle, complete subgraph, or tree of fixed size, being planar, or containing a giant component. For each of these properties, they discovered that if we consider the probability that a graph has that property as n tends to infinity, there is a definite threshold $m_0 = m_0(n)$ in the edge density such that for a random graph with n vertices and $m = m(n)$ edges, if $m \ll m_0$, the graph almost surely does not have the property, and for $m \gg m_0$, the graph almost surely has the property. By an equivalence theorem for random graphs, these results also hold in the random graph model where each edge exists with probability p. In this model, there is a threshold $p_0 = p_0(n)$ in the edge probability distinguishing the graphs that almost surely have the property from those that do not. Formally, a sequence of events $\mathcal{E}_n$ holds *almost surely* (a.s.) if $\lim_{n\to\infty} \mathbf{Pr}(\mathcal{E}_n) = 1$. Likewise, the sequence holds *with uniform positive probability* (w.u.p.p.) if $\liminf_{n\to\infty} \mathbf{Pr}(\mathcal{E}_n) = \alpha > 0$.

Physicists have long been interested in understanding physical transitions in natural processes such as when materials cool and crystallize. In statistical physics, models such as the spin-glass like model (see [MPV87]) are used to study these physical transitions and discover which conditions lead to threshold behavior in the transitions. Natural systems are generally uniform in the sense that particles interact equally with all other particles in their neighborhood, and the strength of the interaction is determined by the distance separating the particles. However, if we generalize the model to allow for sparse, random interactions, then we can represent the same random problems considered by the mathematicians and computer scientists.

In computer science, it is well known that a large number of important problems are NP-complete/hard, and so they appear to be very difficult to solve in the worst case. However, less is known about the average case difficulty of these problems. Some researchers (see, for example, [CKT91, SML96]) proposed studying uniformly random problem instances as a way of determining average-case hardness. It was discovered in these empirical studies that many random problems demonstrate threshold behavior in some parameter of the problem. Namely, as the parameter value crosses some threshold, the problems go from being almost surely solvable to being almost surely unsolvable. It was also discovered that in many cases the random instances that require the most time to solve are those that are drawn from near this threshold.

Of the open questions in the study of threshold phenomena of random problems, the problem that attracts the most attention from all three communities is the *Satisfiability Threshold Conjecture* for k-SAT. In k-SAT, we are given a formula that is a set of m clauses over n variables. Each clause is a set of k literals where a literal is either a positive or negative occurrence of a variable, and the problem is to assign the values *true* or *false* to the variables so that each clause has at least one true literal. If there exists such an assignment to the variables, the formula is *satisfiable*, and if there is not, the formula is *unsatisfiable*. The k-SAT decision problem is NP-complete for $k \geq 3$. The random formula model is to fix n and m and to choose a uniformly random k-SAT formula with n variables and m clauses. The interesting region, as far as satisfiability is concerned, is when m is a linear function of n.

Conjecture 1.1 (Satisfiability Threshold Conjecture [CR92]) *There exists a value c_k^* such that a uniformly random k-SAT formula on n variables and cn clauses, with n tending to infinity, is almost surely unsatisfiable if $c > c_k^*$ and almost surely satisfiable if $c < c_k^*$.*

Neither the existence nor the location of c_k^* is known for any $k \geq 3$. Much of the work on the conjecture over the past decade has been in improving the upper and lower bounds for the thresholds, if they exist.

Research on the satisfiability threshold has extended to generalizations of k-SAT such as the Schaefer [Sch78] generalizations: 1-in-k SAT [ACIM01] (only one true literal per clause), NAE-SAT [ACIM01, AM06] (at least one true and one false literal per clause), and XOR-SAT [DM02, MRTZ03] (each clause is an exclusive-or rather than a disjunction), and to more general random constraint satisfaction problems.

In a constraint satisfaction problem (CSP), we can allow variables to take on values from a domain of size larger than 2, and we have more freedom as to the types of constraints to apply to each clause. There have been various models proposed and studied for random constraint satisfaction problems [AMK$^+$01, CD03, Mit02, Mol02, Mol]. For each such model and generalization of the satisfiability conjecture, the primary focus of the research has been to determine the satisfiability threshold for the generalized model. In addition, there has been a large body of experimental studies [GMP$^+$01] to find the approximate location of the satisfiability threshold and to study the difficulty of solving random instances of the CSP models. Two reasons these generalizations are studied are that some of these generalizations are interesting problems in their own right, and some of these generalizations have led to a greater understanding of random 3-SAT [AM06]. In addition, researchers have studied models where the domain size grows with n [DFM03, FM03, Smi01, XL00, XL06] or where the constraint size grows with n [FW01, Fla03].

The exact satisfiability threshold is known for very few SAT-like problems, and we are not yet close to answering the Satisfiability Threshold Conjecture. In fact, each problem for which we know the exact satisfiability threshold appears to be fundamentally different from k-SAT, $k \geq 3$. For example, we know the exact satisfiability threshold for 2-SAT [CR92, FdlV92, Goe96] and 3-XOR-SAT [DM02, MRTZ03]), but neither of these problems is NP-complete. Exact thresholds are known for some models where the domain size grows with n [XL00, XL06] or where the constraint size grows with n [FW01, Fla03], but the satisfiability threshold for these models occurs when the number of clauses is superlinear in the number of variables, and the structure of a random constraint satisfaction problem with a superlinear number of clauses is very different from one with a linear number of clauses. Molloy [Mol02] shows that we can force the satisfiability threshold to occur with a linear number of clauses if we restrict our problem model to have constant sized constraints and domain. In addition, we know the satisfiability threshold for a few NP-complete problems with constant sized domain and constraints: 1-in-k-SAT [ACIM01], a mixture of 2-SAT and 3-SAT when the number of clauses of size 3 is kept small [MZK$^+$99], and a model of [MS07]. However, in each of these cases the proofs of the threshold demonstrate that, unlike k-SAT, $k \geq 3$, the models are very similar to random 2-SAT at the satisfiability threshold and thus easy to solve almost surely.

This thesis identifies a new class of constraint satisfaction problems, and this class contains a problem that has the following properties, all of which are known to hold for k-SAT, $k \geq 3$. The problem is NP-complete, has constant size constraints and domain, and a uniformly random instance with a linear number of clauses a.s. has exponential resolution complexity. None of the CSPs that have a known satisfiability threshold also have all of these properties. We come very close to determining an exact satisfiability threshold for the random model of the problem defined in this thesis, and if we can find an exact satisfiability threshold, it will be the first NP-complete CSP that both has an exact satisfiability threshold located when the number of clauses is linear in the number of variables and also is not known to have a polynomial time algorithm that correctly, with uniform positive probability, decides an instance drawn from close to the satisfiability threshold. Because these properties suggest that problems in this CSP model may be closer to random k-SAT than the other CSP's for which we know the satisfiability threshold, studying this new class of problems may lead to a better understanding of random k-SAT. This thesis will examine several subclasses of CSPs from this model, characterize their complexity, study their satisfiability thresholds, and explore the behavior of several algorithms on these problems. The new class of problems includes XOR-SAT. Therefore, many of the results proven about the general class of CSPs will also be true for XOR-SAT.

The analysis of the satisfiability threshold for these problems involves a complicated second moment analysis, and part of this analysis depends on a certain function having a unique maximum in a specified domain. We are able to identify one local maximum in this domain, and we provide numerical evidence that there are no other maxima in the domain, but we can not prove non-existence of another maximum. We specify as the Maximum Hypothesis this assumption that the function has one maximum in the domain, and verifying the Maximum Hypothesis is the only obstacle preventing our determining the exact satisfiability threshold for the CSP. Hypothesis 4.2 gives the precise description of the Maximum Hypothesis. Section 4.4 discusses, in more detail, the function, its known local maximum, and the domain for the Maximum Hypothesis. Section 4.6 gives numerical evidence supporting the Maximum Hypothesis.

As a short summary of the main results of this thesis, Table 1.1 lists the threshold behavior we prove for one NP-complete problem, called $(3,4)$-UE-CSP, that is contained in the class of constraints satisfaction problems defined in this thesis.

Threshold Type	Threshold Value
Threshold for DPLL with the unit clause heuristic running in polynomial vs. exponential time, w.u.p.p.	.666666...
Maximum threshold at which any known solver will find a satisfying assignment in polynomial time, a.s.	.818469...
Threshold for satisfiability, a.s., conditional on the Maximum Hypothesis	.917935...

Table 1.1: Summary of threshold results for the NP-complete constraint satisfaction problem $(3,4)$-UE-CSP.

If we consider the related problem of XOR-SAT, the first and third thresholds of Table 1.1 still hold. The third threshold, without the condition, was originally proven for XOR-SAT in [DM02], and the first is proven for XOR-SAT in this thesis. The second threshold does not hold for XOR-SAT because XOR-SAT is in P, but we do get a similar threshold for XOR-SAT at the same value if we restrict the analysis to greedy algorithms.

The outline of the thesis is as follows. In Section 1.1, we formally define what is meant by a constraint satisfaction problem, and we give notations that we will use throughout the thesis. In Section 1.2, we list some known probability bounds and some facts on random hypergraphs that we will use to prove results in this thesis.

In Chapter 2, we survey the current state of research in resolving the Satisfiability Threshold Conjecture. We also examine what is known about the behavior of DPLL and other SAT-solving algorithms on random 3-SAT instances with a linear number of clauses, how techniques from statistical mechanics give us a better understanding of the properties of random k-SAT, and what is known about the satisfiability threshold and algorithm behavior for random XOR-SAT.

In Chapter 3, we generalize XOR-SAT to a new family of CSPs that we denote UE-CSP, and we identify NP-complete variations of the family. We show that for random k-UE-CSP, similar to random k-SAT, $k \geq 3$, a uniformly random instance of k-UE-CSP with n variables and cn clauses, $c > 0$, a.s. has exponential resolution complexity.

In Chapter 4, we show that, under the Maximum Hypothesis, k-UE-CSP has an exact satisfiability threshold and we identify the location for each k. We also give numerical evidence supporting the Maximum Hypothesis.

In Chapter 5, we completely characterize the behavior of two DPLL variations on UE-CSP, and we prove a theorem for random UE-CSP that is analogous to an open question for random SAT.

In Chapter 6, we prove the size of cores of non-uniform hypergraphs building on results for uniform hypergraphs of [PSW96, MWHC96, Mol05, Coo04, Kim06, CW06, DN, Rio07]. This result is required for the main theorem of Chapter 5.

Since introducing the k-UE-CSP problem at the International Conference on Theory and Applications of Satisfiability Testing (SAT 2004), the k-UE-CSP problem has been used to test satisfiability solvers [BS04, LSB05, HvM06, HvM07]. Other work includes a study of the clause structure that is created when transforming an instance of random k-UE-CSP into a boolean formula in conjunctive normal form [Her06], and a study of how the unit clause and generalized unit clause algorithms perform on an instance of random k-UE-CSP [AMZ07]. The results of this latter study are non-rigorous and similar to the results we achieve in Chapter 5 of this thesis.

1.1 Constraint Satisfaction Problems

First, we will formally define what is meant by a constraint satisfaction problem. A *constraint satisfaction problem* is a set of n variables where each variable has a non-empty domain of possible values and a set of m clauses where a *clause* both is an ordered subset of variables and has one or more constraints applied to it. A *constraint*, applied to a clause, restricts the values we may assign the variables of the clause. The goal is to find an assignment to the variables such that every constraint is satisfied. One common constraint satisfaction problem model is to use the same domain of values for every variable. Typically, a domain of d values is represented by the set $\{0, \ldots, d-1\}$, and the domain is assumed to contain at least two values.

A constraint is usually represented as either a list of the value tuples permitted or a list of the value tuples forbidden by the constraint. In the literature, each clause typically has exactly one constraint that lists all the forbidden or permitted tuples for that clause. As a result, the terms *clause* and *constraint* are often used interchangeably in the literature. In this thesis, we will deviate from the standard notation slightly.

Definition 1.2 (Constraint) *A constraint is a fixed relation on a canonical ordered set of variables. A constraint lists the permitted (or forbidden) tuples of values that we may assign to the variables.*

We will usually assume each clause has exactly one constraint, but we will occasionally apply more than one constraint to a clause. If a clause has multiple constraints, then we can apply a tuple of values to the variables of the clause only if that tuple is permitted by each of the constraints applied to the clause. This notation will simplify the presentation in Chapter 3 .

In keeping with SAT notation, we will denote an instance of a constraint satisfaction problem as a *formula.* A formula consists of the set of variables, the set of clauses, and for each clause, the constraint or constraints applied to the clause.

For the remainder of this text, n will denote the number of variables and m the number of clauses of a constraint satisfaction problem. Unless otherwise noted, m is always a linear function of n, and c is used to denote the clause density $m = cn$. Likewise, k will always refer to the clause or constraint size and d will denote the domain size. For simplicity, clauses of size k will be denoted as k-clauses.

1.2 Useful Tools

Second we will list the probability bounds that we use in this thesis, and we will give some basic facts on random hypergraphs.

1.2.1 Probability Bounds

Given a random variable X, the following are useful tools for bounding the probability that X deviates from its expected value.

Markov's inequality states that if X is non-negative, then

$$\mathbf{Pr}(X \geq t) \leq \frac{\mathbf{E}(X)}{t}.$$

If we set $t = 1$, we get $\mathbf{Pr}(X > 0) \leq \mathbf{E}(X)$. If $Y_1, Y_2, \ldots$ is a sequence of non-negative random variables with $\mathbf{E}(Y_n) = \mathrm{o}(1)$, then Y_n is a.s. 0. The technique of using Markov's inequality to bound the probability that X is larger than some value t is known as the *first moment method.*

Chebychev's inequality states

$$\mathbf{Pr}(|X - \mathbf{E}(X)| \geq t) \leq \frac{\mathbf{Var}(X)}{t^2}.$$

The technique of using Chebychev's inequality to bound the probability that X differs from its expected value is known as the *second moment method.*

If X is non-negative, a well known application of the Cauchy-Swartz inequality gives

$$\mathbf{Pr}(X > 0) \geq \frac{\mathbf{E}(X)^2}{\mathbf{E}(X^2)},$$

and using this inequality is also known as the second moment method.

The *Chernoff bound* has several forms. Sufficient for our purposes is the following simple variation. Let X be a binomial random variable that counts the number of successful trials from a set of n trials with probability of success p. Then,

$$\mathbf{Pr}(|X - \mathbf{E}(X)| \geq t) \leq e^{-\frac{2t^2}{n}}.$$

Similarly, *Azuma's inequality* [Azu67] has more than one variation. The following variation, sufficient for our purposes, is a corollary of the original Azuma's inequality.

Lemma 1.3 *Let X be determined by a series of random trials $T_1, \ldots, T_n$ such that for all i,*

$$|\mathbf{E}(X \mid T_1, \ldots, T_{i-1}) - \mathbf{E}(X \mid T_1, \ldots, T_i)| \leq c_i.$$

Then

$$\mathbf{Pr}(|X - \mathbf{E}(X)| > t) < 2e^{-\frac{t^2}{2\sum c_i^2}}.$$

1.2.2 Some Facts on Random (Hyper)graphs

A *hypergraph* is a finite set of vertices and a set of hyperedges. Each hyperedge is a subset of the vertices. If every hyperedge has the same size, the hypergraph is called *uniform*, and if every hyperedge has size 2, we have a graph.

A useful technique when analyzing the structure of an instance C of a constraint satisfaction problem is to consider the underlying hypergraph H of C. Define H to have as vertices the set of variables of C, and define the hyperedges of H to be exactly the clauses of C. Usually, each clause is assumed to have one constraint. If a clause has multiple constraints applied to it, we can model this by a multi-hyperedge in the hypergraph. As with clauses, a hyperedge of size i will be denoted as an i-edge.

The two basic models of random graphs extend to uniform hypergraphs, and we will use the same notation for the random uniform hypergraph models as is used for the random graph models. For any fixed hyperedge size k, we will let $G_{n,m}$ denote a random hypergraph drawn from the model where we choose m random hyperedges uniformly at random from all possible hyperedges of size k, and we will let $G_{n,p}$ denote a random hypergraph drawn from the model where we consider each possible hyperedge of size k and include it with probability p. These two models are equivalent in the sense that if $m = \binom{n}{k}p$, note that strict equality is not needed but is sufficient for our purposes, then for a property Q if $\lim_{n\to\infty} \mathbf{Pr}(G_{n,p} \text{ has } Q) = a$ then $\lim_{n\to\infty} \mathbf{Pr}(G_{n,m} \text{ has } Q) = a$. Likewise, for a monotone property Q, if $\lim_{n\to\infty} \mathbf{Pr}(G_{n,m} \text{ has } Q) = a$ then $\lim_{n\to\infty} \mathbf{Pr}(G_{n,p} \text{ has } Q) = a$ [Bol79, Łuc90].

Here are a few useful facts on random hypergraphs when there is a linear number of edges. These facts were first discovered by Erdös and Rényi [ER59, ER60] for random graphs and extended by Schmidt and Shamir [SS85] and Karoński and Łuczak [KŁ02] for random uniform hypergraphs. Let k be the hyperedge size. The hypergraph a.s. has a linear number of isolated vertices and isolated components that are trees. On a random hypergraph with n vertices and cn edges, if $c < \frac{1}{k(k-1)}$ almost all components are hypertrees with possibly a constant number of cycles. The largest such component a.s. has size $\mathrm{O}(\log n)$. If $c > \frac{1}{k(k-1)}$, a.s. one component of the hypergraph will have size $\Theta(n)$, and the number of cycles now grows unbounded as n tends to infinity. Most of these cycles have length greater than $\log n$, a.s., and for each constant j the number of cycles of length j a.s. remain bounded by a constant.

Chapter 2

Background

2.1 The Satisfiability Threshold Conjecture

The Satisfiability Threshold Conjecture states that there exists a value c_k^* such that a uniformly random k-SAT formula on n variables and cn clauses is almost surely unsatisfiable if $c > c_k^*$ and almost surely satisfiable if $c < c_k^*$.

For 2-SAT, $c_2^* = 1$, a result proven independently by Chvátal and Reed [CR92], Fernandez de la Vega [FdlV92], and Goerdt [Goe96]. The proof uses the well-known fact, first observed in Aspvall, Plass, and Tarjan [APT79], that we can model a 2-SAT formula as a directed graph with the literals as the vertices. For each clause (l_1, l_2) there are two directed edges $\overline{l_1}l_2$ and $\overline{l_2}l_1$. A 2-SAT formula is unsatisfiable if and only if a variable and its complement both appear in the same strongly connected component of the directed graph. The threshold proof then follows by showing that if $c < 1$, there a.s. is no such strongly connected component and if $c > 1$ there a.s. is such a strongly connected component.

The satisfiability threshold conjecture remains open for $k \geq 3$. While neither the existence nor the location of c_k^* is known for any $k \geq 3$, Friedgut [Fri99] proves that the satisfiability threshold for k-SAT is sharp. However, the location of the threshold might not be at the same clause density for each value of n. Specifically, for each $k \geq 2$ there exists a function $c_k^*(n)$ such that for a uniformly random k-SAT instance with n variables and cn clauses, if $c < c_k^*(n)$ the formula is a.s. satisfiable and if $c > c_k^*(n)$ the formula is a.s. unsatisfiable.

On the other hand, we say a constraint satisfaction problem has a *coarse threshold* if there exists a function $r(n)$, and for each $\delta > 0$ there exists $\epsilon_1, \epsilon_2 > 0$ such that for a uniformly random instance with n variables and cn clauses, if $c = r(n)$, the probability the instance is satisfiable is $\frac{1}{2}$, if $c = r(n) - \epsilon_1$, the probability the instance is satisfiable is $\frac{1}{2} + \delta$, and if $c = r(n) + \epsilon_2$, the probability the instance is satisfiable is $\frac{1}{2} - \delta$. In this thesis we will only consider thresholds that occur when there is a linear number of clauses.

Friedgut proves that monotonic properties with coarse thresholds on hypergraphs can be approximated by the property of the existence of a finite number of small subgraphs. From Bourgain's extension to Friedgut's theorem, located in the appendix of [Fri99], to prove that k-SAT has a sharp threshold, it is sufficient to show that for a formula F on n variables drawn uniformly randomly from all formulae with clause density close to the satisfiability threshold, neither of the following two cases hold: that F contains w.u.p.p. an unsatisfiable subformula of constant size nor that there exists a satisfiable formula ϕ of constant size such that the probability F is satisfiable conditional on ϕ being a subformula of F is larger than the probability F is satisfiable, and the difference in the two probabilities depends on the size of ϕ and not on n. Roughly speaking, we can think of properties with coarse thresholds as being local in nature while properties with sharp thresholds are not. Friedgut's theorem provides a very useful corollary: if we can prove a uniformly random formula with $c'n$ clauses is w.u.p.p. satisfiable then a uniformly random formula with cn clauses $c < c'$ is a.s. satisfiable.

We have a tight asymptotic bound for the conjectured c_k^*. The observation, first made by Franco and Paull [FP83], that the expected number of satisfying assignments of a random formula is $2^n \left(1 - 2^{-k}\right)^{cn}$ yields $c_k^* \leq 2^k \log 2$. Achlioptas and Peres [AP04] proves this bound is asymptotically tight, as k tends to infinity, by proving $c_k^* \geq 2^k \log 2 - \mathrm{O}(k)$. The proof by Achlioptas and Peres is non-algorithmic and gives the best known lower bounds for c_k^*, $k > 3$, but the lower bound it gives for 3-SAT is weaker than the bounds found by algorithmic analysis. From experimental evidence, the threshold for 3-SAT is $c_3^* \approx 4.2$ [KS94, SML96, CA96], and the current state of the research has 3.52 [HS03, KKL03] $\leq c_3^* \leq$ 4.506 [DBM03].

2.1.1 Lower Bounds and Algorithm Analysis

The proof of the current lower bound for the 3-SAT satisfiability threshold uses algorithm analysis. To prove a threshold bound with this method, one must develop both an algorithm for 3-SAT and the techniques to analyze that algorithm. Then one must identify the maximum density c of clauses for which the algorithm will w.u.p.p. find a satisfying assignment on a uniformly random instance drawn from that clause density, and Friedgut's result [Fri99] implies $c_3^* \geq c$. A survey of this technique is in [Ach01]. However, it is not clear that this technique will succeed in finding the exact threshold, if it exists. The algorithms that are currently amenable to analysis only find solutions on instances drawn from well below the conjectured satisfiability threshold, and it is not known whether there even exists an algorithm that w.u.p.p. finds a satisfying assignment of a random instance near the conjectured threshold. However, empirical results for a new algorithm, survey propagation [BMZ05], suggest that the algorithm can find solutions to random problems drawn from quite close to the conjectured threshold, but the algorithm is too complicated for exact analysis by current techniques.

In general, the algorithms that can be analyzed are greedy, non-backtracking algorithms. The reason for this restriction is that the analysis requires that after each step or sequence of steps by the algorithm, the subformula induced by the unassigned variables is still uniformly random, possibly conditional on some parameter such as the degree sequence. Algorithms that have been considered for 3-SAT and that have led to incremental improvements in the lower bound for the 3-SAT satisfiability threshold include the following. The *unit clause* (UC) algorithm works by repeatedly setting the literal of a clause of size 1 to *true*, and if there is no clause of size 1 then assigning a variable at random. Chao and Franco [CF86] proves that UC succeeds w.u.p.p. when the clause density is less than $\frac{8}{3}$. Chao and Franco also proves that for each variable chosen at random, if you assign the variable in such a way as to satisfy the majority of the 3-clauses in which that variable appears, then the algorithm succeeds w.u.p.p. for densities less than 2.9. These results did not, at the time, give lower bounds on c_3^* because we did not have the Friedgut result. The first lower bound for the 3-SAT satisfiability threshold was from Broder, Frieze, and Upfal's [BFU93] analysis of the pure literal rule. The *pure-literal rule* is to iteratively set to *true* each literal whose complement does not occur in the formula, and [BFU93] proves that this algorithm succeeds a.s. up to clause density 1.63. The next improvement in the lower bound for the 3-SAT threshold came from Frieze and Suen's [FS96] study of *generalized unit clause* (GUC): choose a clause of shortest length from the subformula induced by the unassigned variables and set to *true* a random literal from that clause. GUC succeeds w.u.p.p. for clause densities less than $3.003\ldots$, and by showing that modifying GUC with a limited amount of backtracking succeeds a.s. for clause densities less than $3.003\ldots$, [FS96] establishes that $c_3^* \geq 3.003\ldots$. Achlioptas [Ach00] modifies UC such that if there is no unit clause and there is a 2-clause then set both its literals in such a way as to minimize the number of 3-clauses that become 2-clauses, and [Ach00] uses this algorithm to prove that $c_3^* > 3.145$.

Except for the Frieze and Suen modification to GUC, all the algorithms considered do not change the assignment to a variable once it is made, and the Frieze and Suen algorithm uses a very limited backtracking that leaves most of the formula uniformly random. In addition, each algorithm considered, except the pure-literal rule, has the property that the next variable to assign is either chosen uniformly at random or selected from a random clause of a specific length. Once a variable is chosen, every literal on that variable is exposed and a value for the variable is selected. Achlioptas and Sorkin [AS00] gives the term *myopic* for such algorithms, and Achlioptas and Sorkin proves that the maximum clause density at which a myopic algorithm that selects one variable at a time will succeed a.s. is 3.22, and if the algorithm selects up to two variables at a time, the maximum clause density is 3.26. This result establishes both that $c_3^* > 3.26$ and that a myopic algorithm will not achieve additional improvements in the lower bound of the satisfiability threshold without considering more than two variables at a time, and these improvements will be insignificant and tedious.

Kaporis, Kirousis, and Lalas [KKL02] improves the lower bound to $c_3^* \geq 3.42$ by considering the algorithm that sets to *true* a literal that appears in as many clauses as possible while also satisfying any unit clauses that appear. The current lower bound on the 3-SAT threshold comes from an algorithm that selects and sets a literal to *true* based on the degree of the literal and the degree of its complement. Both Kaporis, Kirousis, and Lalas [KKL03] and Hajiaghayi and Sorkin [HS03] independently analyze slightly different variations of this algorithm to get the lower bound of 3.52.

The upper bounds are proven using statistical counting techniques. Being non-algorithmic, this method appears to hold more promise for finding the threshold if it exists, but this method has its own challenges that are dealt with in the next section.

2.1.2 Upper Bounds and Statistical Techniques

The typical statistical tools used to find bounds for the satisfiability threshold are known as the *first moment method* and the *second moment method*. If we let the random variable X be the number of solutions to a uniformly random CSP, the first moment method yields $\mathbf{Pr}(X > 0) \leq \mathbf{E}(X)$. Thus, the goal is to find the minimum constant c such that for a random formula with n variables and cn clauses $\mathbf{E}(X) = \mathrm{o}(1)$. To compute a lower bound, we use the second moment method which involves computing $\mathbf{E}(X^2)$. If we can show $\mathbf{E}(X^2) = (1+\mathrm{o}(1))\mathbf{E}(X)^2$, then, by an application of the Cauchy-Swartz inequality, we have $\mathbf{Pr}(X > 0) \geq \frac{\mathbf{E}(X)^2}{\mathbf{E}(X^2)} = 1 - \mathrm{o}(1)$.

The main challenge in using these techniques to find good bounds on the satisfiability threshold is in dealing with what are known as "jackpot phenomena". Namely, the property that one solution yields an exponential number of other solutions by changing the values of some of the variables. For the first moment method, this phenomenon hinders computation of the threshold because even if formulae with solutions are exponentially rare, one such formula with an exponential number of solutions is enough to give a high expected number of solutions. For the second moment method, the existence of jackpots means the random variable X will have a large variance. Because of this challenge, the current best lower bounds for c_3^* found using the second moment method are weaker than the lower bounds found using algorithmic analysis. However, the second moment method has been more successful with c_k^*, $k \geq 4$ [AP04].

A well known observation, the first known citation is by Franco and Paull [FP83], is that the expected number of satisfying assignments to a random 3-SAT formula with n variables and cn clauses is

$$2^n \left(\frac{7}{8}\right)^{cn}.$$

Therefore, Markov's inequality implies that a formula is a.s. unsatisfiable if $c \geq \log_{8/7} 2 \approx 5.191$. The improvements in the upper bound for the threshold have come from methods that reduce the jackpot phenomena.

The first improvement, due to Fernandez de la Vega and El Maftouhi [MdlV95], reduces the upper bound to 5.081. Kamath, Motwani, Palem, and Spirakis [KMPS95] improves the upper bound to 4.758 by computing the probability that a formula is satisfiable by dividing the expected number of satisfying assignments computed with Markov's inequality by a lower bound on the average number of satisfying assignments for all formulae that are satisfied by a given assignment.

Dubois and Boufkhad [DB97] decreases the upper bound to 4.643 by roughly calculating the expected number of solutions that have the property that flipping any variable value from *false* to *true* will yield an unsatisfying assignment. Kirousis, Kranakis, Krizanc, and Stamatiou [KKKS98], independently of [DB97], introduces the generalized technique of lexicographically ordering assignments as bit strings with *true* assigned 1 and *false* assigned 0. A satisfying assignment is *l-maximal* if switching the values of up to l variables does not yield a lexicographically larger assignment that is also satisfying. The technique of [DB97] is equivalent to counting the number of 1-maximal assignments. By counting the number of 2-maximal assignments, [KKKS98] lowers the upper bound for 3-SAT satisfiability to 4.6011. Janson, Stamatiou, and Vamvakari [JSV00] further improves this bound to 4.596 by improving the estimate for the number of 2-maximal assignments. By providing a better estimate of the probability that a satisfying assignment is 1-maximal and combining this with the probability determined in [KKKS98] that a satisfying assignment is maximal over double-flips in which one literal is flipped *false* to *true* and another *true* to *false*, Kaporis, et al, [KKS$^+$01] further improves the lower bound to 4.571. We could get better upper bounds by calculating the expected number of l-maximal assignments for $l > 2$, but the calculations quickly get complicated for larger values of l.

The current upper bound for the 3-SAT satisfiability threshold, 4.506 by Dubois, Boufkhad, and Mandler [DBM03], combines the idea of 1-maximal assignments with a structural argument that considers only "typical" formulae, proving that the "atypical" formulae occur almost never. Dubois, Boufkhad, and Mandler [DBM03] demonstrates that typical formulae have the property that the number of occurrences of each variable follows a Poisson distribution, and the number of positive and negative occurrences of the variable follows a binomial distribution. Then, [DBM03] groups the typical formulae into equivalence classes where two formulae are equivalent if one can be transformed into the other by repeatedly selecting a vertex and flipping all its literals. The expected number of satisfying assignments for an equivalence class is found by counting the number of 1-maximal assignments for the representative that is assumed to have the fewest 1-maximal assignments and multiplying that value by the number of formulae in the equivalence class. The formula assumed to have the fewest assignments is the one for which every variable has at least as many occurrences as a negative literal as it does as a positive literal. Note, the authors are inverting the definition of 1-maximal.

In all of the cases for the upper bound of 3-SAT, the first moment method is used. One variation of SAT for which the second moment method is useful is NAE-SAT [ACIM01, AM06] (at least one true and one false literal per clause). NAE-SAT has a greatly reduced jackpot phenomenon because there is less freedom on how to satisfy each clause. Although an exact satisfiability threshold for NAE-SAT is not known, the bound is very tight for large k. This tight bound is used in Achlioptas and Moore [AM06] to greatly improve the asymptotic lower bound for k-SAT, and the k-SAT lower bound is further improved in Achlioptas and Peres [AP04] for all $k \geq 4$ by using the second moment method on balanced assignments that are similar to NAE-SAT assignments.

2.2 The $(2+p)$-SAT Model

With random 2-SAT and random 3-SAT behaving differently and in order to understand what happens between $k = 2$ and $k = 3$, Monasson, et al, [MZK$^+$96] introduces the $(2+p)$-SAT model which contains pcn clauses of size 3 and $(1-p)cn$ clauses of size 2. The analogous conjecture to the satisfiability threshold conjecture is that $(2+p)$-SAT has an exact satisfiability threshold for each value of p. Clearly, the results for SAT give us an exact satisfiability threshold for $p = 0$, and the threshold is not known to exist for $p = 1$. Likewise, it is clear that the random $(2+p)$-SAT instance will be a.s. unsatisfiable if the 2-clauses alone are a.s. unsatisfiable, when $(1-p)c > 1$, or when the 3-clauses alone are a.s. unsatisfiable, when $pc > 4.506$.

The current bounds on the satisfiability threshold for random $(2+p)$-SAT come from Achlioptas, Kirousis, Kranakis and Krizanc [AKKK01]. The exact satisfiability threshold exists for $p \leq \frac{2}{5}$, and the threshold is at the clause density $\frac{1}{1-p}$. The proof of the satisfiability threshold involves analyzing the unit clause heuristic to find the greatest clause density for which UC will w.u.p.p. find a satisfying assignment, and the authors prove that Friedgut's [Fri99] result that k-SAT has a sharp threshold also applies to $(2+p)$-SAT. As the location of the satisfiability threshold indicates, when $p \leq \frac{2}{5}$ the $(2+p)$-SAT formula is a.s. satisfiable if the 2-SAT problem induced by the 2-clauses is a.s. satisfiable. This result implies that if we have $(1-\epsilon)n$ random 2-clauses, we can add up to $\frac{2}{3}n$ random 3-clauses and still be a.s. satisfiable. The conjecture is that this bound on the behavior of $(2+p)$-SAT is tight. That is, for $p > \frac{2}{5}$, the satisfiability threshold for $(2+p)$-SAT, if it exists, will be at a clause density strictly less than $\frac{1}{1-p}$. If true, this conjecture implies that for every $\delta > 0$ there exists an $\epsilon > 0$ such that a uniformly random instance of $(2+p)$-SAT with $(1-\epsilon)n$ 2-clauses and $\left(\frac{2}{3}+\delta\right)n$ 3-clauses is a.s. unsatisfiable. We denote this last conjecture the *$(2+p)$-SAT Conjecture.*

For the case when $p > \frac{2}{5}$, [AKKK01] gives a lower bound for the satisfiability threshold, if it exists, by again analyzing the greatest clause density at which the unit clause algorithm will find, w.u.p.p., a satisfying assignment. The upper bound is found by using the same technique of counting maximal satisfying assignments used in [KKKS98] for the upper bound of 3-SAT. This upper bound is strictly less than $\frac{1}{1-p}$ when $p > 0.695$, and from this upper bound, we know that there exists an $\epsilon > 0$ such that a random $(2+p)$-SAT formula with $(1-\epsilon)n$ 2-clauses and $2.28n$ 3-clauses is a.s. unsatisfiable.

As a result, we have a gap where, at $p \leq \frac{2}{5}$, a random $(2+p)$-SAT formula is a.s. satisfiable if and only if the density of the 2-clauses lies below the satisfiability threshold for 2-SAT, and when $p > 0.695$ the a.s. satisfiability of the formula depends on the densities of both the 2-clauses and 3-clauses. Also, for any random 2-SAT instance drawn from close to but below the satisfiability threshold, we can add up to $\frac{2}{3}n$ 3-clauses to the formula and still be a.s. satisfiable, but once we add $2.28n$ 3-clauses, the formula is a.s. unsatisfiable. We would like to close this gap, and the $(2+p)$-SAT Conjecture implies that the lower bound is tight.

2.3 Algorithm Behavior on Random SAT

2.3.1 The Davis Putnam Logemann Loveland (DPLL) Algorithm

DPLL [DLL62, DP60] forms the basis of most current complete SAT solvers where a complete solver is an algorithm that will find a solution if one exists. The DPLL algorithm has a simple backtracking framework. At each step, an unassigned variable v is assigned a value. Any clause that is satisfied by the assignment is removed, and v is removed from any clause in which it occurs. DPLL then recurses on this reduced formula. If a conflict occurs, DPLL backtracks and tries a different value for v. There are many variations that can be made in choosing the next variable, choosing the appropriate value to try, propagating the implications of variable assignments through the constraints, learning new clauses, and restarting. Researchers have noticed experimentally, for example the study by Selman, Mitchell, and Levesque [SML96], that DPLL works fast on random problems drawn from well below or above the conjectured satisfiability threshold, but it performs poorly on problems drawn from near the satisfiability threshold.

The idea that DPLL quickly proves unsatisfiable a problem drawn from well above the satisfiability threshold is somewhat misleading and is an artifact of the small problem sizes used in the studies. A resolution proof of unsatisfiability for a SAT instance is a sequence of clauses, ending with the empty clause, and such that each clause is either a clause of the instance or is derived from two previous clauses of the sequence, C_i and C_j, by the following rule. Clause C_i contains the literal x, clause C_j contains the literal $\overline{x}$, and the derived clause contains all literals of C_i and C_j not involving the variable x. Chvátal and Szemerédi [CS88] proves that an unsatisfiable random k-SAT instance with a linear number of clauses will a.s. require a resolution proof with an exponential number of clauses. We define the length of a resolution proof to be the number of clauses in the proof. The length of the shortest resolution proof of unsatisfiability is the *resolution complexity*, and a well known observation of Galil [Gal77] is that exponential resolution complexity implies that DPLL will require an exponential amount of time to prove the problem unsatisfiable.

The algorithm analysis used to find the lower bound on the satisfiability threshold also gives a bound on the running time of DPLL. DPLL can use a variety of heuristics to guide it in finding a solution to a problem instance. For example, to select the next variable to assign a value, to choose the value to assign, and to trim the search space. If this heuristic, running as a stand-alone greedy algorithm, can find a solution w.u.p.p., then DPLL using that heuristic will w.u.p.p. not have to backtrack. In particular, DPLL using unit clause (DPLL+UC) as its heuristic will run in linear time w.u.p.p. if $c \leq 2.66$ [CF86], and DPLL using generalized unit clause (DPLL+GUC) will run in linear time w.u.p.p. if $c \leq 3.003$ [FS96].

One might think that at a slightly higher clause density, DPLL will backtrack a few times but still run in linear or polynomial time. However, if the $(2+p)$-SAT Conjecture is true, then the bounds listed in the preceding paragraph are in fact the border between linear and exponential running times for the algorithms.

Instead, we currently have a gap between the greatest density at which DPLL w.u.p.p. runs in linear time and the least density at which DPLL will require w.u.p.p. exponential time. To find the bound for exponential behavior, Achlioptas, Beame, and Molloy [ABM04b] starts with a random $(2+p)$-SAT instance with $2.28n$ 3-clauses and $(1-\epsilon)n$ 2-clauses, for sufficiently small $\epsilon > 0$. Such a random formula is proven a.s. unsatisfiable in [AKKK01], and Achlioptas, Beame, and Molloy proves the formula a.s. has exponential resolution complexity. Results of Chao and Franco [CF90] for UC and Frieze and Suen [FS96] for GUC, both of which are simplified in Achlioptas [Ach01], prove that we can trace the behavior of UC and GUC using a system of differential equations. Using these systems, [ABM04b] works backward to find the smallest clause density of a random 3-SAT instance on which UC and GUC will w.u.p.p. reach the unsatisfiable $(2+p)$-SAT instance without backtracking. As a result, DPLL+UC will require exponential time w.u.p.p. to solve a random 3-SAT instance with n variables and cn clauses if $c \geq 3.81$. For DPLL+GUC, the exponential behavior occurs w.u.p.p. when $c \geq 3.98$.

2.3.2 Other Algorithms for Random SAT

Two other classes of algorithms used on random SAT are local search and belief propagation. Local search is a very general framework where the algorithm starts at an arbitrary assignment to the variables, and until a solution is found, the value to a selected variable is flipped. The *pure random walk* algorithm starts with a random assignment to the variables. Then it repeatedly chooses at random an unsatisfied clause and a variable from that clause, and the assignment to that variable is flipped. Alekhnovich and Ben-Sasson [ABS03] proves that the pure random walk algorithm a.s. finds a satisfying assignment in polynomial time if $c < 1.63$. The similarity of the bound with that of the pure-literal rule is not coincidental and is due to the heavy reliance on pure literals in the proof. However, experimental evidence in Parkes [Par02] suggests that the pure random walk algorithm will succeed in polynomial time up to $c < 2.65$, and this value is supported by non-rigorous analysis of Semerjian and Monasson [SM03]. Other variations of local search include *gradient descent*: flip a randomly chosen variable only if it decreases the number of unsatisfied clauses, and GSAT: a hill climbing procedure that chooses a random variable from those variables that yield the maximum number of satisfied clauses when their value is changed.

Roughly speaking, the standard belief propagation algorithm works by starting with an arbitrary probability distribution on each variable where the probability distribution is over possible assignments to the variable. At each iteration of the algorithm, a variable

v recomputes its marginal distribution as follows. For each neighbor u of v, v computes a probability distribution for itself using the probability distributions, received in the previous iteration, for each of its neighbors except u, and it sends this new probability distribution to u. Once v receives a new probability distribution from each neighbor, it uses these to recompute its own marginal distribution. This process repeats until either the marginal distributions converge or until a set number of iterations is reached. At that time, the variable, or set of variables, that has the strongest bias in its marginal distribution is assigned the value with greatest bias, and then the whole process repeats.

These latter algorithms are too complicated for rigorous analysis, but none of them are known, experimentally, to w.u.p.p. find a satisfying assignment in polynomial time (in the case of a random walk algorithm), or to find a satisfying assignment at all (in the case of gradient descent and belief propagation), at clause densities above 3.921. The only algorithm that is known, experimentally, to find satisfying assignments above this density in polynomial time is the survey propagation algorithm, a recent variation of belief propagation, that is discussed in Section 2.4.4.

2.4 Contributions from Physics

Statistical mechanics is a branch of physics that attempts to derive the behavior of large systems from an understanding of the behavior of individual particles within the system. In the systems studied, the number of particles is enormous making an exact calculation impossible. Instead, statistics are used to discover the almost sure behavior of the system. For example, one family of models of statistical mechanics, called the spin-glass-like models, is used to study physical transitions, such as when materials cool and crystallize, but where the particles do not all align in the same direction. A large survey of the model is in Mézard, Parisi, and Virisoro [MPV87]. In most natural processes particles interact with all other particles in their neighborhood, and the strength of the interaction is proportional to the distance between the particles. However, [MPV87] notes that by allowing arbitrary particle interactions, the spin-glass models could model any constraint satisfaction problem, and finding the expected behavior of the model at zero temperature is equivalent to solving the CSP.

2.4.1 The Replica Method

Monasson and Zecchina [MZ96, MZ97] model k-SAT as a spin-glass problem and use the statistical mechanics technique known as the replica method with symmetry breaking to study the model. While Monasson and Zecchina conjecture that the replica method should be able to find the 3-SAT satisfiability threshold, even if it does it will not be a proof of the Satisfiability Threshold Conjecture. The replica method is not mathematically sound. In particular, one step of the replica method involves determining, or estimating, an expression for the integer nth moment of a random variable, and then taking the limit as the real n goes to 0. There is some work, notably by Talagrand [Tal01, Tal03a, Tal03b], in determining when the assumptions implicit in the replica method hold. Also, properly applying the needed symmetry breaking is as much an art as a science. On its own, the replica method provides a (not sound) upper bound of the threshold location, and symmetry breaking is used to tighten the bound. Through iterative improvements in symmetry breaking, Biroli, Monasson, and Weight [BMW00] gets the upper bound of 4.48 and Franz, Leone, Ricci-Tersenghi, and Zecchina [FLRTZ01] gets the upper bound 4.396. Mézard and Zecchina [MZ02] achieves a non-rigorous upper bound of 4.267 by applying a technique known as the cavity method with one-step replica symmetry breaking and introduces the survey propagation algorithm for random SAT, inspired by this technique. In [MZ02] and [MMZ06], the authors conjecture that this bound is very close to the satisfiability threshold because there is evidence that applying additional symmetry breaking to the cavity method will only yield very small improvements to the estimate. Additional evidence for the validity of the cavity method comes from Mertens, Mézard, and Zecchina [MMZ06] where the authors apply the cavity method to estimate the satisfiability threshold for larger values of k, and in each case, the estimate provided by the cavity method falls between the current proven bounds for the satisfiability threshold. The current best estimate of the threshold for 3-SAT using the the cavity method is 4.26675 ± 0.00015 [MMZ06]. The replica method does correctly predict the threshold for both 2-SAT [MZ97] and 3-XOR-SAT [FLRTZ01], and the cavity method correctly finds the 3-XOR-SAT threshold [MRTZ03].

2.4.2 Order Parameters

When studying thresholds, physicists look for *order parameters*. An order parameter is a value that is a.s. zero on one side of the threshold and a.s. non-zero on the other. Monasson and Zecchina [MZ97] uses an order parameter called the *backbone*, and the backbone is defined as the set of variables that must have the same value in all assignments that minimize the number of unsatisfied clauses. The analysis of the replica technique and empirical evidence of [MZ97] suggests a difference between 2-SAT and k-SAT, $k \geq 3$. In 2-SAT, the size of the backbone appears to increase continuously as the clause density crosses the satisfiability threshold, but for $k \geq 3$, the size of the backbone appears to jump discontinuously to $\Omega(n)$.

Bollobas, et al, [BBC+01] uses a different order parameter called the *spine* in a study of 2-SAT, and [BBC+01] defines the spine as the number of literals that, if added as a unit clause to some satisfiable subformula of the formula, makes that subformula unsatisfiable. Bollobas, et al, proves that the spine is an order parameter for 2-SAT, that 2-SAT has a continuous spine size at the satisfiability threshold, and [BBC+01] completely characterizes what is known as the finite size scaling window of random 2-SAT satisfiability. The sharp threshold for k-SAT satisfiability is an asymptotic result. An interesting question is to determine, for each n, the actual probability that a uniformly random instance of k-SAT on n variables and cn clauses is satisfiable. In this case, we do not have a sharp transition from 0 to 1 at the asymptotic threshold. Rather, the probability is close to 1 if we are well below the asymptotic threshold, gradually moves from near 1 to near 0 in a region around the threshold, and is close to 0 if we are well above the threshold. This "broadening" of the transition due to finite instances is the finite size scaling window. Specifically, for each n and constant $\delta > 0$, the window is defined to be the region in which the probability of satisfiability lies between δ and $1 - \delta$.

The spine is easier to manipulate analytically than the backbone because the spine is monotone in the sense that adding clauses to a formula can not decrease the spine size. Because of its nice properties, Boettcher, Istrate, and Percus [BIP05], generalizes the notion of a spine to generic CSPs and proves that the size of the spine for XOR-SAT jumps discontinuously to $\Omega(n)$ at the satisfiability threshold. [BIP05] also proves that if a CSP has a sharp satisfiability threshold when the number of clauses is linear in the number of variables and if the size of the spine jumps discontinuously to $\Omega(n)$ at that threshold, then a uniformly random instance with any linear number of clauses a.s. has exponential resolution complexity.

2.4.3 The Solution Space Topology

A further insight gained by the physical analysis is in understanding the topology of the solution space for k-SAT. The analysis of the replica technique and empirical evidence of Monasson and Zecchina [MZ97] suggests a clustering behavior of the solutions to the k-SAT instance. Given a random instance of 3-SAT drawn from well below the satisfiability threshold, all the satisfying assignments belong to a single cluster. Two satisfying assignments belong to the same cluster if we can transform one assignment into the other through a sequence of satisfying assignments where each intermediate assignment is formed by flipping the value of O(1) variables. However, close to the satisfiability threshold, the solution space appears to break into exponentially many clusters where two assignments in different clusters are separated by $\Theta(n)$ variable flips. In [MZ97], the onset of clustering is said to occur at a clause density of approximately 4. This estimated location of this threshold was improved to 3.96 by Biroli, Monasson, and Weight [BMW00], to 3.94 by Franz, Leone, Ricci-Tersenghi, and Zecchina [FLRTZ01], and finally to 3.921 by Mézard and Zecchina [MZ02].

Mézard and Zecchina [MZ02] conjectures that it is the existence of multiple clusters that causes any algorithm that depends on local information, such as the algorithms of Section 2.3, to fail. We consider an assignment to be minimal if changing the value of one variable will not decrease the number of unsatisfied clauses and minimum if the assignment leaves the minimum number of unsatisfied clauses, over all possible assignments. [MZ02] conjectures that there are exponentially more clusters that contain minimal assignments than there are that contain minimum assignments. In addition, [MZ02] conjectures that each cluster is distinguished by a subset of variables that have the same value in all assignments in the cluster. Achlioptas and Ricci-Tersenghi [ART06] proves that, for $k \geq 8$, there is a clause density d_k below the conjectured satisfiability threshold c_k^*, where the solution space a.s. breaks into an exponential number of clusters. Specifically, if we define an assignment graph where each vertex is an assignment to the variables and two assignments are adjacent if they differ in exactly one variable assignment, then a solution graph is a subgraph of the assignment graph induced on those vertices that correspond to a satisfying assignment. The clusters are the connected components of the solution graph, and [ART06] proves that for $k \geq 8$, there exist constants $a_k < b_k < \frac{1}{2}$ such that at a clause density $d_k < c_k^*$, a random k-SAT formula on n variables and $d_k n$ clauses a.s. has clusters of diameter at most $a_k n$, there are no satisfying assignments at distance more than $a_k n$ and less than $b_k n$ from each other in the assignment graph, and there are an exponential number of clusters at distance at least $b_k n$ from one another in the assignment graph.

Two other papers, Mézard, Mora, and Zecchina [MMZ05] and Daudé, Mézard, Mora, and Zecchina [DMMZ05], also show that for $k \geq 8$ there is a clause density below the conjectured satisfiability threshold at which the solution space a.s. breaks into clusters. Though the results in these papers are not rigorous.

2.4.4 Survey Propagation

As mentioned in Section 2.4.1, the use of the cavity method in [MZ02] led to the creation of the survey propagation algorithm. The cavity method was first developed by Mézard, Parisi, and Virasoro [MPV86] for analyzing spin-glass-like models, and the method is the same as the belief propagation algorithm developed by Pearl [Pea82] for Bayesian networks. The cavity method (and belief propagation) is a heuristic for solving a function on a graph when the value of the function at a node depends on the value of the function at the neighbors of the node. At each iteration of the algorithm, the function value at each node is computed using the current values at the neighbors of the node. The algorithm repeats until the values converge so that the change in the values after each iteration is below some threshold. In belief propagation, the value that is sent from one variable to another is called the *message*.

A series of three papers, Mézard and Zecchina [MZ02], Mézard, Parisi, and Zecchina [MPZ02], and Braunstein, Mézard, and Zecchina [BMZ05], introduces the survey propagation algorithm. Survey propagation is a variation of belief propagation, but it has a more refined message. In traditional belief propagation, the messages indicate the probability that a variable is assigned *true* versus *false*. In survey propagation, the messages indicate the probability that the variable is constrained to take a specific value or free to take any value. The result is that in traditional belief propagation, variables that are far apart in the formula can converge toward assignments that belong to different clusters while survey propagation does a better job of converging toward a single cluster. More specifically, the main assumption of survey propagation is that we can better estimate the fraction of clusters in which a variable is *true*, *false*, or unconstrained than we can estimate the fraction of solutions in which the variable is *true* versus *false*. Once all constrained variables are found and set, a faster local algorithm will be able to assign the remaining variables appropriately. Hsu and McIlraith [HM06] gives a modification to the belief propagation and survey propagation algorithms that guarantees convergence. The modification is based on the expectation maximation algorithm of Dempster, Laird, and Rubin [DLR77], but it is not known how the modification affects the likelihood of finding a solution.

Maneva, Mossel and Wainwright [MMW07] notes that survey propagation is essentially calculating the *core* assignment of a cluster. The core assignment of a cluster is found by taking any assignment from the cluster and repeatedly marking a variable as unconstrained if all clauses in which it occurs have either a different true literal or an unconstrained variable. The assignment to the constrained variables and the set of unconstrained variables form the core assignment. Note that the core is the same for all assignments in the same cluster and that the set of constrained variables in the core is a subset of the constrained variables in the cluster. A core is called trivial if all variables are unconstrained.

Achlioptas and Ricci-Tersenghi [ART06] proves that for $k \geq 9$, there exists a clause density $d_k < c_k^*$ such that a.s. every cluster in a uniformly random instance of k-SAT with n variables and $d_k n$ clauses has a non-trivial core and thus has constrained variables. However, experimental evidence of [MMW07] and evidence of [ART06] suggest that 3-SAT clusters a.s. have trivial cores. Therefore, [MMW07] suggests that the success of survey propagation on 3-SAT is partly due to luck, and [MMW07] makes a slight modification to survey propagation by changing the weights of the different messages in order to deal with the trivial cores.

2.5 XOR-SAT

XOR-SAT is one of the variations of SAT discussed by Schaeffer [Sch78]. In XOR-SAT, each clause is an exclusive-or of the literals, rather than a disjunction, and unlike SAT, XOR-SAT is in P because it can be solved by Gaussian elimination modulo 2. The statistical physics community studies XOR-SAT because it is exactly the p-spin model, which is considered to be the simplest non-trivial spin-glass-like model on random graphs at zero temperature [MRTZ03]. The importance of XOR-SAT and the p-spin model is that it is easier to analyze than k-SAT, and physicists can use XOR-SAT to prove predictions made by their non-rigorous techniques [CMMS03, MRTZ03], and these proofs lend justification to the predictions made by the physicists for the behavior of k-SAT.

2.5.1 The Satisfiability Threshold for XOR-SAT

3-XOR-SAT is one of the few SAT-like problems which has constant size domain and constraints for which the satisfiability threshold is known. The threshold occurs at clause density $.917935\ldots$, and Franz, Leone, Ricci-Tersenghi, and Zecchina [FLRTZ01] correctly predicts the threshold with the replica method before Dubois and Mandler [DM02] and Mézard, Ricci-Tersenghi, and Zecchina [MRTZ03] prove it.

Although XOR-SAT is in P, unlike the NP-complete, constant size domain and constraint problems for which the satisfiability threshold is known, the proof of the 3-XOR-SAT threshold is non-algorithmic. The proof uses the second moment method and does not rely in any way on the Gaussian elimination algorithm. Also, unlike these NP-complete problems, there is no known greedy algorithm that w.u.p.p. solves a random instance of 3-XOR-SAT drawn from close to the satisfiability threshold. This property as well as details in the threshold proofs support the notion that although XOR-SAT is in P and 1-in-k SAT is NP-complete, in the random model, 3-XOR-SAT behaves more like 3-SAT and 1-in-k SAT behaves more like 2-SAT.

The key property that makes the non-algorithmic proof of the satisfiability threshold for 3-XOR-SAT possible is that the constraint on each clause in XOR-SAT is *uniquely extendible.* That is, for each possible assignment to any $k-1$ variables of a clause of size k, there is a unique value for the kth variable that will satisfy the constraint. In particular, the proof makes crucial use of the fact that the constraints are both "at most one extendible" and "at least one extendible". The technique used to calculate the satisfiability threshold is to reduce the random formula to the *2-core*, the unique maximal subformula where each variable occurs in at least two clauses. Then, the first moment and second moment methods are used to give coinciding upper and lower bounds on the satisfiability threshold of the 2-core. Standard calculations translate the satisfiability threshold of the 2-core into a satisfiability threshold for 3-XOR-SAT.

It is the property that the constraint on each clause is "at least one extendible" that permits us to consider only the 2-core. Consider the following procedure to find the 2-core:

> **CORE:** While the formula has any variable that occurs in at most one clause, choose an arbitrary such variable and delete it along with any clause that contains it.

The order in which variables are chosen to be deleted is easily seen to be irrelevant in that it does not affect the final output of the procedure.

Lemma 2.1 *Let F be an instance of a CSP with one constraint per clause, and let the constraint on each clause have the property that if we assign values to all but one variable of the clause, then there is at least one possible assignment to the unassigned variable that will satisfy the constraint. F is satisfiable iff the 2-core of F is satisfiable.*

Proof. Clearly, if the 2-core of F is unsatisfiable then so is F. Assume that the 2-core of F is satisfiable. Consider running CORE on F, and suppose that the deleted variables are $x_1, x_2, ..., x_t$ in that order. Start with any satisfying assignment of the 2-core. Now restore the deleted variables in reverse order, i.e. $x_t, x_{t-1}, ..., x_1$, each time adding the variable along with the at most one clause that was deleted when the variable was deleted. Because the constraint on that clause is at least one extendible, there is a value that can be assigned to the variable that does not violate the constraint. This will result in a satisfying assignment for F. □

Let N be the number of satisfying assignments of a random formula F. Suppose that the d^n possible assignments are $\sigma_1, \ldots, \sigma_{d^n}$, and let N_i be the indicator variable that σ_i is a satisfying assignment. Then $N = N_1 + \cdots + N_{d^n}$ and $N^2 = \sum_{i,j} N_i N_j$. Therefore, $\mathbf{E}(N^2)$ is the sum over all pairs of assignments of the probability that both assignments satisfy the formula. Since the goal of the second moment method is to show that $\mathbf{E}(N^2) = \mathbf{E}(N)^2(1 + \mathrm{o}(1))$, for the second moment argument to work we must have the sum for $\mathbf{E}(N^2)$ dominated by the terms where the event one assignment is satisfying is independent of the other. In particular, this sum must not be dominated by the terms where the two assignments differ by the values of only a constant number of variables because the indicator variables for these assignments are not independent.

It is the property that the constraint on each clause is "at most one extendible" that enables the second moment method to give a tight bound on the satisfiability threshold for the 2-core of a random 3-XOR-SAT formula. The second moment method fails to do so for 3-SAT because of the "jackpot phenomena" that creates a strong correlation between satisfying assignments. Here is one simple way we get a jackpot where the existence of one satisfying assignment implies there are a.s. an exponential number of satisfying assignments. Let F be a random 3-SAT formula, conditional on the event that setting every variable to *true* satisfies the 2-core of F, and consider the 2-core. Standard arguments show that the formula will a.s. contain $\Omega(n)$ variables that occur in the formula as pure literals of the form $\overline{x}$. The assignment to each of these variables can be flipped independently to achieve another satisfying assignment. As a result, conditioning on having one satisfying assignment for the 2-core implies that there will a.s. be $2^{\Omega(n)}$ satisfying assignments. Other CSPs behave in a similar manner; more generally, rather than $\Omega(n)$ variables that can be flipped independently, we may a.s. have $\Omega(n)$ subsets of variables such that the assignment to the variables in one subset may be changed independently of the assignments in any other subset. This also yields $2^{\Omega(n)}$ satisfying assignments.

On the other hand, the "at most one extendible" property of an XOR-SAT clause prevents this type of jackpot in the 2-core of a random 3-XOR-SAT formula. When we change the assignment to a variable x in a satisfying assignment, we must change the assignment to exactly one other variable in every clause containing x. Since each variable lies in at least two clauses, we must repeat this process. The result will be a chain of variables that must be flipped, and the chain will contain at least one cycle. Now assume the random 3-XOR-SAT formula has a satisfying assignment and there are $\Omega(n)$ subsets of variables such that the assignment to the variables in one subset can be changed, independently of the other subsets, while maintaining satisfiability. A simple counting argument shows that for at least $\frac{n}{2}$ of the sets, the number of variables in the set is bounded by some constant. This implies the underlying random hypergraph has $\Omega(n)$ small cycles, and such a configuration a.s. does not occur in a random hypergraph with a linear number of hyperedges. This idea can be summarized in the following remark.

Remark 2.2 *Let F be a uniformly random instance of a CSP with one constraint per clause, let each constraint have the property that if we assign values to all but one variable of the clause, then there is at most one possible assignment to the unassigned variable that will satisfy the constraint, and suppose we have a satisfying assignment for F. F does not contain $\Omega(n)$ subsets of variables such that we can produce another satisfying assignment by changing the assignments for any one subset of variables independently of the assignments to the remaining subsets of variables.*

As a result, there is at least one type of jackpot phenomena that occurs in random 3-SAT and that does not occur in random 3-XOR-SAT. Note that reducing the formula to the 2-core is critical to the success of the second moment method. In a uniformly random hypergraph with a linear number of hyperedges and an edge density large enough that there a.s. is a connected component of $\Omega(n)$ variables, there are a.s. $\Omega(n)$ clauses such that each contains one variable that is in more than one clause of this giant component and two variables that are in no other clause. If we have a satisfying assignment for the formula, we could choose one such clause and flip the assignments to the two variables that are in only that clause, and we would achieve another satisfying assignment. As above, conditioning on having one satisfying assignment implies that there will a.s. be $2^{\Omega(n)}$ satisfying assignments for this giant component.

This proof, of course, requires a good understanding of the 2-core of random 3-XOR-SAT. In fact, what we need to understand is the 2-core of the underlying hypergraph. Both Dubois and Mandler [DM02] and Mézard, Ricci-Tersenghi, and Zecchina [MRTZ03] give the threshold for the appearance of the 2-core in 3-XOR-SAT as $.818469\ldots$, and this threshold is easily calculated for any k-XOR-SAT, or any k-uniform hypergraph for that matter, using a theorem of either Molloy [Mol05] or Cain and Wormald [CW06] on cores of random hypergraphs that extends the results of Pittel, Spencer, and Wormald [PSW96] for cores of random graphs. There, arguments also yield the number of vertices and edges in the 2-core.

2.5.2 Algorithm Behavior on XOR-SAT

XOR-SAT is in P because it can be solved by Gaussian elimination; however, no greedy algorithm is known to work a.s. on a random instance of XOR-SAT. In fact, no greedy algorithm is known to work at densities above $.818469\ldots$, the appearance of the 2-core. In Section 5.2, we prove that unit clause fails at densities above $\frac{2}{3}$, and a variation of generalized unit clause fails at densities above $.75087\ldots$. The other algorithms used to increase the lower bound of the conjectured satisfiability threshold of 3-SAT, discussed in Section 2.1.1, do not apply to XOR-SAT because they exploit an asymmetry in the literals of SAT. For example, because any true literal satisfies a clause, if a variable occurs more often as a positive rather than as a negative literal, setting that variable to *true* will satisfy more literals than setting it to *false*. This dichotomy does not hold for XOR-SAT.

For local search, Cocco, Monasson, Montanari, and Semerjian [CMMS03] shows that gradient descent, and by extension GSAT, can get trapped in a simple configuration that a.s. appears in a random formula with a linear number of clauses. Non-rigorous analysis of the pure random walk algorithm on k-XOR-SAT by Semerjian and Monasson [SM03] suggests that the algorithm a.s. finds assignments in satisfiable random instances in polynomial time up to the density $\frac{1}{k}$.

It is straightforward to see that the standard belief propagation algorithm on XOR-SAT is equivalent to unit clause. By the uniquely extendible property, a literal in a clause has an equal probability of being assigned *true* or *false* until all the other literals in the clause get assigned a value. Likewise, survey propagation does poorly on XOR-SAT. Mézard, Ricci-Tersenghi, and Zecchina [MRTZ03] notes that the threshold for the appearance of the 2-core corresponds to the threshold for the appearance of multiple clusters in the solution space. By the same argument that the 2-core of XOR-SAT does not suffer from the jackpot phenomena, we can argue that the fixed variables in the core assignment corresponding to each cluster include all the variables that exist in the 2-core. In an investigation of this phenomenon, Mora and Mézard [MM06] uses the cavity method to estimate the size of and distance between solution clusters for XOR-SAT.

Chapter 3

Uniquely Extendible Constraint Satisfaction Problems

3.1 Defining Uniquely Extendible CSPs

In this chapter, we will define a new class of constraint satisfaction problems with constant sized domain and constraints. We focus on a particular subclass that we show to be NP-complete. We also prove that, for each problem in our subclass, a uniformly random instance with n variables and cn clauses, $c > 0$, a.s. has exponential resolution complexity. In the next chapter, we prove that, for each problem in our subclass, the random model has an exact satisfiability threshold, subject to the Maximum Hypothesis. These problems are the first NP-complete problems that are proven to have all these characteristics. Prior to this thesis, Xu and Li [XL00, XL06] defines two random CSP models that have exact satisfiability thresholds, and [XL06] proves that both these models contain many problems that a.s. have exponential tree resolution complexity. In addition, [XL06] notes that the model of [FW01], which also has an exact satisfiability threshold, contains problems that a.s. have exponential resolution complexity. However, the model of [FW01] has a constraint size that grows with n, and the models of [XL00] have a domain size that grows with n. In these cases, the satisfiability threshold occurs when the number of clauses is superlinear in n, and the structure of such random formulae is very different from the structure of random formulae with a linear number of clauses.

The inspiration for the problem defined in this chapter is from the proof of the satisfiability threshold for random 3-XOR-SAT. The proof does not depend on the Gaussian elimination algorithm, and this suggests that the computational complexity of 3-XOR-SAT is not the property that yields the satisfiability threshold. Given the proof for 3-XOR-SAT, a natural question to ask is whether the same techniques can be applied to prove the exact satisfiability threshold for an NP-complete problem. Our goal is to generalize XOR-SAT to a NP-complete problem while maintaining the properties that make XOR-SAT amenable to calculating the exact satisfiability threshold.

Recall, from Section 2.5.1, that the key property of 3-XOR-SAT that permits the technique of [DM02] to determine its precise threshold of satisfiability is that each constraint is *uniquely extendible.* That is, for each possible assignment to any $k-1$ variables of a clause, there is a unique legal value for the kth variable that satisfies the constraint on the clause. The first step to finding the desired NP-complete problem is to generalize XOR-SAT to a universal uniquely extendible constraint satisfaction problem that we denote UE-CSP.

Definition 3.1 (Uniquely Extendible Constraint) *A uniquely extendible constraint on a canonical ordered set of variables restricts the values we can assign to the variables as follows. For any subset of $k-1$ of these variables and for any assignment to those $k-1$ variables, there is exactly one value we can give to the unassigned variable such that the constraint will permit that tuple.*

Definition 3.2 (UE-CSP) *A UE-CSP instance is a constraint satisfaction problem where every variable is assigned a value from the same domain and every constraint* *is uniquely extendible.*

Definition 3.3 (k-UE-CSP) *An instance of k-UE-CSP is an instance of UE-CSP where we restrict every clause to have size k.*

Definition 3.4 ((k, d)-UE-CSP) *An instance of (k, d)-UE-CSP is an instance of k-UE-CSP where we specify that the domain size is d.*

In k-XOR-SAT, each clause is a parity constraint on the values of the variables in the clause. The parity of the variables assigned to *true* (or 1) is the opposite parity of the negative literals in the clause. For example, a clause with two negative literals requires an odd number of variables to be assigned *true*. Similarly, a straightforward induction on k yields that there are only two different uniquely extendible constraints of size k with $d = 2$. This observation implies that k-XOR-SAT is exactly $(k, 2)$-UE-CSP.

3.2 Links to Combinatoric Structures

It turns out that uniquely extendible constraints correspond to well studied combinatorial structures. For $k = 2$, each constraint corresponds to a permutation on d objects. For $k = 3$, uniquely extendible constraints correspond to quasigroups and Latin squares, and for $k > 3$, the constraints correspond to the somewhat less studied natural generalization of quasigroups and Latin squares to higher dimensions.

A *Latin square* is a $d \times d$ matrix with elements from $\{0, \ldots, d-1\}$ such that no element appears twice in a row or column. A *quasigroup* is a set Q with a binary operation defined in Q such that for any two (not necessarily distinct) elements a, b of Q, the equations $ax = b$ and $ya = b$ each have exactly one solution. This is equivalent to stating that $ab = c$ defines a triple of values of Q and for any assignment of elements of Q to two of the values a, b, c, there is a unique element of Q to assign to the third. Equivalently, a quasigroup is an algebraic group without the associative law. A well known result is that the multiplication table of a quasigroup is a Latin square, and every Latin square is the multiplication table of a quasigroup.

Two subsets of quasigroups are used in the proofs of this chapter. Using terminology from [DK74], a *totally symmetric* quasigroup has the property that if $xy = z$, then we must have all the symmetric relations:

$$xy = z \quad yx = z \quad xz = y \quad zx = y \quad yz = x \quad zy = x.$$

A *medial* quasigroup has the property that

$$(ab)(cd) = (ac)(bd).$$

Finally, two quasigroups Q_1, Q_2 are called *isomorphic* if Q_2 can be formed by permuting the values of Q_1.

As a result, there are different ways to represent a constraint. Table 3.1 demonstrates each of these representations. We can think of a constraint as a list of ordered tuples where each tuple represents a legal assignment to the variables of a clause. If the constraint is totally symmetric, then we can compact the representation by using unordered tuples. Finally, we can think of the constraint on clause $(v_1, \ldots, v_k)$ as the mathematical operation $v_1 \cdots v_{k-1} = v_k$. For $k = 3$, the permitted values for the variables can be represented by the Latin square that corresponds to that constraint's multiplication table.

3.3 Complexity Results

In Section 3.3.1, we prove that (k, d)-UE-CSP is in P whenever $k \leq 2$ or $d \leq 3$. In Section 3.3.2, we prove that $(3, d)$-UE-CSP is NP-complete for any $d \geq 4$. We conjecture that (k, d)-UE-CSP is NP-complete for all $k \geq 3$ and $d \geq 4$, but we cannot prove this. Section 3.3.2.3 discusses the open conjectures.

0	0	0
0	1	2
0	2	1
0	3	3
1	0	2
1	1	3
1	2	0
1	3	1
2	0	1
2	1	0
2	2	3
2	3	2
3	0	3
3	1	1
3	2	2
3	3	0

(0, 0, 0)
(0, 1, 2)
(0, 3, 3)
(1, 1, 3)
(2, 2, 3)

	0	1	2	3
0	0	2	1	3
1	2	3	0	1
2	1	0	3	2
3	3	1	2	0

Table 3.1: Three representations of the same constraint with $k = 3$ and $d = 4$. The left example is a list of all legal ordered triples of three values. The center example is a list of all legal unordered triples of three values. This representation is possible only if the constraint is totally symmetric. The right example is as a multiplication table. (x, y, z) is a legal tuple for the constraint if the value at the xth row and yth column is z.

3.3.1 Polynomial Time Variations

In this section, we prove that (k, d)-UE-CSP $\in$ P whenever $k \leq 2$ or $d \leq 3$.

Theorem 3.5 1-*UE-CSP is in P.*

Proof. Trivial. □

Theorem 3.6 $(k, 2)$-*UE-CSP is in P for all* $k \geq 2$.

Proof. Recall from Section 3.1 that $(k, 2)$-UE-CSP is exactly k-XOR-SAT, and it is well known that XOR-SAT is in P because every uniquely extendible constraint must be a parity constraint, and so the problem reduces to solving a system of linear equations modulo 2. □

Theorem 3.7 $(2, d)$-*UE-CSP is in P for all* $d \geq 2$.

Proof. By the nature of $(2, d)$-UE-CSP, setting a variable v forces the value for each variable that shares a clause with v. Therefore, assigning a value to v will force the value for every other variable that, in the underlying graph of the formula, is in the connected component containing v. Given an instance F of $(2, d)$-UE-CSP with n variables and m clauses, consider the following algorithm to find a satisfying assignment:

> Choose a variable v and assign it a value from $\{0, \ldots, d-1\}$. Then traverse the connected component containing v in a systematic manner (for example, in a breadth first traversal) setting each variable to its forced value. If every clause of this component is satisfied, repeat the procedure with the next connected component. Otherwise, try a new value for v.

Since each variable may be set at most d different times and each clause will be tested at most d times for validity, the running time of the algorithm is $O(d(n + m))$. □

Theorem 3.8 $(k, 3)$-*UE-CSP is in P for all* $k \geq 2$.

The proof will rely on the following two lemmas. In the next lemma we will consider a constraint to be a list of acceptable tuples.

Lemma 3.9 *The number of constraints of* $(k, 3)$*-UE-CSP is twice the number of possible constraints of* $(k-1, 3)$*-UE-CSP,* $k \geq 3$.

Proof. We begin with a general observation on generating uniquely extendible constraints of size k from uniquely extendible constraints of size $k-1$. Let C be a constraint of (k, d)-UE-CSP. For each $i \in \{0, \ldots, d-1\}$, let C_i be the set of tuples defined by

$$C_i = \{(v_1, \ldots, v_{k-1}) \mid (v_1, \ldots, v_k) \in C \text{ and } v_k = i\}.$$

Note that C_i is a constraint of $(k-1, d)$-UE-CSP because, in constraint C, for any setting of $v_1, \ldots, v_{k-2}$, there is a unique value for v_{k-1} that satisfies the constraint when $v_k = i$. With a slight abuse of notation, we can write $C = \{(0, C_0), \ldots, (d-1, C_{d-1})\}$ for $C_0, \ldots, C_{d-1}$ constraints of $(k-1, d)$-UE-CSP where

$$(i, C_i) = \{(v_1, \ldots, v_k) \mid (v_1, \ldots, v_{k-1}) \in C_i \text{ and } v_k = i\}.$$

Also note that each pair of constraints in the set $\{C_0, \ldots, C_{d-1}\}$ cannot have the same acceptable tuple. If two constraints C_i, C_j shared an acceptable tuple $(a_1, \ldots, a_{k-1})$ then $(a_1, \ldots, a_{k-1}, i)$ and $(a_1, \ldots, a_{k-1}, j)$ would both be acceptable tuples of C, and then C would not be uniquely extendible.

Call a pair of uniquely extendible constraints of size k *compatible* if the constraints do not contain the same acceptable tuple, and a set is called pairwise compatible if every pair of constraints in that set is compatible. Given a maximal set $\mathcal{C}$ of pairwise compatible uniquely extendible constraints of size $k-1$, we can form a uniquely extendible constraint C of size k by taking any d constraints from $\mathcal{C}$ and piecing them together in the following manner:

$$C = \{(0, C_{i_0}), \ldots, (d-1, C_{i_{d-1}})\} \text{ with } C_{i_0}, \ldots, C_{i_{d-1}} \in \mathcal{C}.$$

A maximal set of pairwise compatible uniquely extendible constraints contains at least d constraints, and for any uniquely extendible constraint C there is a simple construction for creating a set of d pairwise compatible constraints containing C. Given

$$C = \{(0, C_{i_0}), \ldots, (d-1, C_{i_{d-1}})\},$$

create $d-1$ uniquely extendible constraints by rotating the C_i's. That is,

$$\begin{aligned} C &= \{(0, C_{i_0}), (1, C_{i_1}), \ldots, (d-1, C_{i_{d-1}})\} \\ C_1 &= \{(0, C_{i_1}), (1, C_{i_2}), \ldots, (d-1, C_{i_0})\} \\ C_2 &= \{(0, C_{i_2}), (1, C_{i_3}), \ldots, (d-1, C_{i_1})\} \\ &\vdots \\ C_{d_1} &= \{(0, C_{i_{d-1}}), (1, C_{i_0}), \ldots, (d-1, C_{i_{d-2}})\}, \end{aligned}$$

and it is straightforward to verify that the set $\{C, C_1, C_2, \ldots, C_{d-1}\}$ is pairwise compatible.

The key observation for Lemma 3.9 is that for every constraint C of $(k, 3)$-UE-CSP, $k \geq 2$, there are exactly two constraints compatible with C and these three constraints form a pairwise compatible set. We can prove this observation by induction. For $k = 2$ and constraint $\{(0, \alpha), (1, \beta), (2, \gamma)\}$, with (α, β, γ) a permutation of $\{0, 1, 2\}$, the pairwise compatible set of constraints containing $\{(0, \alpha), (1, \beta), (2, \gamma)\}$ is

$$\{\{(0, \alpha), (1, \beta), (2, \gamma)\}, \{(0, \beta), (1, \gamma), (2, \alpha)\}, \{(0, \gamma), (1, \alpha), (2, \beta)\}\}.$$

For arbitrary k, let C be a constraint of size k. C is of the form $\{(0, C_0), (1, C_1), (2, C_2)\}$ where $\{C_0, C_1, C_2\}$ are pairwise compatible constraints of size $k-1$. By the induction hypothesis, this set contains the only compatible constraints for C_0, C_1, and C_2. As a result, the set of pairwise compatible constraints containing C is

$$\{C, \{(0, C_1), (1, C_2), (2, C_0)\}, \{(0, C_3), (1, C_0), (2, C_1)\}\}.$$

To complete the proof, note that for each constraint C' of size $k-1$ we can build two constraints of size k using C' for C_0. Namely,

$$\{(0,C'),(1,C'_1),(2,C'_2)\} \text{ and } \{(0,C'),(1,C'_2),(2,C'_1)\}$$

where C'_1 and C'_2 are the compatible constraints of C'. □

Lemma 3.10 *For $k \geq 2$ and p a prime, let the equation*

$$x_k + a_{k-1}x_{k-1} + \cdots + a_1x_1 + a_0 = 0 \tag{3.1}$$

define a constraint on $x_1, \ldots, x_k$ such that $a_0 \in \{0, \ldots, p-1\}$, $a_1, \ldots, a_{k-1} \in \{1, \ldots, p-1\}$, and all operations are modulo p. Such a constraint is uniquely extendible and each choice of $a_0, \ldots, a_{k-1}$ defines a unique such constraint.

Proof. The constraint is obviously extendible. To show it is uniquely extendible assume, w.l.o.g., $(x_1, x_2, \ldots, x_k)$ and $(y_1, x_2, \ldots, x_k)$ are both solutions to (3.1). Straightforward algebraic manipulation shows that $x_1 = y_1$.

Now, assume $\{a_0, \ldots, a_{k-1}\}$ and $\{b_0, \ldots, b_{k-1}\}$ are two different sets that define the same constraint on $\{x_1, \ldots, x_k\}$. Since the constraints are uniquely extendible, for any setting of $x_{k-1}, \ldots, x_1$, there is a unique setting of x_k such that

$$x_k + a_{k-1}x_{k-1} + \cdots + a_1x_1 + a_0 = 0 \text{ and}$$
$$x_k + b_{k-1}x_{k-1} + \cdots + b_1x_1 + a_0 = 0.$$

Thus, we must have

$$a_{k-1}x_{n-1} + \ldots + a_1x_1 + a_0 = b_{k-1}x_{n-1} + \ldots + b_1x_1 + b_0 \tag{3.2}$$

for all choices of $x_{k-1}, \ldots, x_1 \in \{0, \ldots, p-1\}$. In particular, let every $x_i = 0$, and we have $a_0 = b_0$. Now let x_1 be the only non-zero variable. Since p is prime, every element of $\{0, \ldots, p-1\}$ has a unique inverse modulo p, we get $a_1 = b_1$. Repeating this process shows that $a_i = b_i$ for all $i = 0, \ldots, k-1$. □

Note that if p were not prime, the argument above fails because there will exist choices of $a_0, \ldots, a_{k-1}$ for which the constraint defined by (3.1) is not extendible. In particular fix $x_2, \ldots, x_k$, if $x_k + a_{k-1}x_{k-1} + \cdots + a_2x_2 + a_0$ is a multiple of $\gcd(a_1, p)$ then there exists $\gcd(a_1, p)$ values for x_1 that satisfy (3.1) otherwise there are no values for x_1 that satisfy (3.1).

Proof. [Theorem 3.8] For $(2,3)$-UE-CSP the number of possible constraints is the number of permutations on 3 elements. From this observation and Lemma 3.9, there are $6\left(2^{k-2}\right)$ uniquely extendible constraints for $(k,3)$-UE-CSP. Likewise, there are $3\left(2^{k-1}\right)$ equations of the form (3.1), and from Lemma 3.10 each equation uniquely defines such a constraint. As a result, there is a one-to-one correspondence between the linear equations and the uniquely extendible constraints. Therefore, given an instance F of $(k,3)$-UE-CSP, we can replace each clause and its constraint by a corresponding linear equation over the variables of the clause. Then we find a solution to F by solving the system of linear equations modulo 3 using Gaussian elimination. □

The proof of Theorem 3.8 fails to extend to $d > 3$ because, as d increases, the number of possible constraints of (k,d)-UE-CSP grows faster than the number of equations of the form (3.1).

3.3.2 NP-Complete Variations

3.3.2.1 The Basic Proof for NP-Completeness of $(3,4)$-UE-CSP

Theorem 3.11 $(3,4)$-*UE-CSP is NP-complete.*

This theorem appears in [CM04]. The proof is a straightforward reduction from 3-coloring a graph. Recall that in a general CSP, multiple constraints may be applied to a clause, and this proof makes use of that property. However, this proof forms the basis of a more complicated extension, given in Section 3.3.2.2, to show that $(3,d)$-UE-CSP is NP-complete for $d \geq 4$ even when restricted to inputs where there is only one constraint

Constraint 1			Constraint 2			Constraint 3		
u	v	e	u	v	e	u	v	e
0	0	0	0	0	3	0	0	3
0	1	2	0	1	2	0	1	2
0	2	1	0	2	1	0	2	1
0	3	3	0	3	0	0	3	0
1	0	2	1	0	2	1	0	2
1	1	3	1	1	1	1	1	3
1	2	0	1	2	0	1	2	0
1	3	1	1	3	3	1	3	1
2	0	1	2	0	1	2	0	1
2	1	0	2	1	0	2	1	0
2	2	3	2	2	3	2	2	2
2	3	2	2	3	2	2	3	3
3	0	3	3	0	0	3	0	0
3	1	1	3	1	3	3	1	1
3	2	2	3	2	2	3	2	3
3	3	0	3	3	1	3	3	2

Table 3.2: The constraints used in the proof of Theorem 3.11. Each row of a constraint lists the ordered triples of values that the constraint permits to be assigned to the 3 variables.

per clause and no two clauses intersect on more than one variable.

Proof. Clearly the problem is in NP. We will prove it is NP-hard by showing 3-COLOR $\leq_p$ $(3,4)$-UE-CSP. For 3-COLOR, we are given a graph G with n vertices and m edges, and we wish to color G with three colors such that any two vertices joined by an edge cannot receive the same color. Now given G, we will create an instance of $(3,4)$-UE-CSP with $n+m$ variables, m clauses, and we apply 3 constraints to each clause so that G is 3-colorable if and only if there is a valid assignment to the variables of the CSP. We will create one variable for each vertex of G and one variable for each edge of G. For each edge $e = uv$ of G, we will create a clause containing the three variables corresponding to u, v, and e, and we will apply 3 uniquely extendible constraints to this clause (u, v, e). These constraints are listed in Table 3.2.

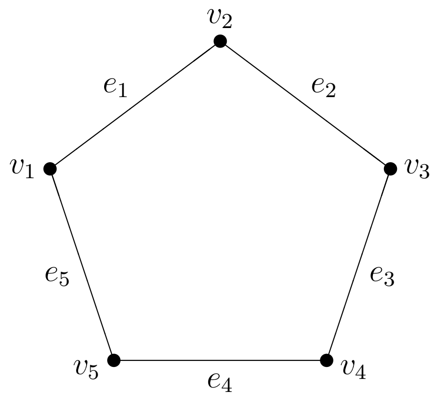

Figure 3.1: An example graph for the proof of Theorem 3.11. We convert the 3-COLOR problem on this graph to a $(3,4)$-UE-CSP problem by creating the clauses $\{(v_1,v_2,e_1),(v_2,v_3,e_2),(v_3,v_4,e_3),(v_4,v_5,e_4),(v_5,v_1,e_5)\}$ and placing the three constraints of Table 3.2 onto each clause.

Note that the variables u, v, and e may be set any permutation of $\{0,1,2\}$. However, no variable can receive a value of 3 without violating one of the constraints. Likewise, u and v cannot be assigned the same value without violating the constraints. If we let the colors for G be $\{0,1,2\}$, the proof follows. □

For an example, consider the graph of Figure 3.1 with vertices $v_1,\ldots,v_5$ and edges labeled $e_1,\ldots,e_5$. We convert this graph to an instance of $(3,4)$-UE-CSP with variables $\{v_1,\ldots,v_5,e_1,\ldots,e_5\}$. For edge e_1 connecting vertices v_1 and v_2, we create the clause (v_1,v_2,e_1). Similarly, we create a clause for each of the other edges giving the following set of clauses: $\{(v_1,v_2,e_1),(v_2,v_3,e_2),(v_3,v_4,e_3),(v_4,v_5,e_4),(v_5,v_1,e_5)\}$. Then on each clause, we apply all three constraints from Table 3.2.

3.3.2.2 Extending the NP-Completeness Result

In this section we extend the proof of NP-completeness to the case for $d \geq 4$ and with only one constraint applied to each clause. In addition, we will use a restricted model where no pair of variables appears in more than one clause. It is important to extend the NP-completeness proof because the reduction of Theorem 3.11 is unsatisfactory for our purposes. The random model described in Section 3.4 has one constraint per clause; however, Theorem 3.11 only proves the problem is NP-complete if we have multiple constraints per clause. At the least, we should extend the model to the case where each clause has a single constraint. In addition, we should restrict the model further because in a uniformly random instance with the number of clauses linear in the number of variables, the expected number of clauses that share a pair of variables is $\Theta(1)$. Since we expect there are at most a constant number of clauses that share a pair of variables, it is not hard to show that w.u.p.p. a uniformly random formula will have no clauses sharing a pair of variables.

As a first step, we restrict the problem to use only constraints that are totally symmetric and medial. Such constraints are defined in Section 3.2.

Lemma 3.12 *$(3, d)$-UE-CSP is NP-complete for $d \geq 4$ even if every constraint must be totally symmetric and medial.*

Proof. The key step of the proof of Theorem 3.11 for the reduction from 3-COLOR is that we can find a set of constraints for $(3, 4)$-UE-CSP such that every permutation of $\{0, 1, 2\}$ is a legal tuple in each constraint and that no other tuple exists in every constraint. To prove Lemma 3.12, we perform the same step by finding a set of constraints that permit every permutation of $\{0, 1, 2\}$ but forbid every other tuple.

Both [BPZ78] and [Bru44] discuss techniques for generating totally symmetric quasigroups. Let $C_\emptyset$ be the constraint that consists of every triple of values (a, b, c) such that

$$(a + b + c - 3) \mod d = 0.$$

From the associative and symmetric properties of addition, it is straightforward to verify that the constraint is totally symmetric and medial. Note that in $C_\emptyset$, every permutation of $\{0, 1, 2\}$ is a legal tuple. Let σ be a permutation of $\{0, \ldots, d - 1\}$. Given constraint $C_\emptyset$, define C_σ to be a constraint isomorphic to $C_\emptyset$ formed by permuting the values of $C_\emptyset$

by the permutation σ. For example, if we represent the constraint as a Latin square, we permute the constraint by σ by permuting the rows, columns, and elements of the table by σ. If we represent the constraint as a list of tuples, we apply the permutation σ to each element in every tuple of the list.

We require two simple observations. First, every constraint isomorphic to $C_\emptyset$ is also a totally symmetric and medial constraint. Second, if

$$(\alpha+\beta+\gamma) \mod d \;\neq\; (\alpha'+\beta'+\gamma') \mod d$$

then $\{\alpha,\beta,\gamma\}$ and $\{\alpha',\beta',\gamma'\}$ are different sets of values.

Define

$$\phi(a,b,c) = (a+b+c-3) \mod d.$$

Given a permutation σ of $\{0,\ldots,d-1\}$, let

$$\phi_\sigma(a,b,c) = \phi(\sigma(a),\sigma(b),\sigma(c)).$$

Note that a triple (a,b,c) belongs to $C_\emptyset$ if and only if $\phi(a,b,c)=0$.

To extend the NP-completeness proof of Theorem 3.11, we apply a finite set $\mathcal{C}$ of totally symmetric, medial constraints to each clause. The set $\mathcal{C}$ will have the following properties. For every constraint $C \in \mathcal{C}$, $(0,1,2) \in C$; for any triple of values (a,b,c) that is not $(0,1,2)$, there is a constraint $C \in \mathcal{C}$ such that $(a,b,c) \notin C$; and the cardinality of $\mathcal{C}$ does not depend on the number of variables and clauses in the input formula.

If $d=4$, let $\mathcal{C} = \{C_\emptyset, C_{(0,1,2)}, C_{(0,2,1)}\}$, and $\mathcal{C}$ is exactly the set used in Theorem 3.11.

If $d>4, d\neq 6,9$, let $\mathcal{C} = \{C_\emptyset, C_{(0,1,2)}, C_{(3,\ldots,d)}\}$. Note that every permutation of $\{0,1,2\}$ exists in both $C_{(0,1,2)}$ and $C_{(3,\ldots,d)}$. Let X be the set of all tuples that contain an element of $\{0,1,2\}$ but are not a permutation of $\{0,1,2\}$. We now prove that each such tuple is absent in either $C_\emptyset$ or $C_{(0,1,2)}$. Let $A = X \cap C_\emptyset$.

$$A = \{(0,0,3),(1,1,1),(2,2,d-1),(0,r,d+3-r),(1,s,d+2-s),(2,s,d+1-s)\}$$

where $r > 3, s \geq 3$. Let

$$B = \{(\sigma(a), \sigma(b), \sigma(c)) \mid (a, b, c) \in A\}$$

where $\sigma = (0, 1, 2)$. Note that $B = X \cap C_{(0,1,2)}$.

To show $X \cap C_{\emptyset} \cap C_{(0,1,2)} = \emptyset$, it is sufficient to show $A \cap B = \emptyset$. Since for each tuple $\tau \in A$, $\phi(\tau) = 0$, if for every tuple $\tau \in B$, $\phi(\tau) \neq 0$, then $A \cap B = \emptyset$.

$$\begin{array}{lclcrl}
\phi_{(0,1,2)}(0,0,3) & = & \phi(1,1,3) & = & 2 & \mod d \\
\phi_{(0,1,2)}(1,1,1) & = & \phi(2,2,2) & = & 3 & \mod d \\
\phi_{(0,1,2)}(2,2,d-1) & = & \phi(0,0,d-1) & = & -4 & \mod d \\
\phi_{(0,1,2)}(0,r,d+3-r) & = & \phi(1,r,d+3-r) & = & 1 & \mod d \\
\phi_{(0,1,2)}(1,s,d+2-s) & = & \phi(2,s,d+2-s) & = & 1 & \mod d \\
\phi_{(0,1,2)}(2,s,d+1-s) & = & \phi(0,s,d+1-s) & = & -2 & \mod d
\end{array}$$

where $r > 3$, $s \geq 3$. As a result, every triple of $C_{\emptyset}$ that contains an element of $\{0, 1, 2\}$ except for the triple $(0, 1, 2)$ does not exist in $C_{(0,1,2)}$ assuming $d > 4$.

Using the same reasoning, consider every triple of $C_{\emptyset}$ that does not contain an element of $\{0, 1, 2, d-1\}$, and we will show that such a triple does not exist in $C_{(3,\ldots,d-1)}$. Following similar steps to the above computation yields

$$\phi_{(3,\ldots,d-1)}(a, b, c) = 3 \mod d$$

for $a, b, c \in \{3, \ldots, d-2\}$.

Finally, assume at least one element of the triple is $d - 1$. The possible triples of $C_{\emptyset}$ are:

$$\{(d-1, d-1, 5), (d-1, r, d+4-r)\}$$

where $r > 4$. $(d-1, d-1, d-1)$ is not a valid tuple of $C_{\emptyset}$ since $d \neq 6$.

$$\begin{array}{lcrl}
\phi_{(3,\ldots,d-1)}(d-1,d-1,5) & = & 9 & \mod d \\
\phi_{(3,\ldots,d-1)}(d-1,r,d+4-r) & = & 6 & \mod d
\end{array}$$

Therefore, every triple of $C_\emptyset$ that does not contain an element of $\{0, 1, 2\}$ does not exist in $C_{(3,\ldots,d-1)}$ assuming $d \neq 6, 9$.

If $d = 6$, let $\mathcal{C} = \{C_\emptyset, C_{(0,1,2)}, C_{(0,3,1),(2,4,5)}\}$. Table 3.3 lists the triples in each constraint. It is clear that only the triples that are a permutation of $\{0, 1, 2\}$ occur in all constraints.

If $d = 9$, let $\mathcal{C} = \{C_\emptyset, C_{(0,1,2)}, C_{(3,5,7),(4,6,8)}\}$. Table 3.4 lists the triples in each constraint. Likewise, only the permutations of $\{0, 1, 2\}$ occur in all constraints.

The rest of the proof follows exactly the same steps as the proof of Theorem 3.11 using $\mathcal{C}$ as the set of constraints to apply to each edge. □

We need one more step before we can prove the main theorem of this section. The following lemma describes a construction with which we can replace a constraint on a clause by a gadget containing a constant number of variables and clauses and such that the satisfiability of the formula is unchanged.

Construction 3.13 *Given a $(3, d)$-UE-CSP formula F, let (a, b, c) be a clause of F and let C be a constraint on (a, b, c). Add six new variables*

$$x_1, x_2, x_3, x_4, x_5, x_6$$

and five new clauses

$$(a, x_1, x_2), (b, x_3, x_4), (c, x_5, x_6), (x_1, x_3, x_5), (x_2, x_4, x_6)$$

to F. Remove C from (a, b, c) and add C to each new clause.

Lemma 3.14 *Given a $(3, d)$-UE-CSP formula F, we can form a new formula F' by replacing any medial constraint C on a clause c using Construction 3.13. The substitution will be such that no clause added by the construction will share a pair of variables with any other clause of F and such that F is satisfiable if and only if F' is satisfiable. Moreover, any assignment that satisfies F can be extended to an assignment satisfying F' by finding*

$C_\emptyset$	$C_{(0,1,2)}$	$C_{(0,3,1),(2,4,5)}$
(0, 0, 3)	(0, 0, 5)	(0, 0, 0)
(0, 1, 2)	(0, 1, 2)	(0, 1, 2)
(0, 4, 5)	(0, 3, 4)	(0, 3, 4)
(1, 1, 1)	(1, 1, 3)	(0, 5, 5)
(1, 3, 5)	(1, 4, 5)	(1, 1, 1)
(1, 4, 4)	(2, 2, 2)	(1, 3, 3)
(2, 2, 5)	(2, 3, 5)	(1, 4, 5)
(2, 3, 4)	(2, 4, 4)	(2, 2, 2)
(3, 3, 3)	(3, 3, 3)	(2, 3, 5)
(5, 5, 5)	(5, 5, 5)	(2, 4, 4)

Table 3.3: The constraints used for the case $d = 6$ in Lemma 3.12. Each row of a constraint lists the unordered triples of values that the constraint permits to be assigned to a clause.

$C_\emptyset$	$C_{(0,1,2)}$	$C_{(3,5,7),(4,6,8)}$
(0, 0, 3)	(0, 0, 8)	(0, 0, 5)
(0, 1, 2)	(0, 1, 2)	(0, 1, 2)
(0, 4, 8)	(0, 3, 7)	(0, 3, 7)
(0, 5, 7)	(0, 4, 6)	(0, 4, 6)
(0, 6, 6)	(0, 5, 5)	(0, 8, 8)
(1, 1, 1)	(1, 1, 3)	(1, 1, 1)
(1, 3, 8)	(1, 4, 8)	(1, 3, 6)
(1, 4, 7)	(1, 5, 7)	(1, 4, 5)
(1, 5, 6)	(1, 6, 6)	(1, 7, 8)
(2, 2, 8)	(2, 2, 2)	(2, 2, 4)
(2, 3, 7)	(2, 3, 8)	(2, 3, 5)
(2, 4, 6)	(2, 4, 7)	(2, 6, 8)
(2, 5, 5)	(2, 5, 6)	(2, 7, 7)
(3, 3, 6)	(3, 3, 6)	(3, 3, 3)
(3, 4, 5)	(3, 4, 5)	(3, 4, 8)
(4, 4, 4)	(4, 4, 4)	(4, 4, 7)
(5, 8, 8)	(5, 8, 8)	(5, 5, 8)
(6, 7, 8)	(6, 7, 8)	(5, 6, 7)
(7, 7, 7)	(7, 7, 7)	(6, 6, 6)

Table 3.4: The constraints used for the case $d = 9$ in Lemma 3.12. Each row of a constraint lists the unordered triples of values that the constraint permits to be assigned to a clause.

appropriate values for the new variables, and any assignment satisfying F', restricted to the variables of F, will be a satisfying assignment for F.

Proof. Given a formula F, let C be a medial constraint on the clause (a, b, c). Create a new formula F' by Construction 3.13 on clause (a, b, c) and constraint C. Since each constraint is a quasigroup, the original constraint C on (a, b, c) is equivalent to the statement $ab = c$ where the operation is the quasigroup operator. This statement continues to hold in F'.

$$\begin{aligned} c &= x_5x_6 \\ &= (x_1x_3)(x_2x_4) \\ &= (x_1x_2)(x_3x_4) && \text{by medial property} \\ &= ab. \end{aligned}$$

Therefore, if we have a solution to F', we can form a solution to F by giving every variable in F the same value as in the solution to F'. Similarly, if we have a solution to F, we can form a solution to F' by assigning all original variables the same value as in the solution to F, assigning x_1 and x_3 arbitrary values, and assigning to x_2, x_4, x_5, and x_6 the values forced by the values of a, b, x_1, and x_3. The following calculation shows that $c = x_5x_6$ and thus every constraint is satisfied.

$$\begin{aligned} x_5x_6 &= (x_1x_3)(x_2x_4) \\ &= (x_1x_2)(x_3x_4) && \text{by medial property} \\ &= ab \\ &= c \end{aligned}$$

□

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

Theorem 3.15 *$(3, d)$-UE-CSP is NP-complete for $d \geq 4$ even if there is only one constraint applied to each clause and even if no pair of variables appears in more than one clause.*

Proof. From Lemma 3.12, we have a reduction that converts a graph G into a $(3, d)$-UE-CSP formula F such that F contains only totally symmetric, medial constraints. However, F has three constraints on each clause. Note that if G is a simple graph, no pair of variables appears in more than one clause. Iteratively apply Construction 3.13 to each clause that has multiple constraints until we are left with a formula F'' that has only one constraint per clause. As each application of Construction 3.13 adds a constant number of variables and constraints, the size of F'' will be a polynomial in the size of F. From Lemma 3.14, each application of Construction 3.13 does not change the satisfiability of the formula. Therefore F'' is satisfiable if and only if F is satisfiable, and we have successfully modified the reduction of Lemma 3.12 into a reduction that converts G to a $(3, d)$-UE-CSP formula F'' such that F'' has no pair of variables that appear in more than one constraint. □

3.3.2.3 Open Problems on the Complexity of UE-CSP

Note that the proofs of NP-completeness each require only three constraints from the set of all possible uniquely extendible constraints. It should be possible to reduce that number to two for most cases by a careful choice of the constraints. For the case of $d = 4$, three constraints are required for the reduction from 3-COLOR of Theorem 3.11. This can be proven by generating the uniquely extendible constraints that contain all permutations of $(0, 1, 2)$ and observing that any pair has an additional tuple of values in common. However, it appears that if $d \geq 5$, we can always find two constraints so the reduction of Theorem 3.11 works. In addition, it seems that we can still find two such constraints even if we restrict the constraints to be totally symmetric and medial.

	0	1	2	3	4	5
0	4	2	1	5	0	3
1	2	1	0	4	3	5
2	1	0	3	2	5	4
3	5	4	2	3	1	0
4	0	3	5	1	4	2
5	3	5	4	0	2	1

	0	1	2	3	4	5
0	3	2	1	0	4	5
1	2	3	0	1	5	4
2	1	0	4	5	2	3
3	0	1	5	4	3	2
4	4	5	2	3	0	1
5	5	4	3	2	1	0

Table 3.5: Two constraints with $k = 3$ and $d = 6$ that are totally symmetric, medial and only share the tuple $(0, 1, 2)$. The constraints are listed as multiplication tables. (x, y, z) is a legal tuple for the constraint if the value at the xth row and yth column is z.

Conjecture 3.16 *$(3, d)$-UE-CSP is NP-complete for all $d \geq 5$ even if restricted to only two totally symmetric, medial constraints.*

For example, if $d = 5$, the following two totally symmetric, medial constraints work: $\{C_\emptyset, C_{(0,1)(3,4)}\}$. However, this pattern of using a constraint that is isomorphic to $C_\emptyset$ does not always work. For $d = 6$, Table 3.3 lists the tuples of $C_\emptyset$. Notice that

$$(1, 1, 1), (3, 3, 3), (5, 5, 5) \in C_\emptyset.$$

This implies, for the isomorphic constraint, the permutation must map $\{1, 3, 5\}$ onto $\{0, 2, 4\}$. However,

$$(0, 1, 2), (0, 4, 5), (1, 3, 5), (2, 3, 4) \in C_\emptyset$$

are the only tuples with three distinct values. As a result the permutation must map either $\{0, 1, 2\}$, $\{0, 4, 5\}$, $\{1, 3, 5\}$, or $\{2, 3, 4\}$ onto $\{0, 1, 2\}$. The result is that the permutation must map 5 values onto 4. Therefore, it is impossible to find a totally symmetric, medial constraint isomorphic to $C_\emptyset$ and that violates every tuple of $C_\emptyset$ except $(0, 1, 2)$. Yet, it is possible to find two totally symmetric, medial constraints with $d = 6$ that only share the tuple $(0, 1, 2)$. Such a pairing is listed in Table 3.5.

Conjecture 3.17 *(k, d)-UE-CSP is NP-complete for all $k \geq 3$ and $d \geq 4$.*

It should be possible to extend the technique of Section 3.3.2.2 to the case where $d \geq k$ by using a reduction from k-COLOR. One example is Theorem 3.18. However, it is not clear how to extend the technique to the case where $d < k$.

Theorem 3.18 $(4, 4)$-*UE-CSP is NP-complete even when restricted to the case that exactly one constraint is applied to each clause and no pair of variables appears in more than one clause.*

Proof. The proof uses a reduction from 4-COLOR. Replace each edge uv of the graph by a clause of size 4, (u, v, e_1, e_2), and apply the following three constraints to the clause: $C_\emptyset$, $C_{(1,2)}$, $C_{(2,3)}$. Here we define $C_\emptyset$ to be the set of tuples (a, b, c, d) that satisfy the function $(a + b + c + d - 6) \mod 4 = 0$. Table 3.6 lists these constraints, and it is clear that the only assignment that satisfies all constraints is to set the variables of the clause a permutation of $\{0, 1, 2, 3\}$. The rest of the reduction from 4-COLOR follows the same reasoning as Theorem 3.11.

To convert this formula to one in which there is only one constraint per clause and no pair of variables appears in more than one clause, note that the constraints are totally symmetric and medial. In the case of $k = 4$, the medial property is that

$$(abc)(pqr)(xyz) = (apx)(bqy)(crz).$$

Following the same logic as Lemma 3.15, assume (a, b, c, d) is a clause in the formula. Let C be a constraint on (a, b, c, d) such that C is medial. Then remove C from (a, b, c, d) and add 12 new variables:

$$x, x', x'', y, y', y'', z, z', z'', \alpha, \alpha', \alpha''$$

and seven new clauses:

$$(a, x, y, z), (b, x', y', z'), (c, x'', y'', z''), (x, x', x'', \alpha), (y, y', y'', \alpha'),$$

$$(z, z', z'', \alpha''), (\alpha, \alpha', \alpha'', d).$$

$C_\emptyset$	$C_{(1,2)}$	$C_{(2,3)}$
(0, 0, 0, 2)	(0, 0, 0, 1)	(0, 0, 0, 3)
(0, 0, 1, 1)	(0, 0, 2, 2)	(0, 0, 1, 1)
(0, 0, 3, 3)	(0, 0, 3, 3)	(0, 0, 2, 2)
(0, 1, 2, 3)	(0, 1, 1, 1)	(0, 1, 2, 3)
(0, 2, 2, 2)	(0, 1, 2, 3)	(0, 3, 3, 3)
(1, 1, 1, 3)	(1, 1, 2, 2)	(1, 1, 1, 2)
(1, 1, 2, 2)	(1, 1, 3, 3)	(1, 1, 3, 3)
(1, 3, 3, 3)	(2, 2, 2, 3)	(1, 2, 2, 2)
(2, 2, 3, 3)	(2, 3, 3, 3)	(2, 2, 3, 3)

Table 3.6: The constraints used for Theorem 3.18. Each row of a constraint lists the unordered triples of values that the constraint permits to be assigned to a clause.

Finally, to each new clause, add the constraint C.

The new formula preserves the relation $abc = d$,

$$\begin{aligned} d &= \alpha\alpha'\alpha'' \\ &= (xx'x'')(yy'y'')(zz'z'') \\ &= (xyz)(x'y'z')(x''y''z'') && \text{by medial property} \\ &= abc, \end{aligned}$$

and the rest of the proof uses the same steps as the proof of Theorem 3.15. □

3.4 The Random Model

For each appropriate n, m, we define $\Omega^{(k,d)}_{n,m}$ to be the set of (k, d)-UE-CSP instances with m clauses on variables $\{v_1, ..., v_n\}$ and one uniquely extendible constraint on each clause. We define $U^{(k,d)}_{n,m}$ to be a uniformly random member of $\Omega^{(k,d)}_{n,m}$. When m is defined to be some function $g(n)$, we often write $U^{(k,d)}_{n,m=g(n)}$. As is common in the study of random problems of this sort, we will be most interested in the case where $m = cn$ for some constant c. This model is equivalent to first choosing a uniformly random hypergraph on n vertices and m hyperedges to be the underlying hypergraph of the (k, d)-UE-CSP instance, and then for each hyperedge, choosing a uniformly random uniquely extendible constraint of size k and domain size d.

We can consider a second random model. Define $U_{n,p}^{(k,d)}$ to be an instance of (k,d)-UE-CSP on n variables where each of the $\binom{n}{k}$ clauses occurs in $U_{n,p}^{(k,d)}$ with probability p and a uniformly random constraint is applied to each clause.

From results of [Bol79, Łuc90] on random structures, the two models are asymptotically equivalent in the sense that

$$\lim_{n\to\infty} \mathbf{Pr}\left(U_{n,m}^{(k,d)} \text{ has property } \mathcal{A}\right) = \lim_{n\to\infty} \mathbf{Pr}\left(U_{n,p}^{(k,d)} \text{ has property } \mathcal{A}\right)$$

if $\mathcal{A}$ is a monotone (increasing or decreasing) property and m and $\binom{n}{k}p$ are "close" to each other. Formally, m is "close" to $\binom{n}{k}p$ if

$$m = \binom{n}{k}p + \mathrm{O}\left(\sqrt{\binom{n}{k}p(1-p)}\right),$$

and p is "close" to $\frac{m}{\binom{n}{k}}$ if

$$p = \frac{m}{\binom{n}{k}} + \mathrm{O}\left(\sqrt{\frac{m\left(\binom{n}{k}-m\right)}{\binom{n}{k}^3}}\right).$$

The ability to switch between the two models will be useful in some of the proofs of this thesis.

3.5 Resolution Complexity

While the uncertainty of the P versus NP question means that we can not state definitively whether an efficient algorithm exists to either solve an instance of (k,d)-UE-CSP or prove no solution to the instance exists, we can state that there is no efficient resolution based algorithm, such as DPLL, that will correctly handle unsatisfiable instances of (k,d)-UE-CSP for $k \geq 3$. The proof of a.s. exponential resolution complexity follows along the same lines as the techniques of [Mit02] and [MS07].

Theorem 3.19 *For any constant $c > 0$, and any $k \geq 3$, $d \geq 2$, the resolution complexity of a uniformly random instance of (k, d)-UE-CSP with n variables and cn clauses is a.s. $2^{\Theta(n)}$.*

Proof. From techniques developed in [BSW01, Mit02, MS07], a.s. the shortest resolution proof of unsatisfiability of a constraint satisfaction problem on n variables has exponential size if there exists constants $\alpha, \zeta > 0$ such that a.s. the following three conditions hold.

1. Every subformula on at most αn variables is satisfiable.

2. Every subproblem on v variables, where $\frac{1}{2}\alpha n \leq v \leq \alpha n$, has at least ζn variables of degree at most 1, where the degree of a variable is the number of clauses containing that variable.

3. If x is a variable of degree at most 1 in a CSP F then, letting F' be the subproblem obtained by removing x and its clause, any satisfying assignments of F' can be extended to a satisfying assignment of F by assigning some value to x.

Because our random model for UE-CSP applies one uniquely extendible constraint to each clause, the third condition is trivially true. The following lemma from [MS07] states a useful property of random formulae with a linear number of clauses: a.s. every subproblem on at most αn variables has a low clause density. A similar lemma is proven in [Mit02] and several other papers.

Lemma 3.20 ([MS07]) *Let $c > 0$ and $k \geq 2$ and let H be a random k-uniform hypergraph with n vertices and $m = cn$ edges. Then for any $\delta > 0$, there exists $\alpha = \alpha(c, k, \delta) > 0$ such that a.s. H has no subgraph with $0 < h \leq \lfloor \alpha n \rfloor$ vertices and at least $\left(\frac{1+\delta}{k-1}\right) h$ edges.*

Let F' be a minimally unsatisfiable subformula of F with at most $h \leq \lfloor \alpha n \rfloor$ variables. Then F' cannot have a variable of degree less than 2 because condition 3 above is trivially true for UE-CSP. Thus, F' must have at least $\left(\frac{2}{k}\right) h$ edges. However, if $k > 2$, there exists $\delta = \delta(k) > 0$ such that $\frac{2}{k} > \frac{1+\delta}{k-1}$. As a result, a.s. every subformula on h variables, and in particular F', will be satisfiable. By the same argument, let $\zeta = 1 - \frac{(1-\delta)k}{2(k-1)}$, and F' must have at least ζn variables of degree at most 1. $\square$

Chapter 4

The Satisfiability Threshold for (k, d)-UE-CSP

4.1 Introduction

This chapter studies the satisfiability thresholds for uniformly random instances of (k, d)-UE-CSP. The random model considered is described in Section 3.4. Section 4.2 contains a simple proof for the $(2, d)$-UE-CSP threshold. Section 4.3 contains the precise description of the Maximum Hypothesis. Section 4.4 contains a proof that if the Maximum Hypothesis holds, then the satisfiability threshold for $(3, d)$-UE-CSP is at a specific constant. Section 4.5 extends the analysis from Section 4.4 to all $k \geq 3$. Section 4.6 presents numerical evidence supporting the Maximum Hypothesis.

For $d = 2$, i.e. XOR-SAT, the threshold for the case $k = 3$ is proven in [DM02], and the proof of Section 4.4 closely follows the technique from this paper. However, the increased domain size leads to a more complicated calculation. Similarly, a formula for the threshold of random k-XOR-SAT, $k \geq 2$, is given in [MRTZ03], with some proof details omitted, and the threshold values for $k \leq 6$ are stated in that paper.

Section 4.4 is joint work with Michael Molloy and originally appeared in [CM04].

4.2 The Satisfiability Threshold for 2-UE-CSP

This section contains a proof that the satisfiability transition for random $(2, d)$-UE-CSP is $\frac{1}{2}$. Note that, unlike the case $k \geq 3$, $(2, d)$-UE-CSP does not have a sharp satisfiability threshold.

Lemma 4.1 *For $0 < c < \frac{1}{2}$, a uniformly random instance C of $(2, d)$-UE-CSP, $d \geq 2$ with n variables and cn clauses is w.u.p.p. satisfiable, and for $c > \frac{1}{2}$, it is a.s. unsatisfiable.*

Proof. Given c, let F be a uniformly random instance of $(2, d)$-UE-CSP on n variables and cn clauses. As described in Section 1.1, consider the random graph G that is the underlying hypergraph of F. The proof follows from the following well known properties of random graphs on n vertices and cn edges. For any constant c, the number of cycles of constant length in G is asymptotically equivalent to a Poisson random variable with a mean that depends on c, not on n. If $c < \frac{1}{2}$ then G a.s. has no cycles with a length that tends to ∞ as n grows. If $c > \frac{1}{2}$ then the number of cycles with length $\Omega(\log n)$ grows unbounded as n increases.

Choose an arbitrary variable v of F and assign it a value. We will then expose the rest of the formula in rounds. We will start by exposing, one at a time, the clauses that contain v. We will then expose, one at a time, the clauses containing a neighbor of v, and then the clauses containing a variable at distance 2 from v, and so on. We will continue this process until all clauses of the connected component containing v are exposed, and then we will repeat the process with another component until the entire formula is exposed.

In this exposure process, when we expose a clause, at least one of the variables in that clause will have been assigned a value. If the other variable has not been assigned a value, then we will assign it the value forced by the constraint on that clause from the value of the already assigned variable. If the clause contains two variables that have already been assigned values, then with probability $\frac{1}{d}$ the constraint on that clause will permit the pair of assigned values. This probability is derived from the observation of Section 3.2 that uniquely extendible constraints with $k = 2$ correspond to permutations. There are $d!$ possible constraints, and $(d-1)!$ will contain a particular ordered pair of values.

We will only expose a clause with both variables already assigned values if the underlying hypergraph of F contains cycles. From the above properties of random graphs, if $c < \frac{1}{2}$, w.u.p.p. the underlying hypergraph of F has no cycles, and so the exposure process will never expose a clause containing two variables that have already been assigned a value. As a result, w.u.p.p. the formula can be satisfied given any initial assignment to v. If $c > \frac{1}{2}$, then the number of times the process exposes a clause between two already assigned variables will grow as n grows and so the probability that the formula is satisfiable will tend to 0 as n tends to infinity. In particular, a well known property of random graphs with $c > \frac{1}{2}$ is that the giant component contains $\Omega(n)$ more edges than vertices. This implies that when we try to assign a value to a variable in this component, we will expose $\Omega(n)$ clauses with both variables already assigned. With probability $\left(\frac{1}{d}\right)^{\Omega(n)}$ we will satisfy all the constraints on all of these clauses, and so any assignment will a.s. fail. $\square$

4.3 The Maximum Hypothesis

At this time, we can only prove the location of the satisfiability threshold for $(3,d)$-UE-CSP under the assumption that the following technical hypothesis holds for the same d.

Hypothesis 4.2 (Maximum Hypothesis) *Consider any integer constant $d \geq 2$ and* *any real constant* $0.8 < c \leq 1$. *The function*

$$\begin{aligned} f(\alpha, r, t) \;=\;& \ln d - c\ln d + (1-\alpha)\ln(d-1) - c(2+t-3r)\ln(d-1) \\ &+ c(1-3r+2t)\ln(d-2) - \alpha\ln\alpha - (1-\alpha)\ln(1-\alpha) \\ &- c(1-3r+2t)\ln(1-3r+2t) - c(3r-3t)\ln(r-t) - ct\ln t + r3c\ln r \\ &+ (1-r)3c\ln(1-r) + \alpha\ln(e^z-1-z) - r3c\ln z + (1-\alpha)\ln(e^y-1-y) \\ &- (1-r)3c\ln y - \ln(e^x-1-x) + 3c\ln x, \end{aligned}$$

where $x, y, z > 0$ *are defined as*

$$\frac{e^x-1-x}{e^x-1} - \frac{x}{3c} = \frac{e^y-1-y}{e^y-1} - \frac{y(1-\alpha)}{3c(1-r)} = \frac{e^z-1-z}{e^z-1} - \frac{z\alpha}{3cr} = 0,$$

has a unique maximum in the region bounded by $0 \leq \alpha, r, t \leq 1$. *Furthermore, this maximum is at the point* $\alpha = \frac{1}{d}$, $r = \frac{1}{d}$, *and* $t = \frac{1}{d^2}$ *with* $f\left(\frac{1}{d}, \frac{1}{d}, \frac{1}{d^2}\right) = 2(1-c)\ln d$.

The Maximum Hypothesis is used in the proof of the lower bound for the satisfiability threshold given in Section 4.4.2. To find the lower bound, we use a second moment argument that gives a rather complicated formula. The equation f of the Maximum Hypothesis represents the exponential terms of the second moment. We know that the point $\alpha = \frac{1}{d}$, $r = \frac{1}{d}$, and $t = \frac{1}{d^2}$ is a local maximum, and we need the Maximum Hypothesis in order assume that it is also the unique global maximum so that we can use the Laplace method to approximate the second moment. In Section 4.6, we provide some evidence supporting this hypothesis.

4.4 The Satisfiability Threshold for $(3, d)$-UE-CSP

As described in Section 2.1.2, a major hurdle in studying the satisfiability threshold for 3-SAT is that 3-SAT suffers from “jackpot phenomena”. Namely, the property that even if a random formula is a.s. unsatisfiable, one satisfiable formula will have an exponential number of solutions. One way to reduce the jackpot phenomena for both 3-SAT and

3-UE-CSP is to first trim away any variables of degree 0 or 1 from the formula. Unlike for 3-SAT, for 3-UE-CSP this trimming procedure is enough to remove the jackpots. As a result, counting the expected number of satisfying assignments is sufficient to prove the location of the satisfiability threshold, conditional on the Maximum Hypothesis.

Theorem 4.3 *Under the Maximum Hypothesis, the satisfiability threshold for random $(3, d)$-UE-CSP is $c^* = .917935...$, where c^* is the clause density at which a random $(3, d)$-UE-CSP instance with n variables and c^*n clauses will a.s. contain a non-empty 2-core with clause to variable ratio $1 + o(1)$.*

This theorem, limited to $d = 4$, appears in [CM04]. The proof, on both sides of the threshold, is non-algorithmic and closely follows the proof of [DM02] for 3-XOR-SAT.

The proof of Theorem 4.3 will proceed as follows: first the variables of degree 0 and 1 are stripped away to produce a *2-core*. Then the satisfiability threshold on the 2-core is proven using the first and second moment methods, and this threshold on the 2-core yields the threshold for the original problem. The upper bound for satisfiability is a simple use of the first moment method, but the lower bound is more challenging. Applying the second moment method produces a complicated summation. Following the example of [DM02], the summation is approximated by a multiple integral, and then the Laplace Method is used to approximate the integral.

4.4.1 The 2-Core of the Underlying Hypergraph

Similar to the technique of Section 4.2, we will consider the underlying hypergraph of an instance F of (k, d)-UE-CSP. The *2-core* of a hypergraph is the largest (possibly empty) subgraph that has no vertices of degree less than 2. The 2-core is unique, and from Lemma 2.1 in Section 2.5.1, we know that if F is an instance of (k, d)-UE-CSP, then F is satisfiable if and only if the 2-core of F is satisfiable because of the “at least one extendibility” of (k, d)-UE-CSP.

Cores of random uniform hypergraphs are well-studied (see, for example, [PSW96, MWHC96, Mol05, CW06, Kim06, Rio07, DN]). From [CW06], the 2-core of a k-uniform random hypergraph on n vertices and cn edges has a.s. $(1-e^{-x}-xe^{-x})\,n+\mathrm{o}(n)$ vertices and a.s. $(1-e^{-x})^k\,cn+\mathrm{o}(n)$ hyperedges where x is the largest solution to $x=ck(1-e^{-x})^{k-1}$. As a result, letting

$$\gamma(c)=\frac{x(1-e^{-x})}{3(1-e^{-x}-xe^{-x})},$$

[Mol05] implies the following fact:

Fact 4.4 *Let $U^{(3,d)}_{n,m=cn}$ be a uniformly random instance of $(3,d)$-UE-CSP with n variables and $m=cn$ clauses. A.s. the 2-core of $U^{(3,4)}_{n,m=cn}$ has $\Theta(n)$ variables and $\gamma(c)+\mathrm{o}(1)$ times as many constraints as variables.*

Lemmas 4.6 and 4.7 and Fact 4.5 below prove that, under the Maximum Hypothesis, the satisfiability threshold for the 2-cores of random $(3,d)$-UE-CSP is at clause density 1. Thus we define c^* to be the solution to $\gamma(c)=1$, and we obtain Theorem 4.3.

Let $\Psi_{n,m}$ denote the subset of $\Omega^{(3,d)}_{n,m}$ in which every variable lies in at least 2 constraints, and let $U^*_{n,m}$ denote a uniformly random member of $\Psi_{n,m}$.

Fact 4.5 *For any n,m,n',m', if we condition on the event that the 2-core of $U^{(3,d)}_{n,m=cn}$ has n' variables and m' constraints, then that 2-core is a uniformly random member of $\Psi_{n',m'}$.*

Proof. This is a straightforward variation of the proof of Claim 1 in the proof of Lemma 4(b) from [Mol05], which is itself a very standard argument. Consider a hypergraph H and its 2-core H_c. Assume H_c has n' variables and m' edges. Replace H_c in H by a arbitrary member of $\Psi_{n',m'}$, H'_c. Call this new hypergraph H'. Note that H'_c is the 2-core of H' and that H' and H have the same number of edges. Thus, the probability a random hypergraph is equal to H is the same as the probability it is equal to H', and this implies the probability that the 2-core of a random hypergraph is H_c is equal to the probability that the 2-core is H'_c. □

All this implies that Theorem 4.3 is equivalent to the following two lemmas:

Lemma 4.6 *For every $c > 1$, $U^*_{n,m=cn}$ is a.s. unsatisfiable.*

Proof. We apply what is, in this field, a very standard and straightforward first moment argument. Consider a random instance F chosen from $\Psi_{n,m=cn}$, and let N denote the number of satisfying assignments of F. We will show that $\mathbf{E}(N) = \mathrm{o}(1)$; this implies that a.s. $N = 0$; i.e., that a.s. F is unsatisfiable.

Consider any assignment σ of values to the variables of F. Since each constraint is uniquely extendible, for each possible setting of $k - 1$ variables in a constraint, there is exactly one possible value for the kth variable. Because the random model considered includes all possible uniquely extendible constraints and because there are d possible values for the kth variable, the probability that a particular constraint is satisfied by σ is $\frac{1}{d}$. As there are d^n choices for σ, we have

$$\mathbf{E}(N) = d^n d^{-m} = d^{(1-c)n} = \mathrm{o}(1),$$

since $c > 1$. □

Lemma 4.7 *Under the Maximum Hypothesis, for every $c < 1$, $U^*_{n,m=cn}$ is a.s. satisfiable.*

The proof of this lemma is much more involved, and the proof is presented in the next subsection.

4.4.2 A Second Moment Argument

This section contains the proof of Lemma 4.7, the hardest part of Theorem 4.3. Inspired by the proof of the corresponding theorem in [DM02], this proof applies the second moment argument. Unfortunately, the increased domain size yields a larger set of constraints to choose from and more complicated calculations than those in [DM02].

Unique extendibility, and in particular "at most one extendibility", is crucial to proving Lemma 4.7 by the second moment method, as described in Remark 2.2 in Section 2.5.1.

Proof. As in the proof of Lemma 4.6, consider a random instance F chosen from $\Psi_{n,m=cn}$, and let N denote the number of satisfying assignments of F. Again, we have $\mathbf{E}(N) = d^{(1-c)n}$. The main step of this proof is to compute the second moment of N obtaining:

Lemma 4.8 *Under the Maximum Hypothesis,* $\mathbf{E}(N^2) = \mathbf{E}(N)^2(1 + \mathrm{o}(1))$.

Because N is non-negative, a well known application of the Cauchy-Swartz inequality implies

$$\mathbf{Pr}(N > 0) \geq \frac{\mathbf{E}(N)^2}{\mathbf{E}(N^2)},$$

and so Lemma 4.8 implies Lemma 4.6.

Following the technique of [DM02], the proof will compute $\mathbf{E}(N^2)$ by putting $\frac{\mathbf{E}(N^2)}{\mathbf{E}(N)^2}$ into the form

$$\int_{a_1}^{b_1}\int_{a_2}^{b_2}\int_{a_3}^{b_3} g(x_1, x_2, x_3)e^{nh(x_1,x_2,x_3)}dx_1\, dx_2\, dx_3$$

where g is polynomial in n and h has a unique maximum in the range of the integrals. Then we use the Laplace Method to approximate the triple integral.

As mentioned above, let the d^n possible assignments be $\sigma_1, ..., \sigma_{d^n}$, and let N_i be the indicator variable that σ_i is a satisfying assignment. Then $N = N_1 + ... + N_{d^n}$ and so $N^2 = \sum_{i,j} N_i N_j$. Since $N_i N_j = 1$ if and only if F is satisfied by both σ_i and σ_j, this indicates that we must focus on counting the number of instances satisfied by two assignments to the variables.

Similarly to [DM02], let σ and τ be arbitrary assignments to the variables, let $\#\mathcal{C}$ be the total number of instances in $\Psi_{n,m}$, and let $\#\mathcal{C}_{\sigma,\tau}$ be the total number of instances in

$\Psi_{n,m}$ that are satisfied by both σ and τ. Then,

$$\mathbf{E}(N^2) = \frac{1}{\#\mathcal{C}} \sum_{\sigma,\tau} \#\mathcal{C}_{\sigma,\tau}.$$

Let q be the number of possible uniquely extendible constraints of size 3. So for each clause, we will have q choices for the constraint to apply to that clause. As is done in [DM02], we can think of the m clauses as inducing a distribution of $3m$ "places" to the n variables such that each variable receives at least 2 "places". So, $\#\mathcal{C} = q^m S(3m, n, 2)n!$ where $S(i, j, 2)$, known as a generalized Stirling number of the second kind, counts the number of ways to partition i elements into j sets such that each set has at least 2 elements, for i and j positive integers.

Consider a clause and a random constraint on that clause. We need to determine the probability that both assignments σ and τ satisfy the constraint. Place an arbitrary ordering on the variables of the clause, and let α, β, and γ be the values assigned to those variables by σ, and let α', β', and γ' be the values assigned to those variables by τ. In addition, let $z_{\alpha\beta}$ be the value the constraint forces the third variable to be if the first variable is assigned α and the second variable is assigned β.

If the values of the three variables are unchanged between σ and τ, i.e. if $\alpha = \alpha'$, $\beta = \beta'$, and $\gamma = \gamma'$, then the probability that both assignments satisfy a random constraint is the same as the probability that a random constraint assigns the third variable γ if the first two are assigned α and β respectively. Every constraint of size 3 will permit a tuple of the form $(\alpha, \beta, z_{\alpha\beta})$, and there are d possible choices for $z_{\alpha\beta}$. Exactly $\frac{1}{d}$ of the constraints will have $z_{\alpha\beta} = \gamma$. As a result,

$$\mathbf{Pr}(z_{\alpha\beta} = \gamma) = \frac{1}{d}.$$

Thus, both assignments will satisfy a proportion of $\frac{1}{d}$ of the possible constraints.

Note that the uniquely extendible property means that if exactly one of the clause's variables changes value between σ and τ, for example if $\alpha = \alpha'$, $\beta = \beta'$, and $\gamma \neq \gamma'$, then the constraint can not be satisfied by both σ and τ.

Suppose one variable, assume w.l.o.g. the first variable, is assigned the same value by σ and τ and each of the other two variables is assigned a different value in τ from what it is assigned in σ, i.e. $\alpha = \alpha'$, $\beta \neq \beta'$, and $\gamma \neq \gamma'$. In this case, we need to determine

$$\mathbf{Pr}(z_{\alpha\beta} = \gamma \ \wedge \ z_{\alpha\beta'} = \gamma') = \mathbf{Pr}(z_{\alpha\beta'} = \gamma' | z_{\alpha\beta} = \gamma)\mathbf{Pr}(z_{\alpha\beta} = \gamma).$$

Every constraint of size 3 that permits the tuple (α, β, γ) will also permit a tuple of the form $(\alpha, \beta', z_{\alpha\beta'})$ with $z_{\alpha\beta'} \neq \gamma$. There are $d-1$ choices for $z_{\alpha\beta'}$, and by symmetry, each is equally likely. So exactly $\frac{1}{d-1}$ of the constraints will have $z_{\alpha\beta'} = \gamma'$. Thus,

$$\mathbf{Pr}(z_{\alpha\beta'} = \gamma' | z_{\alpha\beta} = \gamma)\mathbf{Pr}(z_{\alpha\beta} = \gamma) = \frac{1}{d(d-1)}.$$

As a result, both assignments will satisfy a proportion of $\frac{1}{d(d-1)}$ of the possible constraints.

Finally, if none of the variables of the clause receives the same value in τ as it does in σ then we must have $d > 2$ because each constraint with $d = 2$ is a parity check, and it is impossible for both τ and σ to satisfy the same parity check on the clause. If $\alpha \neq \alpha'$, $\beta \neq \beta'$, and $\gamma \neq \gamma'$, then

$$\begin{aligned}
\mathbf{Pr}(z_{\alpha\beta} = \gamma \ \wedge \ z_{\alpha'\beta'} = \gamma') &= \mathbf{Pr}(z_{\alpha\beta} = \gamma \ \wedge \ z_{\alpha\beta'} \neq \gamma' \ \wedge \ z_{\alpha'\beta'} = \gamma') \\
&= \mathbf{Pr}(z_{\alpha'\beta'} = \gamma' \mid z_{\alpha\beta} = \gamma \ \wedge z_{\alpha\beta'} \ \neq \gamma') \\
&\quad \times \ \mathbf{Pr}(z_{\alpha\beta'} \neq \gamma' | z_{\alpha\beta} = \gamma)\mathbf{Pr}(z_{\alpha\beta} = \gamma)
\end{aligned}$$

Consider only the tuples $(\alpha, \beta, z_{\alpha\beta})$, $(\alpha, \beta', z_{\alpha\beta'})$, and $(\alpha', \beta', z_{\alpha'\beta'})$ permitted by the constraint. There are d choices for $z_{\alpha\beta}$, and exactly $\frac{1}{d}$ of the constraints will have $z_{\alpha\beta} = \gamma$. For each constraint that contains the tuple (α, β, γ), there are exactly $d-1$ choices for $z_{\alpha\beta'}$, and $d-2$ of these choices are not γ'. By a straightforward symmetry argument, each is equally likely. As a result, $\frac{d-2}{d-1}$ of the constraints that contain the tuple (α, β, γ) will not contain the tuple $(\alpha, \beta', \gamma')$. Finally, for every choice of $z_{\alpha\beta}$ and $z_{\alpha\beta'}$ there are $d-1$ equally likely choices for $z_{\alpha'\beta'}$. The reason is that we fix the set of 3-tuples that start with α, and then the number of ways we can choose the set of tuples that starts with α' is equal to the number of derangements on d elements, and by symmetry each derangement is equally likely.

As a result, the expression above yields:

$$\begin{aligned}\mathbf{Pr}(z_{\alpha\beta} = \gamma \ \wedge \ z_{\alpha'\beta'} = \gamma') &= \frac{1}{d-1} \cdot \frac{d-2}{d-1} \cdot \frac{1}{d} \\ &= \frac{d-2}{d(d-1)^2}.\end{aligned}$$

So both assignments will simultaneously satisfy a proportion of $\frac{d-2}{d(d-1)^2}$ of the possible constraints.

Using the same notation as [DM02], let $I_k = \{0, \frac{1}{k}, \frac{2}{k}, \ldots, \frac{k-1}{k}, 1\}$, and let $\alpha \in I_n$ be the proportion of variables having the same value in both assignments. To enumerate all pairs of assignments, we must count the number of choices for the αn variables and count the possible assignments to the variables. This gives $\sum_{\alpha \in I_n} \binom{n}{\alpha n} d^n (d-1)^{(1-\alpha)n}$ pairs of assignments.

To enumerate all satisfied instances for one pair of assignments, let $r \in I_{3m}$ be the proportion of $3m$ "places" in the second assignment that receive one of the αn variables, and let T_k be the number of clauses with $3-k$ of these αn variables. Recall that $T_1 = 0$.

For each choice of T_0, T_2, T_3, we need to

(a) count the ways to choose the clauses for T_0, T_2, T_3:

$$\binom{m}{T_0}\binom{m-T_0}{T_2};$$

(b) for each clause, count the number of ways we can choose a constraint for the clause given the number of variables in the clause that receive the same value in τ as in σ:

$$\left(\frac{q(d-2)}{d(d-1)^2}\right)^{T_3} \left(\frac{q}{d(d-1)}\right)^{T_2} \left(\frac{q}{d}\right)^{T_0};$$

(c) for each clause in T_2, count the 3 positions for the αn variables:

$$3^{T_2};$$

(d) finally, distribute the variables amongst the "places":

$$S(r3m, \alpha n, 2)(\alpha n)! S((1-r)3m, (1-\alpha)n, 2)((1-\alpha)n)!.$$

Recall that $S(i,j,2)$ counts the number of ways to partition i elements into j sets such that each set has at least 2 elements.

In total, we have

$$\begin{aligned}\mathbf{E}(N^2) &= \frac{1}{q^m S(3m,n,2)n!} \sum_{\alpha \in I_n} \sum_{r \in I_{3m}} \sum_{\substack{T_0+T_2+T_3=m \\ T_2+3T_0=3rm}} \binom{n}{\alpha n} d^n (d-1)^{(1-\alpha)n} \\ &\quad \times \binom{m}{T_0} \binom{m-T_0}{T_2} \left(\frac{q(d-2)}{d(d-1)^2}\right)^{T_3} \left(\frac{q}{d(d-1)}\right)^{T_2} \left(\frac{q}{d}\right)^{T_0} 3^{T_2} \\ &\quad \times S(r3m, \alpha n, 2)(\alpha n)! S((1-r)3m, (1-\alpha)n, 2)((1-\alpha)n)!.\end{aligned}$$

Let $t \in I_m = \{0, \frac{1}{m}, \frac{2}{m}, \ldots, \frac{m-1}{m}, 1\}$ be the proportion of m clauses in which all 3 variables have the same assignment. Thus,

$$\begin{aligned} T_0 &= tm \\ T_2 &= 3rm - 3T_0 &&= 3rm - 3tm \\ T_3 &= m - T_0 - T_2 &&= m - 3rm + 2tm. \end{aligned}$$

Note that $T_2 \geq 0$ implies $r \geq t$ and $T_3 \geq 0$ implies $t \geq \frac{3r-1}{2}$. Substituting and factoring out common terms gives

$$\begin{aligned}\mathbf{E}(N^2) &= \frac{1}{S(3m,n,2)n!} \sum_{\alpha \in I_n} \sum_{r \in I_{3m}} \sum_{t \in I_m \cap [\frac{3r-1}{2}, r]} \binom{n}{\alpha n} d^n (d-1)^{(1-\alpha)n} \\ &\quad \times \left(\frac{m!}{(m-3rm+2tm)!(3rm-3tm)!(tm)!}\right) 3^{3rm-3tm} \\ &\quad \times \left(\frac{1}{d}\right)^m \left(\frac{1}{d-1}\right)^{2m+tm-3rm} (d-2)^{m-3rm+2tm} \\ &\quad \times S(r3m, \alpha n, 2)(\alpha n)! S((1-r)3m, (1-\alpha)n, 2)((1-\alpha)n)!.\end{aligned}$$

Next, we use Lemma 4.11 which will be presented in Section 4.4.4. This lemma appears in [DM02] but with a typographical error, and the lemma is a specific case of the general results in [Hen94]. It states:

$$S(i,j,2) \sim \frac{1}{j!} \left(\frac{i}{z_0 e}\right)^i (e^{z_0} - 1 - z_0)^j \Phi(i,j)$$

where z_0 is the positive real solution of the equation

$$\frac{j}{i}z_0 = \frac{e^{z_0}-1-z_0}{e^{z_0}-1}$$

and where

$$\Phi(i,j) = \sqrt{\frac{ij}{z_0 j(i-j) - i(i-2j)}}.$$

Also note that

$$\Phi(vi,vj) = \Phi(i,j) \text{ for any value } v. \tag{4.1}$$

This gives

$$\begin{aligned}
S(3m,n,2)n! &\sim (e^x-1-x)^n x^{-3m} e^{-3m}(3m)^{3m}\Phi(3m,n)\\
S(r3m,\alpha n,2)(\alpha n)! &\sim (e^z-1-z)^{\alpha n} z^{-r3m} e^{-r3m}(r3m)^{r3m}\Phi(r3m,\alpha n)\\
S((1-r)3m,(1-\alpha)n,2)((1-\alpha)n)! &\sim (e^y-1-y)^{(1-\alpha)n} y^{-(1-r)3m}\\
&\quad e^{-(1-r)3m}((1-r)3m)^{(1-r)3m}\\
&\quad \Phi((1-r)3m,(1-\alpha)n)
\end{aligned}$$

for some $x,y,z>0$ such that

$$\frac{e^x-1-x}{e^x-1} - \frac{x}{3c} = \frac{e^y-1-y}{e^y-1} - \frac{y(1-\alpha)}{3c(1-r)} = \frac{e^z-1-z}{e^z-1} - \frac{z\alpha}{3cr} = 0. \tag{4.2}$$

Substituting these terms and using Stirling's Approximation that $i! \sim i^i e^{-i}\sqrt{2\pi i}$, gives

$$\begin{aligned}
\mathbf{E}(N^2) \sim\ & \frac{1}{(e^x-1-x)^n x^{-3m} e^{-3m}(3m)^{3m}\Phi(3m,n)} \sum_{\alpha\in I_n}\sum_{r\in I_{3m}}\sum_{t\in I_m\cap[\frac{3r-1}{2},r]} d^n\\
&\times (d-1)^{(1-\alpha)n}\left(\frac{n}{e}\right)^n \left(\frac{\alpha n}{e}\right)^{-\alpha n}\left(\frac{n-\alpha n}{e}\right)^{-(n-\alpha n)} \frac{\sqrt{2\pi n}}{\sqrt{2\pi\alpha n}\sqrt{2\pi(n-\alpha n)}}\\
&\times \left(\frac{m}{e}\right)^m \left(\frac{m-3rm+2tm}{e}\right)^{-(m-3rm+2tm)} \left(\frac{3rm-3tm}{e}\right)^{-(3rm-3tm)}\\
&\times \left(\frac{tm}{e}\right)^{-tm} \frac{\sqrt{2\pi m}}{\sqrt{2\pi(m-3rm+2tm)}\sqrt{2\pi(3rm-3tm)}\sqrt{2\pi tm}}\\
&\times 3^{3rm-3tm}\left(\frac{1}{d}\right)^m \left(\frac{1}{d-1}\right)^{2m+tm-3rm} (d-2)^{m-3rm+2tm}\\
&\times (e^z-1-z)^{\alpha n} z^{-r3m} e^{-r3m}(r3m)^{r3m}\Phi(r3m,\alpha n)(e^y-1-y)^{(1-\alpha)n}\\
&\times y^{-(1-r)3m} e^{-(1-r)3m}((1-r)3m)^{(1-r)3m}\Phi((1-r)3m,(1-\alpha)n).
\end{aligned}$$

Grouping the non-exponential and exponential terms and simplifying allows us to write $\mathbf{E}(N^2)$ in the form

$$\mathbf{E}(N^2) \sim \sum_{\alpha \in I_n} \sum_{r \in I_{3m}} \sum_{t \in I_m \cap [\frac{3r-1}{2}, r]} g(\alpha, r, t) e^{nf(\alpha,r,t)}$$

where

$$\begin{aligned}
g(\alpha, r, t) &= \Phi(3m, n)^{-1}\Phi(r3m, \alpha n)\Phi((1-r)3m, (1-\alpha)n) \\
&\quad \times (2\pi n\alpha(1-\alpha))^{-\frac{1}{2}}(2\pi m)^{-1}((1-3r+2t)(3r-3t)t)^{-\frac{1}{2}} \\
f(\alpha, r, t) &= \frac{1}{n}\left[n \ln d + (1-\alpha)n\ln(d-1) - m\ln d - m(2+t-3r)\ln(d-1)\right. \\
&\quad + m(1-3r+2t)\ln(d-2) + m(3r-3t)\ln 3 - n\alpha\ln\alpha \\
&\quad - n(1-\alpha)\ln(1-\alpha) - m(1-3r+2t)\ln(1-3r+2t) \\
&\quad - m(3r-3t)\ln(3r-3t) - mt\ln t + \alpha n\ln(e^z-1-z) - r3m\ln z \\
&\quad - r3m + r3m\ln(r3m) + (1-\alpha)\ln(e^y-1-y) - (1-r)3m\ln y \\
&\quad - (1-r)3m + (1-r)3m\ln((1-r)3m) - n\ln(e^x-1-x) + 3m\ln x \\
&\quad \left. +3m - 3m\ln(3m)\right].
\end{aligned}$$

We fix c and replace m with cn. Combining common terms, and dividing f through by n gives

$$\begin{aligned}
g(\alpha, r, t) &= \Phi(3c, 1)^{-1}\Phi(r3c, \alpha)\Phi((1-r)3c, (1-\alpha)) \\
&\quad \times (2\pi n)^{-\frac{3}{2}}c^{-1}(\alpha(1-\alpha)(1-3r+2t)(3r-3t)t)^{-\frac{1}{2}}, \\
f(\alpha, r, t) &= \ln d - c\ln d + (1-\alpha)\ln(d-1) - c(2+t-3r)\ln(d-1) \\
&\quad + c(1-3r+2t)\ln(d-2) - \alpha\ln\alpha - (1-\alpha)\ln(1-\alpha) \\
&\quad - c(1-3r+2t)\ln(1-3r+2t) - c(3r-3t)\ln(r-t) - ct\ln t + r3c\ln r \\
&\quad + (1-r)3c\ln(1-r) + \alpha\ln(e^z-1-z) - r3c\ln z + (1-\alpha)\ln(e^y-1-y) \\
&\quad - (1-r)3c\ln y - \ln(e^x-1-x) + 3c\ln x,
\end{aligned}$$

and thus,

$$\frac{E(N^2)}{E(N)^2} \sim \sum_{\alpha \in I_n} \sum_{r \in I_{3cn}} \sum_{t \in I_{cn} \cap [\frac{3r-1}{2}, r]} g(\alpha, r, t) e^{n(f(\alpha,r,t) - 2(1-c)\ln d)}.$$

Continuing with the technique of [DM02], we will use the Laplace Method to approximate this sum. In order to apply the Laplace Method, we must find the maxima for f and prove that f has only one maximum on the region defined by the bounds of α, r, and t. Unfortunately, we are unable to prove that f has only one maximum on this region, and so we need to condition these results with the assumption that the Maximum Hypothesis holds.

If we differentiate f with respect to α, r, t, we get

$$\begin{aligned}
\frac{\partial f}{\partial \alpha} &= D_\alpha[(1-\alpha)\ln(d-1) - \alpha \ln \alpha - (1-\alpha)\ln(1-\alpha) + \alpha \ln(e^z - 1 - z) \\
&\qquad - r3c \ln x + (1-\alpha)\ln(e^y - 1 - y) - (1-r)3c \ln y] \\
&= -\ln(d-1) - \ln \alpha + \ln(1-\alpha) + \ln(e^z - 1 - z) - \ln(e^y - 1 - y) \\
&\qquad + \left[\frac{\alpha(e^z-1)}{e^z - 1 - z} - \frac{3rc}{z}\right]\frac{\partial z}{\partial \alpha} + \left[\frac{(1-\alpha)(e^y-1)}{e^y-1-y} - \frac{(1-r)3c}{y}\right]\frac{\partial y}{\partial \alpha} \\
&= -\ln(d-1) - \ln \alpha + \ln(1-\alpha) + \ln(e^z - 1 - z) - \ln(e^y - 1 - y) \qquad \text{by (4.2)}
\end{aligned}$$

$$\begin{aligned}
\frac{\partial f}{\partial r} &= D_r[3rc\ln(d-1) - 3rc\ln(d-2) - c(1-3r+2t)\ln(1-3r+2t) \\
&\qquad - c(3r-3t)\ln(r-t) + r3c\ln r + (1-r)3c\ln(1-r) + \alpha\ln(e^z - 1 - z) \\
&\qquad - r3c\ln z + (1-\alpha)\ln(e^y - 1 - y) - (1-r)3c\ln y] \\
&= 3c\ln(d-1) - 3c\ln(d-2) + 3c\ln(1-3r+2t) - 3c\ln(r-t) + 3c\ln r \\
&\qquad - 3c\ln(1-r) - 3c\ln z + 3c\ln y + \left[\frac{\alpha(e^z-1)}{e^z-1-z} - \frac{r3c}{z}\right]\frac{\partial z}{\partial r} \\
&\qquad + \left[\frac{(1-\alpha)(e^y-1)}{e^y-1-y} - \frac{(1-r)3c}{y}\right]\frac{\partial y}{\partial r} \\
&= 3c\ln(d-1) - 3c\ln(d-2) + 3c\ln(1-3r+2t) - 3c\ln(r-t) + 3c\ln r \\
&\qquad - 3c\ln(1-r) - 3c\ln z + 3c\ln y \qquad \text{by (4.2)}
\end{aligned}$$

$$\begin{aligned}\frac{\partial f}{\partial t} &= D_t[-ct\ln(d-1) + c2t\ln(d-2) - c(1-3r+2t)\ln(1-3r+2t) \\ &\quad - c(3r-3t)\ln(r-t) - ct\ln t] \\ &= -c\ln(d-1) + 2c\ln(d-2) - 2c\ln(1-3r+2t) + 3c\ln(r-t) - c\ln t.\end{aligned}$$

Setting $\frac{\partial f}{\partial \alpha} = \frac{\partial f}{\partial r} = \frac{\partial f}{\partial t} = 0$ implies

$$\frac{1-\alpha}{\alpha} = (d-1)\frac{e^y - 1 - y}{e^z - 1 - z} \tag{4.3}$$

$$\frac{1-r}{r} = \frac{d-1}{d-2}\cdot\frac{y}{z}\cdot\frac{(1-3r+2t)}{(r-t)} \tag{4.4}$$

$$\frac{(r-t)^3}{(1-3r+2t)^2} = \frac{(d-1)t}{(d-2)^2}. \tag{4.5}$$

To find a maximum for f, we use the guess that $x = y = z$. The intuition for the guess is that x, y, and z correspond to the parameter for the truncated Poisson random variable used to model the degrees of all the variables in the 2-core, the variables in the 2-core that are not in the set of αn variables, and the variables in the 2-core that are the set of αn variables, respectively. Since f is considering all possible sets of size αn, it is reasonable to guess that the expected degrees for the variables in each set are the same when f is maximized.

Plugging this guess into (4.3), (4.4) and (4.5) gives $\alpha = \frac{1}{d}, r = \frac{1}{d}, t = \frac{1}{d^2}$, and

$$f\left(\frac{1}{d},\frac{1}{d},\frac{1}{d^2}\right) = 2(1-c)\ln d.$$

Setting $\alpha = \frac{1}{d}$ and $r = \frac{1}{d}$ in (4.2) confirms that that $x = y = z$ at a stationary point of f. Below we prove that $x = y = z$ is a local maximum validating the guess.

Next, we replace the summations with integrals. The summations are essentially a Riemann sum, and as n tends to infinity, the error term from approximating the summations with integrals tends to 0.

$$\begin{aligned}
\frac{\mathbf{E}(N^2)}{\mathbf{E}(N)^2} &\sim \sum_{\alpha\in I_n}\sum_{r\in I_{3cn}}\sum_{t\in I_{cn}\cap[\frac{3r-1}{2},r]} g(\alpha,r,t)e^{n(f(\alpha,r,t)-2(1-c)\ln d)} \\
&\sim n\cdot 3cn\cdot cn\cdot \\
&\qquad \int_0^1\int_0^1\int_{\max\{0,\frac{3r-1}{2}\}}^r g(\alpha,r,t)e^{n(f(\alpha,r,t)-2(1-c)\ln d)}dt\,dr\,d\alpha \\
&= 3c^2n^3\int_0^1\int_0^1\int_{\max\{0,\frac{3r-1}{2}\}}^r g(\alpha,r,t)e^{n(f(\alpha,r,t)-2(1-c)\ln d)}dt\,dr\,d\alpha.
\end{aligned}$$

Continuing with the technique of [DM02], we will use the Laplace Method to approximate the value of the integrals, conditional on the Maximum Hypothesis. To apply the Laplace Method, we require that f has a unique local maximum in the domain of integration. We know that $\alpha=\frac{1}{d}, r=\frac{1}{d}, t=\frac{1}{d^2}$ is a local maximum, and the Maximum Hypothesis implies that it is the unique global maximum in the domain of integration. See, for example, [BO78] and [dB70] for descriptions of the Laplace Method. The Laplace Method for a triple integral can be stated as follows.

Lemma 4.9 ([dB70]) *Let*

$$F(n)=\int_{a_1}^{b_1}\int_{a_2}^{b_2}\int_{a_3}^{b_3} g(x_1,x_2,x_3)e^{nh(x_1,x_2,x_3)}dx_1\,dx_2\,dx_3$$

where

(a) h is continuous in $a_i\le x_i\le b_i$,

(b) $h(c_1,c_2,c_3)=0$ for some point (c_1,c_2,c_3) with $a_i<c_i<b_i$ and $h(x_1,x_2,x_3)<0$ for all other points in the range,

(c) $h(x_1,x_2,x_3)=-\frac{1}{2}\sum_{i=1}^{3}\sum_{j=1}^{3}a_{ij}x_ix_j+\mathrm{o}(x_1^2+x_2^2+x_3^2)$ with $(x_1^2+x_2^2+x_3^2\to 0)$, and

(d) the quadratic form $\sum\sum a_{ij}x_ix_j$ is positive definite.

Then,

$$F(n) \sim (2\pi)^{\frac{3}{2}} D^{-\frac{1}{2}} n^{-\frac{3}{2}} g(c_1, c_2, c_3)$$

where D is the determinant of the matrix (a_{ij}).

Just as is done in [DM02], we apply the lemma by letting $h(x_1, x_2, x_3) = f(\alpha, r, t) - 2(1-c)\ln d$. By our choice of h, point (a) is satisfied, and point (b) is assumed true in view of the value of f at the hypothesized maximum. Point (c) is satisfied if we approximate h by the Taylor expansion about the point $\alpha = \frac{1}{d}$, $r = \frac{1}{d}$, $t = \frac{1}{d^2}$ and take the a_{ij}'s from the second partial derivatives of h. The second partial derivatives of f are:

$$\begin{aligned}
f_{\alpha\alpha} &= -\frac{1}{\alpha} - \frac{1}{1-\alpha} + \frac{e^z-1}{e^z-1-z}\frac{\partial z}{\partial \alpha} - \frac{e^y-1}{e^y-1-y}\frac{\partial y}{\partial \alpha} \\
f_{\alpha r} &= \frac{e^z-1}{e^z-1-z}\frac{\partial z}{\partial r} - \frac{e^y-1}{e^y-1-y}\frac{\partial y}{\partial r} \\
f_{\alpha t} &= 0 \\
f_{r\alpha} &= \frac{-3c}{z}\frac{\partial z}{\partial \alpha} + \frac{3c}{y}\frac{\partial y}{\partial \alpha} \\
f_{rr} &= -\frac{3c}{r-t} + \frac{3c}{r} + \frac{3c}{1-r} - \frac{9c}{1-3r+2t} - \frac{3c}{z}\frac{\partial z}{\partial r} + \frac{3c}{y}\frac{\partial y}{\partial r} \\
f_{rt} &= \frac{3c}{r-t} + \frac{6c}{1-3r+2t} \\
f_{t\alpha} &= 0 \\
f_{tr} &= \frac{3c}{r-t} + \frac{6c}{1-3r+2t} \\
f_{tt} &= -\frac{3c}{r-t} - \frac{c}{t} - \frac{4c}{1-3r+2t}
\end{aligned}$$

where, from (4.2),

$$\begin{aligned}
\frac{\partial z}{\partial \alpha} &= \frac{-z(e^z-1)^2}{\alpha(e^z-1)^2 + 3rc(e^z(e^z-1-z) - (e^z-1)^2)} \\
\frac{\partial z}{\partial r} &= \frac{\alpha z(e^z-1)^2}{r[\alpha(e^z-1)^2 + 3rc(e^z(e^z-1-z) - (e^z-1)^2)]} \\
\frac{\partial y}{\partial \alpha} &= \frac{y(e^y-1)^2}{(1-\alpha)(e^y-1)^2 + 3(1-r)c(e^y(e^y-1-y) - (e^y-1)^2)} \\
\frac{\partial y}{\partial r} &= \frac{-(1-\alpha)y(e^y-1)^2}{(1-r)[(1-\alpha)(e^y-1)^2 + 3(1-r)c(e^y(e^y-1-y) - (e^y-1)^2)]}.
\end{aligned}$$

Since $x = y = z$ at the maximum and if we let $K = \frac{c(e^x-1)^2}{(e^x-1)^2+3c(e^x-xe^x-1)}$, we have

$$\begin{aligned}
f_{\alpha\alpha}(\frac{1}{d},\frac{1}{d},\frac{1}{d^2}) &= -\frac{d^2}{d-1} - \frac{3d^2}{d-1}K \\
f_{\alpha r}(\frac{1}{d},\frac{1}{d},\frac{1}{d^2}) &= 3K\left(\frac{d^2}{d-1}\right) \\
f_{\alpha t}(\frac{1}{d},\frac{1}{d},\frac{1}{d^2}) &= 0 \\
f_{r\alpha}(\frac{1}{d},\frac{1}{d},\frac{1}{d^2}) &= 3K\left(\frac{d^2}{d-1}\right) \\
f_{rr}(\frac{1}{d},\frac{1}{d},\frac{1}{d^2}) &= -\frac{9cd^2}{(d-2)(d-1)} - 3K\frac{d^2}{d-1} \\
f_{rt}(\frac{1}{d},\frac{1}{d},\frac{1}{d^2}) &= \frac{3cd^3}{(d-1)(d-2)} \\
f_{t\alpha}(\frac{1}{d},\frac{1}{d},\frac{1}{d^2}) &= 0 \\
f_{tr}(\frac{1}{d},\frac{1}{d},\frac{1}{d^2}) &= \frac{3cd^3}{(d-1)(d-2)} \\
f_{tt}(\frac{1}{d},\frac{1}{d},\frac{1}{d^2}) &= -cd^2\left(1 + \frac{3}{d-1} + \frac{4}{(d-1)(d-2)}\right).
\end{aligned}$$

Thus,

$$(a_{ij}) = \begin{bmatrix} \frac{d^2}{d-1}(1+3K) & -\frac{d^2}{d-1}(3K) & 0 \\ -\frac{d^2}{d-1}(3K) & \frac{d^2}{d-2}\left(\frac{3c}{d-2}+3K\right) & -\frac{d^2}{d-1}\left(\frac{3cd}{d-2}\right) \\ 0 & -\frac{d^2}{d-1}\left(\frac{3cd}{d-2}\right) & \frac{d^2}{d-1}\left(\frac{cd^2}{d-2}\right) \end{bmatrix}.$$

The quadratic form is positive definite if the following determinates are all positive (see, e.g., [Apo74] p. 152).

$$\begin{aligned}
|a_{11}| &= \frac{d^2}{d-1}(1+3K) \\
\begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix} &= \left(\frac{d^2}{d-1}\right)^2 \left((1+3K)\left(\frac{9c}{d-2}+3K\right)-(3K)^2\right) \\
&= \left(\frac{d^2}{d-1}\right)^2 \left(\frac{9c}{d-2}+\frac{27cK}{d-2}+3K\right) \\
\begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix} &= \left(\frac{d^2}{d-1}\right)^3 (1+3K) \begin{vmatrix} a_{22} & a_{23} \\ a_{32} & a_{33} \end{vmatrix} - \left(\frac{d^2}{d-1}\right)^3 (-3K) \begin{vmatrix} a_{21} & a_{23} \\ a_{31} & a_{33} \end{vmatrix} \\
&= \left(\frac{d^2}{d-1}\right)^3 \left((1+3K)\left[3K\left(\frac{cd^2}{d-2}\right)\right]-(3K)^2\left[\frac{cd^2}{d-2}\right]\right) \\
&= \frac{3Kcd^8}{(d-1)^3(d-2)}.
\end{aligned}$$

In Section 4.4.3, we will prove Lemma 4.10 which states that $K > 0$. Thus, the quadratic form is positive definite, and the determinant D of (a_{ij}) is $\frac{3Kcd^8}{(d-1)^3(d-2)}$.

Now, we can apply the Laplace Method and get,

$$\begin{aligned}
\frac{\mathbf{E}(N^2)}{\mathbf{E}(N)^2} &\sim 3c^2n^3(2\pi)^{\frac{3}{2}}\left(\frac{3Kcd^8}{(d-1)^3(d-2)}\right)^{-\frac{1}{2}} n^{-\frac{3}{2}} g\left(\frac{1}{d},\frac{1}{d},\frac{1}{d^2}\right) \\
&\sim 3^{\frac{1}{2}}(2c\pi n)^{\frac{3}{2}} K^{-\frac{1}{2}} d^{-4}(d-1)^{\frac{3}{2}}(d-2)^{\frac{1}{2}} g\left(\frac{1}{d},\frac{1}{d},\frac{1}{d^2}\right).
\end{aligned}$$

$$\begin{aligned}
g\left(\frac{1}{d},\frac{1}{d},\frac{1}{d^2}\right) &= \Phi(3c,1)^{-1}\Phi\left(\frac{3c}{d},\frac{1}{d}\right)\Phi\left(\left(1-\frac{1}{d}\right)3c,\left(1-\frac{1}{d}\right)\right) \\
&\qquad (2\pi n)^{-\frac{3}{2}}c^{-1}\left(\frac{1}{d}\left(1-\frac{1}{d}\right)\left(1-\frac{3}{d}+\frac{2}{d^2}\right)\left(\frac{3}{d}-\frac{3}{d^2}\right)\frac{1}{d^2}\right)^{-\frac{1}{2}} \\
&= \Phi(3c,1)(2\pi n)^{-\frac{3}{2}}c^{-1}\left(\frac{3(d-1)^3(d-2)}{d^8}\right)^{-\frac{1}{2}} \qquad \text{by (4.1)} \\
&= \Phi(3c,1)(2\pi n(d-1))^{-\frac{3}{2}}c^{-1}d^4(3(d-2))^{-\frac{1}{2}}. \qquad (4.6)
\end{aligned}$$

From (4.2), $e^x = 1 + \frac{3cx}{3c-x}$. Using this, we can simplify K.

$$\begin{aligned} K &= \frac{c(e^x-1)^2}{(e^x-1)+3c(e^x-1-xe^x)} \\ &= \frac{c(3cx)^2}{(3cx)^2+3c(3cx)(3c-x)-3cx(3c-x)^2-(3cx)^2(3c-x)} \\ &= \frac{3c^2}{x(3c-1)-3c(3c-2)} \\ &= c(\Phi(3c,1))^2. \end{aligned} \tag{4.7}$$

Thus,

$$\begin{aligned} \frac{\mathbf{E}(N^2)}{\mathbf{E}(N)^2} &\sim 3^{\frac{1}{2}}(2c\pi n)^{\frac{3}{2}}K^{-\frac{1}{2}}d^{-4}(d-1)^{\frac{3}{2}}(d-2)^{\frac{1}{2}}g\left(\frac{1}{d},\frac{1}{d},\frac{1}{d^2}\right) \\ &\sim 3^{\frac{1}{2}}(2c\pi n)^{\frac{3}{2}}K^{-\frac{1}{2}}d^{-4}(d-1)^{\frac{3}{2}}(d-2)^{\frac{1}{2}}\Phi(3c,1)(2\pi n(d-1))^{-\frac{3}{2}}c^{-1}d^4(3(d-2))^{-\frac{1}{2}} \\ &\qquad \text{by (4.6)} \\ &\sim c^{\frac{1}{2}}K^{-\frac{1}{2}}\Phi(3c,1) \\ &\sim c^{\frac{1}{2}}\left(c(\Phi(3c,1))^2\right)^{-\frac{1}{2}}\Phi(3c,1) \qquad \text{by (4.7)} \\ &\sim 1 \end{aligned}$$

which completes the proof. □

4.4.3 Proof That $K > 0$ if $x > 0$.

Lemma 4.10 *If $x > 0$, $c > 0$, then $K = \frac{c(e^x-1)^2}{(e^x-1)^2+3c(e^x-xe^x-1)} > 0$.*

Proof. If $K < 0$, then the denominator of K must be less than 0. By (4.2), $3c = \frac{x(e^x-1)}{e^x-1-x}$, so we can expand the denominator.

$$\begin{aligned} (e^x-1)^2+3c(e^x-1-xe^x) &= (e^x-1)^2+\frac{x(e^x-1)}{e^x-1-x}(e^x-1-xe^x) \\ &= \frac{e^x-1}{e^x-1-x}((e^{2x}+1)-e^x(x^2+2)). \end{aligned}$$

Since $x > 0$, we know $\frac{e^x-1}{e^x-1-x} > 0$. Thus, if $K < 0$, we must have $e^{2x}+1 < e^x(x^2+2)$, and thus $e^x + e^{-x} < x^2+2$, for some value of $x > 0$. However, $e^x + e^{-x} = 2\cosh(x) = 2\left(1+\frac{x^2}{2!}+\frac{x^4}{4!}+\dots\right) > x^2+2$. □

4.4.4 An Approximation for Generalized Stirling Numbers of the Second Kind

Lemma 4.11

$$S(n,k,2) \sim \frac{n}{k!}\left(\frac{n}{z_0 e}\right)^n (e^{z_0}-1-z_0)^k \sqrt{\frac{nk}{z_0 k(n-k)-n(n-2k)}}$$

where z_0 is the positive real solution of the equation

$$\frac{k}{n}z_0 = \frac{e^{z_0}-1-z_0}{e^{z_0}-1}.$$

Proof. Hennecart, in [Hen94], gives the following approximation for the generalized Stirling number $S(n,k,r)$.

$$S(n,k,r) \sim \frac{n!}{k!(n-kr)!}\left(\frac{n-kr}{e}\right)^{n-kr} \frac{B^k(z_0,r)}{z_0^{n+1}}\sqrt{\frac{kt_0}{\phi''(z_0)}}$$

where $B(z,r) = e^z - \sum_{l=0}^{r-1}\frac{z^l}{l!}$, $\phi(z) = -n\ln z + k\ln B(z,r)$, $t_0 = \frac{n-kr}{k}$, and z_0 is the positive real solution of the equation $z_0\frac{B'(z_0,r)}{B(z_0,r)} = \frac{n}{k}$.

If we let $r=2$, then $B(z,2) = e^z - 1 - z$, and

$$\phi''(z) = \frac{n}{z^2} + k\frac{e^z(e^z-1-z)-(e^z-1)^2}{(e^z-1-z)^2}.$$

Since, $z_0\frac{e^{z_0}-1}{e^{z_0}-1-z_0} = \frac{n}{k}$, we have

$$\begin{aligned} e^{z_0} &= \frac{n+nz_0-kz_0}{n-kz_0} \\ e^{z_0}-1 &= \frac{nz_0}{n-kz_0} \\ e^{z_0}-1-z_0 &= \frac{kz_0^2}{n-kz_0}, \end{aligned}$$

and we can simplify $\phi''(z_0)$.

$$\begin{aligned} \phi''(z_0) &= \frac{n}{z_0^2} + k\frac{(n+nz_0-kz_0)kz_0^2-(nz_0)^2}{(kz_0)^2} \\ &= \frac{1}{kz_0^2}(z_0k(n-k)-n(n-2k)). \end{aligned}$$

Thus, using Stirling's approximation that $n! \sim \left(\frac{n}{e}\right)^n \sqrt{2\pi n}$, we get

$$
\begin{aligned}
S(n,k,2) &\sim \frac{n!}{k!(n-2k)!}\left(\frac{n-2k}{e}\right)^{n-2k}\frac{(e^{z_0}-1-z_0)^k}{z_0^{n+1}}\sqrt{\frac{kz_0^2(n-2k)}{z_0k(n-k)-n(n-2k)}} \\
&= \frac{\sqrt{n-2k}}{k!}\left[\frac{n!}{(n-2k)!}\right]\left(\frac{n-2k}{e}\right)^{n-2k}\frac{(e^{z_0}-1-z_0)^k}{z_0^n} \\
&\qquad \sqrt{\frac{k}{z_0k(n-k)-n(n-2k)}} \\
&\sim \frac{\sqrt{n-2k}}{k!}\left[\left(\frac{n}{e}\right)^n\left(\frac{e}{n-2k}\right)^{n-2k}\frac{\sqrt{n}}{\sqrt{n-2k}}\right]\left(\frac{n-2k}{e}\right)^{n-2k} \\
&\qquad \frac{(e^{z_0}-1-z_0)^k}{z_0^n}\sqrt{\frac{k}{z_0k(n-k)-n(n-2k)}} \\
&= \frac{1}{k!}\left(\frac{n}{z_0e}\right)^n(e^{z_0}-1-z_0)^k\sqrt{\frac{nk}{z_0k(n-k)-n(n-2k)}}
\end{aligned}
$$

which completes the proof. □

4.5 Extending the Threshold Results to $k > 3$

This section will extend the proof of the satisfiability threshold for random $(3, d)$-UE-CSP under the Maximum Hypothesis of Section 4.4 to all random (k, d)-UE-CSP with $k \geq 3$. The proof follows the exact same technique, but the resulting expressions are significantly more complicated. To simplify the presentation, some notation will be abused, and some steps will not be justified if the justification directly follows from similar steps in Section 4.4.

To use the same technique of Section 4.4, we need to generalize the Maximum Hypothesis to the case where $k \geq 3$.

Hypothesis 4.12 (General Maximum Hypothesis) *Let κ_k be the threshold for the appearance of a 2-core in a k-uniform hypergraph. Consider any integer constants $k \geq 3$*

and $d \geq 2$ and any real constant $\kappa_k < c \leq 1$. The function

$$\begin{aligned} f(\alpha, r, t_0, \ldots, t_k) &= \ln d - c\ln d + (1-\alpha)\ln(1-d) - \alpha\ln\alpha - (1-\alpha)\ln(1-\alpha) \\ &\quad - t_0 c\ln t_0 - \sum_{i=2}^{k} t_i c\ln t_i + \sum_{i=2}^{k-1} t_i c\binom{k}{i} + \sum_{i=2}^{k} t_i c\ln\left(\frac{(d-1)^{i-1}+(-1)^i}{d(d-1)^{i-1}}\right) \\ &\quad + \alpha\ln(e^z - 1 - z) - krc\ln z + krc\ln r - \ln(e^x - 1 - x) + kc\ln x \\ &\quad + (1-\alpha)\ln(e^y - 1 - y) - k(1-r)c\ln y + k(1-r)c\ln(1-r), \end{aligned}$$

where $\sum_{i=0}^{k} t_i = 1$, $\sum_{i=0}^{k-1}(k-i)t_i = kr$, and $x, y, z > 0$ are defined as

$$\frac{e^x - 1 - x}{e^x - 1} - \frac{x}{kc} = \frac{e^y - 1 - y}{e^y - 1} - \frac{y(1-\alpha)}{kc(1-r)} = \frac{e^z - 1 - z}{e^z - 1} - \frac{z\alpha}{kcr} = 0,$$

has a unique maximum in the region bounded by $0 \leq \alpha, r, t_i \leq 1$, for $i = 0 \ldots k$. Furthermore, the maximum is at the point

$$\alpha = \frac{1}{d}, r = \frac{1}{d}, t_0 = \frac{1}{d^{k-1}}, t_i = \binom{k}{i}\frac{(d-1)^i + (d-1)(-1)^i}{d^k}$$

for $0 \leq i \leq k$.

The function f of the General Maximum Hypothesis contains the exponential terms from the second moment calculation used to lower bound the satisfiability threshold, and if we set $k = 3$, the General Maximum Hypothesis becomes the Maximum Hypothesis of Section 4.3.

The results for random k-XOR-SAT in [MRTZ03] imply that the satisfiability threshold for a random 2-core with n' variables and $c'n'$ clauses is when $c' = 1$ for all $k \geq 3$. This section will prove that, under the General Maximum Hypothesis, this is indeed the case for all (k,d)-UE-CSP, $k \geq 2$.

To find the satisfiability threshold for each k, we can use the results from [Mol05] discussed in Section 4.4.1, and the result is the following theorem.

Theorem 4.13 *Under the General Maximum Hypothesis, the satisfiability threshold for random (k,d)-UE-CSP on n variables and cn clauses is at $c = c_k^*$ where x is the largest solution to $x = c_k^* k(1-e^{-x})^{k-1}$ and $x(1-e^{-x}) = k(1-e^{-x}-xe^{-x})$.*

Extending Theorem 4.3 to Theorem 4.13 follows from the following two lemmas. Let $\Psi_{n,m}$ denote the subset of $\Omega_{n,m}^{(k,d)}$ in which every variable lies in at least 2 constraints, and let $U_{n,m}^*$ denote a uniformly random member of $\Psi_{n,m}$.

Lemma 4.14 *For every $c > 1$, $U_{n,m=cn}^*$ is a.s. unsatisfiable.*

The proof of Lemma 4.14 is identical to the proof of Lemma 4.6.

Lemma 4.15 *Under the General Maximum Hypothesis, for every $c < 1$, $U_{n,m=cn}^*$ is a.s. satisfiable.*

Proof. Similar to the method of Section 4.4.2, we first compute the probability a constraint is satisfied by two different variable assignments σ and τ conditional on the number of variables in the clause for which each such variable is assigned the same value in τ as it is assigned in σ.

We begin with a general observation on the relation between uniquely extendible constraints that are of different sizes but over the same domain. Let C be a constraint of (k, d)-UE-CSP. For each $i \in \{0, \ldots, d-1\}$, let C_i be the set of tuples defined by

$$C_i = \{(x_2, \ldots, x_k) \mid (x_1, \ldots, x_k) \in C \text{ and } x_1 = i\}.$$

Note that each C_i is a constraint of $(k-1, d)$-UE-CSP because, in constraint C, for any setting of $x_2, \ldots, x_{k-1}$, there is a unique value for x_k that satisfies the constraint when $x_1 = i$. More generally, let $C_{\beta_1,\ldots,\beta_j}$ be the set of tuples defined by

$$C_{\beta_1,\ldots,\beta_j} = \{(x_{j+1}, \ldots, x_k) \mid (x_1, \ldots, x_k) \in C \text{ and } x_1 = \beta_1, \ldots, x_j = \beta_j\}, \tag{4.8}$$

and by the same arguments, $C_{\beta_1,\ldots,\beta_j}$ is a constraint of $(k-j, d)$-UE-CSP. Given a constraint C of (k, d)-UE-CSP and a constraint D of $(k-j, d)$-UE-CSP for $1 \le j \le k-2$, we call D a *subconstraint* of C if there exists $\beta_1, \ldots, \beta_j$ such that (4.8) holds with $D = C_{\beta_1,\ldots,\beta_j}$.

We will compute the probability a uniformly random (k, d)-UE-CSP constraint on a k-clause permits two different particular variable assignments that do not agree on exactly i variables of the clause. First, we fix $2 \leq i \leq k$. For a k-clause $(v_1, \dots, v_k)$, let $\alpha_1, \dots, \alpha_k$ be the values assigned by the first assignment to $v_1, \dots, v_k$, and let $\beta_1, \dots, \beta_i, \alpha_{i+1}, \dots, \alpha_k$ be the values assigned by the second assignment to $v_1, \dots, v_k$ where $\beta_j \neq \alpha_j$ for all $j \leq i$.

Consider a constraint to be a list of acceptable k-tuples of values to the variables of the clause. We will choose a uniformly random constraint, C, and expose the acceptable tuples of C one at a time. Assume $(\alpha_1, \dots, \alpha_k)$ is permitted by the constraint C. By symmetry, exactly $\frac{1}{d}$ of the constraints will permit this tuple. Now let us consider the subconstraints C_{α_1} and C_{β_1}. By symmetry, C_{α_1} could be any $(k-1, d)$-UE-CSP constraint that permits the tuple $(\alpha_2, \dots, \alpha_k)$, and C_{β_1} could be any $(k-1, d)$-UE-CSP constraint that does not permit the tuple $(\alpha_2, \dots, \alpha_k)$. In particular, C_{β_1} will permit the tuple $(\alpha_2, \dots, \alpha_{k-1}, \gamma_1)$ for $\gamma_1 \neq \alpha_k$, and by symmetry, all $d-1$ choices for γ_1 are equally likely.

Now, let us repeat this argument focusing on the subconstraint C_{β_1} and consider its subconstraints C_{β_1,α_2} and C_{β_1,β_2}. Without knowing anything else about the tuples C permits, because C_{β_1} can be any $(k-1, d)$-UE-CSP constraint that does not permit $(\alpha_2, \dots, \alpha_k)$, by symmetry C_{β_1,α_2} can be any $(k-2, d)$-UE-CSP constraint that does not permit $(\alpha_3, \dots, \alpha_k)$, and C_{β_1,β_2} can be any $(k-2, d)$-UE-CSP constraint. Now we expose the fact that C permits the tuple $(\beta_1, \alpha_2, \dots, \alpha_{k-1}, \gamma_1)$ where $\gamma_1 \neq \alpha_k$. By symmetry, C_{β_1,α_2} can be any $(k-2, d)$-UE-CSP constraint that permits $(\alpha_3, \dots, \alpha_{k-1}, \gamma_1)$, and C_{β_1,β_2} can be any $(k-2, d)$-UE-CSP that does not permit $(\alpha_3, \dots, \alpha_{k-1}, \gamma_1)$. In particular, C_{β_1,β_2} will permit the tuple $(\alpha_3, \dots, \alpha_{k-1}, \gamma_2)$ for $\gamma_2 \neq \gamma_1$, and by symmetry, all $d-1$ choices for γ_2 are equally likely. Now we expose the fact that C permits $(\beta_1, \beta_2, \alpha_3, \dots, \alpha_{k-1}, \gamma_2)$, and we can then repeat this argument for the subconstraints of C_{β_1,β_2}.

We continue this process of exposing acceptable tuples of C until, for $j = 1$ to i, we have exposed the tuples $(\beta_1, \dots, \beta_j, \alpha_{j+1}, \dots, \alpha_{k-1}, \gamma_j)$ in that order. For each j, let R_j be the probability that $\gamma_j = \alpha_k$. To calculate R_j, by the symmetry argument above, immediately after we exposed $(\beta_1, \ldots, \beta_{j-1}, \alpha_j, \ldots, \alpha_{k-1}, \gamma_{j-1})$, the subconstraint $C_{\beta_1,\ldots,\beta_j}$ could be any $(k-j,d)$-UE-CSP constraint that did not permit the tuple $(\alpha_{j+1}, \ldots, \alpha_{k-1}, \gamma_{j-1})$, and in particular, $C_{\beta_1,\ldots,\beta_j}$ will permit the tuple $(\alpha_{j+1}, \ldots, \alpha_{k-1}, \gamma_j)$ for $\gamma_j \neq \gamma_{j-1}$, and all $d-1$ choices for γ_j are equally likely. So the probability $\gamma_j = \alpha_k$ is $\frac{1}{d-1}$ conditioning on the event that $\gamma_{j-1} \neq \alpha_k$, and the probability of this latter event is $1 - R_{j-1}$. That gives us the following recurrence: $R_i = \frac{1}{d-1}(1 - R_{i-1})$. Finally, to get the probability both $(\alpha_1, \ldots, \alpha_k)$ and $(\beta_1, \ldots, \beta_i, \alpha_{i+1}, \ldots, \alpha_k)$ are acceptable tuples, we have to multiply R_i by the probability $\frac{1}{d}$ that our original assumption holds, i.e. that $(\alpha_1, \ldots, \alpha_k)$ is in the constraint. Therefore the probability a uniformly random (k,d)-UE-CSP constraint permits two different particular variable assignments that disagree on exactly i variables of the clause is $\frac{1}{d}\frac{1}{d-1}(1 - R_{i-1})$. To simplify the presentation below, we will denote this probability $\frac{P_i}{d}$. Note that $P_i = R_i$, and we have

$$\begin{aligned}
\frac{P_0}{d} &= \frac{1}{d} \\
\frac{P_i}{d} &= \frac{1}{d}\frac{1}{d-1}\left(1 - P_{i-1}\right).
\end{aligned}$$

As a result,

$$\frac{\mathbf{E}(N^2)}{\mathbf{E}(N)^2} = \sum_{\alpha}\sum_{r}\sum_{T_i} d^n (d-1)^{(1-\alpha)n} \binom{n}{\alpha n} \frac{m!}{T_0! T_2! \ldots T_k!} \left(\frac{P_0}{d}\right)^{T_0} \prod_{i=2}^{k} \left(\frac{P_i}{d}\right)^{T_i} \prod_{i=2}^{k-1} \binom{k}{i}^{T_i}$$
$$\times \frac{S(krm, \alpha n, 2)(\alpha n)! S(k(1-r)m, (1-\alpha)n, 2)((1-\alpha)n)!}{S(km, n, 2) n!} \qquad (4.9)$$

where $\alpha \in I_n$ and $r \in I_{km}$ and where the final sum is over all $T_0, \ldots, T_k$ such that

$$T_1 = 0,$$
$$T_0 + \sum_{i=2}^{k} T_i = m, \text{and} \qquad (4.10)$$
$$kT_0 + \sum_{i=2}^{k} (k-i) T_i = krm. \qquad (4.11)$$

Since $\sum_{i=0}^{k} T_i = m$, we can simplify (4.9) slightly:

$$\frac{\mathbf{E}(N^2)}{\mathbf{E}(N)^2} = \sum_{\alpha} \sum_{r} \sum_{T_i} d^{n-m}(d-1)^{(1-\alpha)n} \binom{n}{\alpha n} \frac{m!}{T_0! T_2! \dots T_k!} P_0^{T_0} \prod_{i=2}^{k} (P_i)^{T_i} \prod_{i=2}^{k-1} \binom{k}{i}^{T_i}$$
$$\times \frac{S(krm, \alpha n, 2)(\alpha n)! S(k(1-r)m, (1-\alpha)n, 2)((1-\alpha)n)!}{S(km, n, 2)n!}. \qquad (4.12)$$

Let $t_i = \frac{T_i}{m}$. We can write (4.12) as

$$\frac{\mathbf{E}(N^2)}{\mathbf{E}(N)^2} = \sum_{\alpha} \sum_{r} \sum_{t_i} g(\alpha, r, t_0, \dots, t_k) e^{nf(\alpha, r, t_0, \dots, t_k)}$$

where

$$g(\alpha, r, t_0, \dots, t_k) = (2\pi n)^{-\frac{k}{2}} c^{-\frac{k-1}{2}} (\alpha(1-\alpha) t_0 t_2 t_3 \dots t_k)^{-\frac{1}{2}}$$
$$\times \Phi(krm, \alpha n) \Phi(k(1-r)m, (1-\alpha)n) \Phi^{-1}(km, n)$$
$$f(\alpha, r, t_0, \dots, t_k) = \ln d - c \ln d + (1-\alpha)\ln(d-1) - \alpha \ln \alpha - (1-\alpha)\ln(1-\alpha)$$
$$- t_0 c \ln t_0 - \sum_{i=2}^{k} t_i c \ln t_i + \sum_{i=2}^{k-1} t_i c \binom{k}{i} + \sum_{i=2}^{k} t_i c \ln P_i$$
$$+ \alpha \ln(e^z - 1 - z) - krc \ln z + krc \ln r - \ln(e^x - 1 - x) + kc \ln x$$
$$+ (1-\alpha)\ln(e^y - 1 - y) - k(1-r)c \ln y + k(1-r)c \ln(1-r)$$

such that, from Lemma 4.11, $x, y, z > 0$ and

$$\frac{e^x - 1 - x}{e^x - 1} - \frac{x}{kc} = \frac{e^y - 1 - y}{e^y - 1} - \frac{y(1-\alpha)}{kc(1-r)} = \frac{e^z - 1 - z}{e^z - 1} - \frac{z\alpha}{kcr} = 0. \qquad (4.13)$$

Substitute $t_{k-1} = kr - kt_0 - \sum_{i=2}^{k-2}(k-i)t_i$ and $t_k = 1 - kr + (k-1)t_0 + \sum_{i=2}^{k-2}(k-i-1)t_i$, and find the partial derivatives as before:

$$\frac{\partial f}{\partial \alpha} = -\ln(d-1) - \ln\alpha + \ln(1-\alpha) + \ln(e^z-1-z) - \ln(e^y-1-y),$$

$$\begin{aligned}\frac{\partial f}{\partial r} &= D_r[krc\ln r + k(1-r)c\ln(1-r)\alpha\ln(e^z-1-z) - krc\ln z + (1-\alpha)(e^y-1-y)\\ &\qquad - k(1-r)c\ln y - t_kc\ln t_k - t_{k-1}c\ln t_{k-1} + t_{k-1}c\ln k\\ &\qquad - t_kc\ln P_k + t_{k-1}c\ln P_{k-1}]\\ &= kc\ln r - kc\ln(1-r) - kc\ln z + kc\ln y + kc\ln t_k - kc\ln t_{k-1} + kc\ln k\\ &\quad - kc\ln P_k + kc\ln P_{k-1},\end{aligned}$$

$$\begin{aligned}\frac{\partial f}{\partial t_0} &= D_{t_0}[-t_0c\ln t_0 - t_{k-1}c\ln t_{k-1} - t_kc\ln t_k + t_{k-1}c\ln k + t_{k-1}c\ln P_{k-1} + t_kc\ln P_k]\\ &= -c\ln t_0 + kc\ln t_{k-1} - (k-1)c\ln t_k - kc\ln k - kc\ln P_{k-1} + (k-1)c\ln P_k,\end{aligned}$$

$$\begin{aligned}\frac{\partial f}{\partial t_i} &= D_{t_i}[-t_ic\ln t_i - t_{k-1}c\ln t_{k-1} - t_kc\ln t_k + t_{k-1}c\ln k\\ &\qquad + t_ic\ln P_i + t_{k-1}c\ln P_{k-1} + t_kc\ln P_k]\\ &= -c\ln t_i + (k-i)c\ln t_{k-1} - (k-i-1)c\ln t_k + c\ln\binom{k}{k-i} - (k-i)c\ln k\\ &\quad + c\ln P_i - (k-i)c\ln P_{k-1} + (k-i-1)c\ln P_k.\end{aligned}$$

Setting the derivatives to 0 yields the following equations

$$\frac{1-\alpha}{\alpha} = (d-1)\frac{e^y-1-y}{e^z-1-z} \tag{4.14}$$

$$\frac{1-r}{r} = \frac{y}{z}\frac{t_k}{t_{k-1}}\frac{Q_{k-1}}{Q_k}(d-1)k \tag{4.15}$$

$$t_0 = \frac{(t_{k-1})^k}{(t_k)^{k-1}}\frac{(Q_k)^{k-1}}{(Q_{k-1})^k}\frac{1}{(d-1)k^k} \tag{4.16}$$

$$t_i = \frac{(t_{k-1})^{k-i}}{(t_k)^{k-i-1}}\frac{Q_i\,(Q_k)^{k-i-1}}{(Q_{k-1})^{k-i}}\frac{\binom{k}{k-i}}{k^{k-i}} \tag{4.17}$$

where (4.17) is for $2 \le i \le k-2$ and with $Q_2 = 1$ and $Q_j = (d-1)^{j-2} - Q_{j-1}$, for $j > 2$.

Note that

$$Q_i = \frac{(d-1)^{i-1} + (-1)^i}{d}. \tag{4.18}$$

From the observation that applying (4.18) yields $Q_1 = 0$ and $Q_0 = \frac{1}{d-1}$ and the observation that (4.17) holds when $i = k$ and $i = k-1$, we can replace (4.16) and (4.17) by

$$t_i = \frac{(t_{k-1})^{k-i}}{(t_k)^{k-i-1}} \frac{Q_i\,(Q_k)^{k-i-1}}{(Q_{k-1})^{k-i}} \frac{\binom{k}{k-i}}{k^{k-i}} \tag{4.19}$$

for all $0 \leq i \leq k$.

Analyzing the equations and again guessing that $x = y = z$ at the maximum of f yields the following values

$$\alpha = \frac{1}{d}, r = \frac{1}{d}, t_0 = \frac{1}{d^{k-1}}, t_i = \binom{k}{i}(d-1)Q_i\frac{1}{d^{k-1}} \tag{4.20}$$

for $2 \leq i \leq k-2$.

To complete the proof of Lemma 4.15, we need the following steps.

1. Verify (4.20) is a possible maximum of f and has the desired value.

2. Form the negative Hessian matrix of f at (4.20).

3. Verify that (4.20) is a local maximum of f by showing that the negative Hessian matrix of f is positive definite, and get its determinate.

4. Calculate g at this maximum.

5. Assume from the General Maximum Hypothesis that (4.20) is the unique maximum of f, and use the Laplace Method to approximate $\frac{\mathbf{E}(N^2)}{\mathbf{E}(N)^2}$.

Step 1: Verify (4.20) is a possible maximum of f and has the desired value.

Let $t_k = \frac{(d-1)Q_k}{d^{k-1}}$ and let $t_{k-1} = \frac{k(d-1)Q_{k-1}}{d^{k-1}}$. Plugging these values and (4.20) into (4.14)-(4.17) shows that each equation is satisfied and (4.20) is a maximum for f.

Let $\tau = (\alpha = \frac{1}{d}, r = \frac{1}{d}, t_0 = \frac{1}{d^{k-1}}, \tau_i = \binom{k}{i}(d-1)Q_i\frac{1}{d^{k-1}})$ for $i = 2, \ldots, k$.

$$f(\tau) = 2\ln d - c\ln d - kc\ln d + \frac{(k-1)c}{d^{k-1}}\ln d + \frac{k(d-1)c}{d}\ln(d-1) - \sum_{i=2}^{k}\binom{k}{i}\frac{Q_i(d-1)c}{d^{k-1}}\ln\left(\frac{(d-1)^i}{d^{k-1}}\right).$$

We use the following claims.

Claim 4.16

$$\sum_{i=2}^{k}\binom{k}{i}(d-1)Q_i = d^{k-1} - 1.$$

Proof.

$$\begin{aligned}
\sum_{i=2}^{k}\binom{k}{i}(d-1)Q_i &= \frac{d-1}{d}\sum_{i=2}^{k}\binom{k}{i}((d-1)^{i-1} + (-1)^i) && \text{by (4.18)}\\
&= \frac{1}{d}\left(\sum_{i=0}^{k}\binom{k}{i}(d-1)^i - k(d-1) - 1\right)\\
&\quad + \frac{d-1}{d}\left(\sum_{i=0}^{k}\binom{k}{i}(-1)^i + k - 1\right)\\
&= \frac{1}{d}(d^k - k(d-1) - 1) + \frac{d-1}{d}(k-1)\\
&= d^{k-1} - 1.
\end{aligned}$$

□

Claim 4.17

$$\sum_{i=2}^{k} i\binom{k}{i}\frac{Q_i}{d^{k-1}} = \frac{k}{d}.$$

Proof.

$$\begin{aligned}
\sum_{i=2}^{k} i\binom{k}{i}\frac{Q_i}{d^{k-1}} &= \frac{k}{d^k}\left(\sum_{i=2}^{k}\binom{k-1}{i-1}(d-1)^{i-1} + \sum_{i=2}^{k}\binom{k-1}{i-1}(-1)^i\right) && \text{by (4.18)}\\
&= \frac{k}{d^k}\left(\sum_{i=0}^{k-1}\binom{k-1}{i}(d-1)^i - 1 - \sum_{i=0}^{k-1}\binom{k-1}{i}(-1)^i + 1\right)\\
&= \frac{k}{d^k}d^{k-1}\\
&= \frac{k}{d}.
\end{aligned}$$

□

From Claim 4.16,

$$\sum_{i=2}^{k}\binom{k}{i}\frac{Q_i(d-1)c}{d^{k-1}}\ln\left(d^{k-1}\right)=\frac{d^{k-1}-1}{d^k}(k-1)c\ln d,$$

and from Claim 4.17,

$$\sum_{i=2}^{k}\binom{k}{i}\frac{Q_i(d-1)c}{d^{k-1}}\ln\left((d-1)^i\right)=\frac{k(d-1)c}{d}\ln(d-1).$$

As a result, we have

$$f(\tau)=2\ln d-2c\ln d$$

as desired.

Step 2: Form the negative Hessian matrix of f at (4.20).

The second partial derivatives of f are:

$$\begin{aligned}
f_{\alpha\alpha}&=-\frac{1}{\alpha}-\frac{1}{1-\alpha}+\frac{e^z-1}{e^z-1-z}\frac{\partial z}{\partial\alpha}-\frac{e^y-1}{e^y-1-y}\frac{\partial y}{\partial\alpha}\\
f_{\alpha r}&=\frac{e^z-1}{e^z-1-z}\frac{\partial z}{\partial r}-\frac{e^y-1}{e^y-1-y}\frac{\partial y}{\partial r}\\
f_{\alpha t_0}&=0\\
f_{\alpha t_i}&=0\\
f_{r\alpha}&=\frac{-kc}{z}\frac{\partial z}{\partial\alpha}+\frac{kc}{y}\frac{\partial y}{\partial\alpha}\\
f_{rr}&=\frac{kc}{r}+\frac{kc}{1-r}-\frac{kc}{z}\frac{\partial z}{\partial r}+\frac{kc}{y}\frac{\partial y}{\partial r}-\frac{k^2c}{t_k}-\frac{k^2c}{t_{k-1}}\\
f_{rt_0}&=\frac{k(k-1)c}{t_k}+\frac{k^2c}{t_{k-1}}\\
f_{rt_i}&=\frac{k(k-i-1)c}{t_k}+\frac{k(k-i)c}{t_{k-1}}
\end{aligned} \tag{4.21}$$

$$\begin{aligned}
f_{t_0\alpha} &= 0 \\
f_{t_0r} &= \frac{k(k-1)c}{t_k} + \frac{k^2c}{t_{k-1}} \\
f_{t_0t_0} &= -\frac{c}{t_0} - \frac{(k-1)^2c}{t_k} - \frac{k^2c}{t_{k-1}} \\
f_{t_0t_i} &= -\frac{(k-1)(k-i-1)c}{t_k} - \frac{k(k-i)c}{t_k}, && \text{for } i \neq 0
\end{aligned}$$

$$\begin{aligned}
f_{t_i\alpha} &= 0 \\
f_{t_ir} &= \frac{k(k-i-1)c}{t_k} + \frac{k(k-i)c}{t_{k-1}} \\
f_{t_it_0} &= -\frac{(k-1)(k-i-1)c}{t_k} - \frac{k(k-i)c}{t_k} && \text{for } i \neq 0 \\
f_{t_it_i} &= -\frac{c}{t_i} - \frac{(k-i-1)^2c}{t_k} - \frac{(k-i)^2c}{t_{k-1}} \\
f_{t_it_j} &= -\frac{(k-i-1)(k-j-1)c}{t_k} - \frac{(k-i)(k-j)c}{t_{k-1}} && \text{for } i \neq j
\end{aligned}$$

where, from (4.13),

$$\begin{aligned}
\frac{\partial z}{\partial \alpha} &= \frac{-z(e^z-1)^2}{\alpha(e^z-1)^2 + krc(e^z(e^z-1-z) - (e^z-1)^2)} \\
\frac{\partial z}{\partial r} &= \frac{\alpha z(e^z-1)^2}{r[\alpha(e^z-1)^2 + krc(e^z(e^z-1-z) - (e^z-1)^2)]} \\
\frac{\partial y}{\partial \alpha} &= \frac{y(e^y-1)^2}{(1-\alpha)(e^y-1)^2 + k(1-r)c(e^y(e^y-1-y) - (e^y-1)^2)} \\
\frac{\partial y}{\partial r} &= \frac{-(1-\alpha)y(e^y-1)^2}{(1-r)[(1-\alpha)(e^y-1)^2 + k(1-r)c(e^y(e^y-1-y) - (e^y-1)^2)]}.
\end{aligned}$$

Thus,

$$\begin{aligned}
f_{\alpha\alpha}(\tau) &= -\frac{d^2}{d-1} - \frac{kd^2}{d-1}K \\
f_{\alpha r}(\tau) &= kK\frac{d^2}{d-1} \\
f_{\alpha t_0}(\tau) &= 0 \\
f_{\alpha t_i}(\tau) &= 0 \\
f_{r\alpha}(\tau) &= kK\frac{d^2}{d-1} \\
f_{rr}(\tau) &= kc\frac{d^2}{d-1} - kK\frac{d^2}{d-1} - \frac{d^{k-1}c}{d-1}\left(\frac{k^2}{Q_k} + \frac{k}{Q_{k-1}}\right)
\end{aligned}$$

$$\begin{aligned}
f_{rt_0}(\tau) &= \frac{d^{k-1}c}{d-1}\left(\frac{k(k-1)}{Q_k} + \frac{k}{Q_{k-1}}\right) \\
f_{rt_i}(\tau) &= \frac{d^{k-1}c}{d-1}\left(\frac{k(k-i-1)}{Q_k} + \frac{(k-i)}{Q_{k-1}}\right) \\
f_{t_0\alpha}(\tau) &= 0 \\
f_{t_0r}(\tau) &= \frac{d^{k-1}c}{d-1}\left(\frac{k(k-1)}{Q_k} + \frac{k}{Q_{k-1}}\right) \\
f_{t_0t_0}(\tau) &= -cd^{k-1} - \frac{d^{k-1}c}{d-1}\left(\frac{(k-1)^2}{Q_k} + \frac{k}{Q_{k-1}}\right) \\
f_{t_0t_i}(\tau) &= -\frac{d^{k-1}c}{d-1}\left(\frac{(k-1)(k-i-1)}{Q_k} + \frac{k-i}{Q_{k-1}}\right) && \text{for } i \neq 0 \\
f_{t_i\alpha}(\tau) &= 0 \\
f_{t_ir}(\tau) &= \frac{d^{k-1}c}{d-1}\left(\frac{k(k-i-1)}{Q_k} + \frac{(k-i)}{Q_{k-1}}\right) \\
f_{t_it_0}(\tau) &= -\frac{d^{k-1}c}{d-1}\left(\frac{(k-1)(k-i-1)}{Q_k} + \frac{k-i}{Q_{k-1}}\right) && \text{for } i \neq 0 \\
f_{t_it_j}(\tau) &= -\frac{d^{k-1}c}{d-1}\left(\frac{(k-i-1)(k-j-1)}{Q_k} + \frac{(k-i)(k-j)}{kQ_{k-1}}\right) && \text{for } i \neq j \\
f_{t_it_i}(\tau) &= -\frac{d^{k-1}c}{d-1}\left(\frac{1}{\binom{k}{i}Q_i} + \frac{(k-i-1)^2}{Q_k} + \frac{(k-i)^2}{kQ_{k-1}}\right)
\end{aligned}$$

where $K = \frac{c(e^x-1)^2}{(e^x-1)^2+kc(e^x-xe^x-1)}$, and thus the negative Hessian matrix is the matrix

$$(a_{ij}) =$$

$$\begin{bmatrix}
\frac{d^2}{d-1}(1+kK) & -\frac{d^2}{d-1}kK & 0 & \cdots \\
-\frac{d^2}{d-1}kK & \frac{d^2k}{d-1}(K-c) + \frac{d^{k-1}c}{d-1}\left(\frac{k^2}{Q_k} + \frac{k^2}{kQ_{k-1}}\right) & -\frac{d^{k-1}c}{d-1}\left(\frac{k(k-1)}{Q_k} + \frac{k^2}{kQ_{k-1}}\right) & \cdots \\
0 & -\frac{d^{k-1}c}{d-1}\left(\frac{k(k-1)}{Q_k} + \frac{k^2}{kQ_{k-1}}\right) & cd^{k-1} + \frac{d^{k-1}c}{d-1}\left(\frac{(k-1)^2}{Q_k} + \frac{k^2}{kQ_{k-1}}\right) & \cdots \\
0 & -\frac{d^{k-1}c}{d-1}\left(\frac{k(k-3)}{Q_k} + \frac{k(k-2)}{kQ_{k-1}}\right) & \frac{d^{k-1}c}{d-1}\left(\frac{(k-1)(k-3)}{Q_k} + \frac{k(k-2)}{kQ_{k-1}}\right) & \cdots \\
0 & -\frac{d^{k-1}c}{d-1}\left(\frac{k(k-4)}{Q_k} + \frac{k(k-3)}{kQ_{k-1}}\right) & \frac{d^{k-1}c}{d-1}\left(\frac{(k-1)(k-4)}{Q_k} + \frac{k(k-3)}{kQ_{k-1}}\right) & \cdots \\
0 & -\frac{d^{k-1}c}{d-1}\left(\frac{k(k-5)}{Q_k} + \frac{k(k-4)}{kQ_{k-1}}\right) & \frac{d^{k-1}c}{d-1}\left(\frac{(k-1)(k-5)}{Q_k} + \frac{k(k-4)}{kQ_{k-1}}\right) & \cdots \\
\vdots & \vdots & \vdots & \ddots
\end{bmatrix}$$

where

$$a_{ii} = \frac{d^{k-1}c}{d-1}\left(\frac{k}{\binom{k}{i-2}Q_{i-2}} + \frac{(k-(i-2)-1)^2}{Q_k} + \frac{(k-(i-2))^2}{kQ_{k-1}}\right)$$

for $4 \leq i \leq k$.

Step 3: Verify that (4.20) is a local maximum of f by showing that the negative Hessian matrix of f is positive definite, and get its determinate.

To simplify (a_{ij}), multiply the ith row for $i \geq 4$ by k and add the second row multiplied by $(k-i+2)$. Also, add the second row to the third row. This yields the matrix $(b_{ij}) =$

$$\begin{bmatrix}
\frac{d^2}{d-1}(1+kK) & -\frac{d^2}{d-1}kK & 0 & \cdots \\
-\frac{d^2}{d-1}kK & \frac{d^2k}{d-1}(K-c)+\frac{d^{k-1}c}{d-1}\left(\frac{k^2}{Q_k}+\frac{k^2}{kQ_{k-1}}\right) & -\frac{d^{k-1}c}{d-1}\left(\frac{k(k-1)}{Q_k}+\frac{k^2}{kQ_{k-1}}\right) & \cdots \\
-\frac{d^2}{d-1}(kK) & \frac{d^2}{d-1}(kK-kc)+\frac{d^{k-1}c}{d-1}\cdot\frac{k}{Q_k} & cd^{k-1}-\frac{d^{k-1}c}{d-1}\cdot\frac{k-1}{Q_k} & \cdots \\
-\frac{d^2}{d-1}(kK)(k-2) & \frac{d^2}{d-1}(kK-kc)(k-2)+\frac{d^{k-1}c}{d-1}\cdot\frac{k^2}{Q_k} & -\frac{d^{k-1}c}{d-1}\cdot\frac{k(k-1)}{Q_k} & \cdots \\
-\frac{d^2}{d-1}(kK)(k-3) & \frac{d^2}{d-1}(kK-kc)(k-3)+\frac{d^{k-1}c}{d-1}\cdot\frac{k^2}{Q_k} & -\frac{d^{k-1}c}{d-1}\cdot\frac{k(k-1)}{Q_k} & \cdots \\
-\frac{d^2}{d-1}(kK)(k-4) & \frac{d^2}{d-1}(kK-kc)(k-4)+\frac{d^{k-1}c}{d-1}\cdot\frac{k^2}{Q_k} & -\frac{d^{k-1}c}{d-1}\cdot\frac{k(k-1)}{Q_k} & \cdots \\
\vdots & \vdots & \vdots & \ddots
\end{bmatrix}$$

where

$$b_{ii} = \frac{d^{k-1}c}{d-1}\left(\frac{k}{\binom{k}{i-2}Q_{i-2}} - \frac{k(k-(i-2)-1)}{Q_k}\right)$$

for $4 \leq i \leq k$. Note that $D = \det(\,(a_{ij})\,) = k^{-(k-3)}\det(\,(b_{ij})\,)$.

Now, add the appropriate multiple of the first row to each other row to yield

$$\begin{bmatrix}
\frac{d^2}{d-1}(1+kK) & -\frac{d^2}{d-1}kK & 0 & \cdots \\
\frac{d^2}{d-1} & \frac{d^2}{d-1}(-kc)+\frac{d^{k-1}c}{d-1}\left(\frac{k^2}{Q_k}+\frac{k^2}{kQ_{k-1}}\right) & -\frac{d^{k-1}c}{d-1}\left(\frac{k(k-1)}{Q_k}+\frac{k^2}{kQ_{k-1}}\right) & \cdots \\
\frac{d^2}{d-1} & \frac{d^2}{d-1}(-kc)+\frac{d^{k-1}c}{d-1}\cdot\frac{k}{Q_k} & cd^{k-1}-\frac{d^{k-1}c}{d-1}\cdot\frac{k-1}{Q_k} & \cdots \\
\frac{d^2}{d-1}(k-2) & \frac{d^2}{d-1}(-kc)(k-2)+\frac{d^{k-1}c}{d-1}\cdot\frac{k^2}{Q_k} & -\frac{d^{k-1}c}{d-1}\cdot\frac{k(k-1)}{Q_k} & \cdots \\
\frac{d^2}{d-1}(k-3) & \frac{d^2}{d-1}(-kc)(k-3)+\frac{d^{k-1}c}{d-1}\cdot\frac{k^2}{Q_k} & -\frac{d^{k-1}c}{d-1}\cdot\frac{k(k-1)}{Q_k} & \cdots \\
\frac{d^2}{d-1}(k-4) & \frac{d^2}{d-1}(-kc)(k-4)+\frac{d^{k-1}c}{d-1}\cdot\frac{k^2}{Q_k} & -\frac{d^{k-1}c}{d-1}\cdot\frac{k(k-1)}{Q_k} & \cdots \\
\vdots & \vdots & \vdots & \ddots
\end{bmatrix}.$$

Subtracting k times the third row from each following row and the second row yields

$$\begin{bmatrix}
\frac{d^2}{d-1}(1+kK) & -\frac{d^2}{d-1}kK & 0 & \cdots \\
\frac{d^2}{d-1}(1-k) & \frac{d^2}{d-1}(-kc)(1-k)+\frac{d^{k-1}c}{d-1}\left(\frac{k^2}{kQ_{k-1}}\right) & -kcd^{k-1}-\frac{d^{k-1}c}{d-1}\left(\frac{k^2}{kQ_{k-1}}\right) & \cdots \\
\frac{d^2}{d-1} & \frac{d^2}{d-1}(-kc)+\frac{d^{k-1}c}{d-1}\cdot\frac{k}{Q_k} & cd^{k-1}-\frac{d^{k-1}c}{d-1}\cdot\frac{k-1}{Q_k} & \cdots \\
-\frac{d^2}{d-1}2 & \frac{d^2}{d-1}(2kc) & -kcd^{k-1} & \cdots \\
-\frac{d^2}{d-1}3 & \frac{d^2}{d-1}(3kc) & -kcd^{k-1} & \cdots \\
-\frac{d^2}{d-1}4 & \frac{d^2}{d-1}(4kc) & -kcd^{k-1} & \cdots \\
\vdots & \vdots & \vdots & \ddots
\end{bmatrix}$$

where, for row and column i, j with $4 \le i, j \le k$, the entries are 0 for $i \neq j$ and

$$\frac{d^{k-1}c}{d-1}\left(\frac{k}{\binom{k}{i-2}Q_{i-2}}\right)$$

for $i = j$.

Adding kc times the first column to the second column yields

$$\begin{bmatrix}
\frac{d^2}{d-1}(1+kK) & \frac{d^2}{d-1}(kc+k^2cK-kK) & 0 & \cdots \\
\frac{d^2}{d-1}(1-k) & \frac{d^{k-1}c}{d-1}\left(\frac{k^2}{kQ_{k-1}}\right) & -kcd^{k-1}-\frac{d^{k-1}c}{d-1}\left(\frac{k^2}{kQ_{k-1}}\right) & \cdots \\
\frac{d^2}{d-1} & \frac{d^{k-1}c}{d-1}\cdot\frac{k}{Q_k} & cd^{k-1}-\frac{d^{k-1}c}{d-1}\cdot\frac{k-1}{Q_k} & \cdots \\
-\frac{d^2}{d-1}2 & 0 & -kcd^{k-1} & \cdots \\
-\frac{d^2}{d-1}3 & 0 & -kcd^{k-1} & \cdots \\
-\frac{d^2}{d-1}4 & 0 & -kcd^{k-1} & \cdots \\
\vdots & \vdots & \vdots & \ddots
\end{bmatrix}.$$

Finally, we add the appropriate multiple of each row after the third row to the second and third rows and use the diagonal elements to eliminate the first and third columns below the third row. The end result is the matrix (c_{ij}) of the form

$$(c_{ij}) = \begin{bmatrix} L & 0 \\ 0 & M \end{bmatrix}$$

where L is a 3×3 matrix and M is a $(k-3)\times(k-3)$ diagonal matrix. Note that

$$D = \det\left(\,(a_{ij})\,\right) = k^{-(k-3)}\det(L)\det(M).$$

The diagonal elements of M are

$$\frac{d^{k-1}c}{d-1}\left(\frac{k}{\binom{k}{i}Q_i}\right)$$

for $2 \le i \le k-2$. Thus

$$\det(M) = \left(\prod_{i=2}^{k-2} \frac{d^{k-1}c}{d-1} \cdot \frac{k}{\binom{k}{i}Q_i}\right).$$

The elements of L are

$$\begin{aligned}
L_{1,1} &= \frac{d^2}{d-1}(1+kK)\\
L_{1,2} &= \frac{d^2}{d-1}(kc+k^2cK-kK)\\
L_{1,3} &= 0\\
L_{2,1} &= \frac{d^2}{d-1}(1-k) - \frac{d^2}{d-1}\cdot\frac{1}{kQ_{k-1}}\sum_{i=2}^{k-2} Q_i\binom{k}{i} i(k-1)\\
L_{2,2} &= \frac{d^{k-1}c}{d-1}\left(\frac{k^2}{kQ_{k-1}}\right)\\
L_{2,3} &= -kcd^{k-1} - \frac{d^{k-1}c}{d-1}\left(\frac{k^2}{kQ_{k-1}}\right) - \frac{cd^{k-1}}{Q_{k-1}}\sum_{i=2}^{k-2} Q_i\binom{k}{i}(k-1)\\
L_{3,1} &= \frac{d^2}{d-1} - \frac{d^2}{d-1}\cdot\frac{1}{kQ_k}\sum_{i=2}^{k-2} Q_i\binom{k}{i}(k-i-1)\\
L_{3,2} &= \frac{d^{k-1}c}{d-1}\cdot\frac{k}{Q_k}\\
L_{3,3} &= cd^{k-1} - \frac{d^{k-1}c}{d-1}\cdot\frac{k-1}{Q_k} - \frac{cd^{k-1}}{Q_k}\sum_{i=2}^{k-2} Q_i\binom{k}{i}(k-i-1).
\end{aligned}$$

To find the $\det(L)$, multiply out the terms, combine common terms, and simplify. This yields

$$\begin{aligned}
\det(L) = \frac{d^2}{d-1}\cdot\frac{d^{k-1}c}{d-1}k&\left(\frac{cd^{k-1}}{Q_kQ_{k-1}}(1+kK)\left(B_0+Q_k+kQ_{k-1}+\frac{1}{d-1}\right)\right.\\
&+\frac{c+ckK-K}{Q_kQ_{k-1}}\left(-d^2Q_kQ_{k-1} - \frac{kd^2Q_k}{d-1} + \frac{d^2}{k(d-1)}B_3 - \frac{d^2}{d-1}B_1 + d^2Q_{k-1}(B_3-B_1)\right.\\
&\left.\left.+\frac{d^2Q_k}{k}B_3 - d^2Q_kB_2 - d^2(k-1)Q_{k-1}(B_2-B_0) + \frac{d^2}{kQ_kQ_{k-1}}(B_3B_0-B_2B_1)\right)\right)
\end{aligned} \tag{4.22}$$

where

$$\begin{aligned}
B_0 &= \sum_{i=2}^{k-2} \binom{k}{i} Q_i \\
B_1 &= \sum_{i=2}^{k-2} i\binom{k}{i} Q_i \\
B_2 &= \sum_{i=2}^{k-2} (k-i)\binom{k}{i} Q_i \\
B_3 &= \sum_{i=2}^{k-2} i(k-i)\binom{k}{i} Q_i.
\end{aligned}$$

Using (4.18) for Q_i and properties of the binomial gives

$$\begin{aligned}
B_0 &= \frac{d}{d-1}\left(d^k-(d-1)^k-k(d-1)^{k-1}-k(d-1)-1\right)-d(k(-1)^{k-1}+(-1)^k-k+1\big) \\
B_1 &= \frac{k}{d}\left(d^{k-1}-(d-1)^{k-1}-(k-1)(d-1)^{k-2}+(k-1)(-1)^{k-2}+(-1)^{k-1}\right) \\
B_2 &= \frac{k}{d(d-1)}\left(d^{k-1}-(d-1)^{k-1}-(k-1)(d-1)-1)+\frac{k}{d}(-1+(k-1)-(-1)^{k-1}\right) \\
B_3 &= \frac{k(k-1)}{d}\left(d^{k-2}-(d-1)^{k-2}+(-1)^{k-2}\right).
\end{aligned}$$

Plugging these values for the B_i's back into (4.22) and solving gives

$$\det(L) = \frac{kKcd^{3k-1}}{(d-1)^3 Q_k Q_{k-1}},$$

and thus

$$\begin{aligned}
D &= \left(k^{-(k-3)}\right)\frac{kKcd^{3k-1}}{(d-1)^3 Q_k Q_{k-1}}\left(\prod_{i=2}^{k-2}\frac{d^{k-1}c}{d-1}\cdot\frac{k}{\binom{k}{i}Q_i}\right) \\
&= k^2c^{k-2}d^{k^2-k+2}(d-1)^{-k}K\prod_{i=2}^{k}\frac{1}{\binom{k}{i}Q_i}. \qquad (4.23)
\end{aligned}$$

As in Section 4.4.2, to prove the negative Hessian matrix is positive definite one must show the determinate of each of the leading principal minors of the matrix is positive. In this case, it suffices to show the determinate of the second leading principal minor is positive. A very simple generalization of Lemma 4.10, replacing every occurrence of "3" in the statement of the lemma and the proof with "k", proves that $K > 0$, and this fact plus the calculation of the determinate D demonstrate that the rest of the leading principal minors obviously have positive determinates.

$$\begin{aligned}\begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix} &= \left(\frac{d^2}{d-1}\right)(1+kK)\left(-kc\frac{d^2}{d-1}+kK\frac{d^2}{d-1}+\frac{d^{k-1}c}{d-1}\left(\frac{k^2}{Q_k}+\frac{k}{Q_{k-1}}\right)\right) \\ &\quad -\left(kK\frac{d^2}{d-1}\right)^2 \\ &= kK\left(\frac{d^2}{d-1}\right)^2+\left(\frac{d^2}{d-1}\right)^2(1+kK)\left(d^{k-3}\left(\frac{k}{Q_k}+\frac{1}{Q_{k-1}}\right)-1\right)kc.\end{aligned}$$

Thus, it is sufficient to show that $d^{k-3}\left(\frac{k}{Q_k}+\frac{1}{Q_{k-1}}\right) > 1$, and this holds since by the definition of Q_i, $d^{k-3} > Q_{k-1}$.

Step 4: Calculate g at this maximum.

$$\begin{aligned} g(\tau) &= (2\pi n)^{-\frac{k}{2}}c^{-\frac{k-1}{2}}\left(\frac{1}{d}\left(1-\frac{1}{d}\right)\left(\prod_{i=0}^{k}\binom{k}{i}\frac{d-1}{d^{k-1}}Q_i\right)\right)^{-\frac{1}{2}} \\ &\quad \times \Phi(krm,\alpha n)\Phi(k(1-r)m,(1-\alpha)n)\Phi^{-1}(km,n) \\ g(\tau) &= (2\pi n)^{-\frac{k}{2}}c^{-\frac{k-1}{2}}\left((d-1)^k d^{-(k^2-k+2)}\prod_{i=2}^{k}\binom{k}{i}Q_i\right)^{-\frac{1}{2}}\Phi(kc,1). \qquad (4.24)\end{aligned}$$

Step 5: Assume from the General Maximum Hypothesis that (4.20) is the unique maximum of f, and use the Laplace Method to approximate $\frac{\mathbf{E}(N^2)}{\mathbf{E}(N)^2}$.

The general Laplace Method for a multiple integral can be stated as follows.

Lemma 4.18 ([dB70]) *Let*

$$F(n) = \int_{a_1}^{b_1}\cdots\int_{a_k}^{b_k} g(x_1,\ldots,x_k)e^{nh(x_1,\ldots,x_k)}dx_1\ldots dx_k$$

where

(a) h is continuous in $a_i \le x_i \le b_i$,

(b) $h(c_1,\ldots,c_k)=0$ *for some point* $(c_1,\ldots,c_k)$ *with* $a_i<c_i<b_i$ *and* $h(x_1,\ldots,x_k)<0$ *for all other points in the range,*

(c) $h(x_1,\ldots,x_k)=-\frac{1}{2}\sum_{i=1}^{k}\sum_{j=1}^{k}a_{ij}x_ix_j+\mathrm{o}(x_1^2+\cdots+x_k^2)$
with $(x_1^2+\cdots+x_k^2\to 0)$*, and*

(d) the quadratic form $\sum\sum a_{ij}x_ix_j$ *is positive definite.*

Then,

$$F(n)\sim(2\pi)^{\frac{k}{2}}D^{-\frac{1}{2}}n^{-\frac{k}{2}}g(c_1,\ldots,c_k)$$

where D *is the determinant of the matrix* (a_{ij})*.*

This yields,

$$\begin{aligned}
\frac{\mathbf{E}(N^2)}{\mathbf{E}(N)^2}&=\sum_{\alpha}\sum_{r}\sum_{t_i}g(\alpha,r,t_0,\ldots,t_k)e^{nf(\alpha,r,t_0,\ldots,t_k)}\\
&\sim kc^{k-1}n^k\int\cdots\int g(\alpha,r,t_0,\ldots,t_k)e^{nf(\alpha,r,t_0,\ldots,t_k)}dt_{k-2}\ldots dt_2\,dt_0\,dr\,d\alpha\\
&\sim kc^{k-1}n^k(2\pi)^{\frac{k}{2}}D^{-\frac{1}{2}}n^{-\frac{k}{2}}g(\tau) && \text{by the Laplace Method}\\
&\sim kc^{k-1}n^{\frac{k}{2}}(2\pi)^{\frac{k}{2}}D^{-\frac{1}{2}}g(\tau)\\
&\sim kc^{k-1}n^{\frac{k}{2}}(2\pi)^{\frac{k}{2}}k^{-1}c^{-\frac{k-2}{2}}d^{-\frac{k^2-k+2}{2}}(d-1)^{\frac{k}{2}}\prod_{i=2}^{k}\left(\binom{k}{i}Q_i\right)^{\frac{1}{2}}Kg(\tau) && \text{by (4.23)}\\
&\sim c^{\frac{k}{2}}n^{\frac{k}{2}}(2\pi)^{\frac{k}{2}}d^{-\frac{k^2-k+2}{2}}(d-1)^{\frac{k}{2}}\prod_{i=2}^{k}\left(\binom{k}{i}Q_i\right)^{\frac{1}{2}}Kg(\tau)\\
&\sim c^{\frac{k}{2}}n^{\frac{k}{2}}(2\pi)^{\frac{k}{2}}d^{-\frac{k^2-k+2}{2}}(d-1)^{\frac{k}{2}}\prod_{i=2}^{k}\left(\binom{k}{i}Q_i\right)^{\frac{1}{2}}K\\
&\qquad\times(2\pi n)^{-\frac{k}{2}}c^{-\frac{k-1}{2}}(d-1)^{-\frac{k}{2}}d^{\frac{k^2-k+2}{2}}\prod_{i=2}^{k}\left(\binom{k}{i}Q_i\right)^{-\frac{1}{2}}\Phi(kc,1) && \text{by (4.24)}\\
&\sim c^{\frac{1}{2}}K\Phi(kc,1)\\
&\sim 1 && \text{by an analogue of (4.7).}
\end{aligned}$$

□

4.6 Analyzing the Maximum Hypothesis

In this section, we will explore the Maximum Hypothesis and give evidence to support it. The Maximum and General Maximum Hypotheses each state that a certain function has one unique maximum in its domain. We will start this analysis with the General Maximum Hypothesis, and we will find the location of all stationary points of the function.

The General Maximum Hypothesis concerns the following function where we consider any integer constants $k \geq 3$ and $d \geq 4$ and any real constant c such that $\kappa_k < c \leq 1$ where κ_k is the threshold for the appearance of a 2-core in a k-uniform hypergraph. Let

$$\begin{aligned} f(\alpha, r, t_0, \ldots, t_k) = {} & \ln d - c\ln d + (1-\alpha)\ln(d-1) - \alpha\ln\alpha - (1-\alpha)\ln(1-\alpha) \\ & - t_0 c\ln t_0 - \sum_{i=2}^{k} t_i c\ln t_i + \sum_{i=2}^{k-1} t_i c\ln\binom{k}{i} + \sum_{i=2}^{k} t_i c\ln P_i \\ & + \alpha\ln(e^z - 1 - z) - krc\ln z + krc\ln r - \ln(e^x - 1 - x) + kc\ln x \\ & + (1-\alpha)\ln(e^y - 1 - y) - k(1-r)c\ln y + k(1-r)c\ln(1-r) \end{aligned}$$

where $P_i = \frac{(d-1)^{i-1} + (-1)^i}{d(d-1)^{i-1}}$, $\sum_{i=0}^{k} t_i = 1$, $\sum_{i=0}^{k-1}(k-i)t_i = kr$, $t_1 = 0$, and $x, y, z > 0$ are defined as

$$\frac{e^x - 1 - x}{e^x - 1} - \frac{x}{kc} = \frac{e^y - 1 - y}{e^y - 1} - \frac{y(1-\alpha)}{kc(1-r)} = \frac{e^z - 1 - z}{e^z - 1} - \frac{z\alpha}{kcr} = 0. \tag{4.25}$$

The General Maximum Hypothesis states that this function has a unique maximum in the region bounded by $0 \leq \alpha, r, t_i \leq 1$, for $i = 0 \ldots k$, and this maximum is at

$$\alpha = r = \frac{1}{d}, t_i = \binom{k}{i}\frac{d-1}{d^k}\left((d-1)^{i-1} + (-1)^i\right)$$

for $0 \leq i \leq k$.

We can prove that if k is odd, for each y, there are at exactly two stationary points, one with $z = y$, and one when $z > y$. We prove that the stationary point with $z = y$ is at our conjectured unique maximum. We then focus on the Maximum Hypothesis for the case $k = 3$ and $d = 4$. This case corresponds to the first NP-complete variation of

UE-CSP, and we use Mathematica to give numerical evidence that the stationary point with $z > y$ is not a local maximum and that there are no maxima on the boundaries of the domain.

To obtain a formal proof of the Maximum Hypothesis, we need to rigorously prove these last two points. The more challenging will be proving that the stationary point with $z > y$ is not a local maximum. The technique used to prove that should be sufficient to complete the proof that there are no maxima on the boundary. For the General Maximum Hypothesis, we need to expand that proof to $k > 3$ as well as deal with the possibility that there may be stationary points with $z < y$ when k is even.

While the evidence below strongly suggests that the Maximum Hypothesis holds for $k = 3$ and $d = 4$, it does not constitute a proof. The Mathematica results are not rigorous, and the computations required are rather complicated. We can not rule out the possibility that small error terms are accumulating to give a large error in Mathematica's result.

4.6.1 An Equation for All Stationary Points of f

To find all stationary points for f, we use the partial derivatives of f, calculated in Section 4.5, and from these we know that any stationary point must satisfy all of the following equations.

$$\frac{1-\alpha}{\alpha} = (d-1)\frac{e^y - 1 - y}{e^z - 1 - z} \tag{4.26}$$

$$\frac{1-r}{r} = \frac{y}{z}\frac{t_k}{t_{k-1}}\frac{Q_{k-1}}{Q_k}(d-1)k \tag{4.27}$$

$$t_i = \frac{(t_{k-1})^{k-i}}{(t_k)^{k-i-1}}\frac{Q_i\,(Q_k)^{k-i-1}}{(Q_{k-1})^{k-i}}\frac{\binom{k}{k-i}}{k^{k-i}} \tag{4.28}$$

for $0 \le i \le k$ where

$$Q_i = \frac{(d-1)^{i-1} + (-1)^i}{d}. \tag{4.29}$$

Note that these equations are all independent of the constant c.

Let

$$\Gamma = \frac{t_{k-1}Q_k}{kt_kQ_{k-1}}. \tag{4.30}$$

Then we rewrite (4.28) as

$$t_i = \frac{t_k}{Q_k}\Gamma^{k-i}\binom{k}{k-i}Q_i. \tag{4.31}$$

From the fact that $\sum_{i=0}^{k} t_i = 1$ and $\sum_{i=0}^{k-1}(k-i)t_i = kr$, we have

$$\sum_{i=0}^{k} t_i = \frac{1}{kr}\sum_{i=0}^{k-1}(k-i)t_i.$$

and substituting in (4.31) gives

$$\sum_{i=0}^{k}\frac{t_k}{Q_k}\Gamma^{k-i}\binom{k}{k-i}Q_i = \frac{1}{kr}\sum_{i=0}^{k-1}(k-i)\frac{t_k}{Q_k}\Gamma^{k-i}\binom{k}{k-i}Q_i.$$

Applying (4.29) gives,

$$\begin{aligned}
\sum_{i=0}^{k}\binom{k}{i}\Gamma^{k-i}Q_i &= \frac{1}{r}\sum_{i=0}^{k-1}\frac{k-i}{k}\binom{k}{i}\Gamma^{k-i}Q_i\\
\sum_{i=0}^{k}\binom{k}{i}\Gamma^{k-i}\left(\frac{(d-1)^{i-1}+(-1)^i}{d}\right) &= \frac{1}{r}\sum_{i=0}^{k-1}\frac{k-i}{k}\binom{k}{i}\Gamma^{k-i}\left(\frac{(d-1)^{i-1}+(-1)^i}{d}\right)\\
\sum_{i=0}^{k}\binom{k}{i}\Gamma^{k-i}\left(\frac{(d-1)^{i}}{d-1}+(-1)^i\right) &= \frac{\Gamma}{r}\sum_{i=0}^{k-1}\binom{k-1}{i}\Gamma^{k-1-i}\left(\frac{(d-1)^{i}}{d-1}+(-1)^i\right)\\
\frac{(d-1+\Gamma)^k}{d-1}+(\Gamma-1)^k &= \frac{\Gamma}{r}\left[\frac{(d-1+\Gamma)^{k-1}}{d-1}+(\Gamma-1)^{k-1}\right].
\end{aligned}$$

Since $d \geq 2$ and $\Gamma > 0$, we have $(d-1+\Gamma)^k + (d-1)(\Gamma-1)^k > 0$, and

$$\begin{aligned}
r &= \Gamma\left(\frac{(d-1+\Gamma)^{k-1}+(d-1)(\Gamma-1)^{k-1}}{(d-1+\Gamma)^{k}+(d-1)(\Gamma-1)^{k}}\right),\\
1-r &= \frac{(d-1)(d-1+\Gamma)^{k-1}-(d-1)(\Gamma-1)^{k-1}}{(d-1+\Gamma)^{k}+(d-1)(\Gamma-1)^{k}},
\end{aligned}$$

and

$$\frac{1-r}{r} = \frac{d-1}{\Gamma}\left(\frac{(d-1+\Gamma)^{k-1}-(\Gamma-1)^{k-1}}{(d-1+\Gamma)^{k-1}+(d-1)(\Gamma-1)^{k-1}}\right). \tag{4.32}$$

Combining (4.27) and (4.30) yields

$$\frac{1-r}{r} = \frac{y(d-1)}{z\Gamma}, \tag{4.33}$$

and combining (4.32) and (4.33) gives

$$\frac{y}{z} = \frac{(d-1+\Gamma)^{k-1} - (\Gamma-1)^{k-1}}{(d-1+\Gamma)^{k-1} + (d-1)(\Gamma-1)^{k-1}}, \tag{4.34}$$

or

$$\frac{z}{y} = 1 + \frac{d(\Gamma-1)^{k-1}}{(d+\Gamma-1)^{k-1} - (\Gamma-1)^{k-1}}. \tag{4.35}$$

Since all stationary points must satisfy (4.35), we will look at cases. First, we can rule out the case when k is odd and $z < y$. If $z < y$ then the left hand side of (4.35) is smaller than 1, but since $(d+\Gamma-1)^{k-1} - (\Gamma-1)^{k-1} > 0$, the right hand side is larger than 1. Thus, there are no solutions to (4.35) with k odd and $z < y$. We will prove below that, for each y, there is exactly one stationary point with $z > y$.

First we prove that the stationary point with $z = y$ is the maximum that is conjectured to be unique. If $z = y$, then the only solution to (4.35) has $\Gamma = 1$. Plugging $z = y$ into (4.26) gives $\alpha = \frac{1}{d}$, and plugging $\Gamma = 1$ into (4.33) gives $r = \frac{1}{d}$. Since $\alpha = r$, from (4.25) we have $x = y = z = \frac{kc(e^z-1)}{kc+e^z-1}$. Plugging (4.31) into $\sum_{i=0}^{k} t_i = 1$ and setting $\Gamma = 1$ yields $\frac{t_k}{Q_k} = \frac{d-1}{d^{k-1}}$, and plugging this back into (4.31) gives $t_i = \binom{k}{i}\frac{d-1}{d^k}\left((d-1)^{i-1} + (-1)^i\right)$. Plugging these values into f gives the desired maximum of $2\ln d - 2c\ln d$.

Now we prove that for each y, there is exactly one stationary point with $z > y$. We will do this by fixing y and determining how each side of the equation (4.35) changes as z increases. We will calculate the first and second derivatives of the right hand side of (4.35), and we will show that both are 0 when $z = y$ and both are positive when $z > y$. That implies the right hand side of (4.35) will cross $\frac{z}{y}$ exactly once when $z > y$.

First we need an equation for Γ in terms of z. From (4.25),

$$\frac{e^y-1-y}{e^z-1-z} = \frac{y}{z}\frac{(1-\alpha)}{\alpha}\frac{r}{(1-r)}\frac{(e^y-1)}{(e^z-1)}.$$

From (4.26),

$$\frac{e^y-1-y}{e^z-1-z} = \frac{1-\alpha}{\alpha(d-1)},$$

and so

$$\frac{y}{z}\frac{(1-\alpha)}{\alpha}\frac{r}{(1-r)}\frac{(e^y-1)}{(e^z-1)} = \frac{1-\alpha}{\alpha(d-1)},$$

and simplifying gives

$$\frac{y}{z}\frac{r}{(1-r)}(d-1)=\frac{e^z-1}{e^y-1}.$$

From (4.33), we have

$$\Gamma=\frac{yr(d-1)}{z(1-r)},$$

and so

$$\Gamma=\frac{e^z-1}{e^y-1}. \tag{4.36}$$

Therefore, when $\Gamma<1$, $z<y$, and when $\Gamma>1$, $z>y$.

The first derivative of the right hand side of (4.35) with respect to z, with y fixed is

$$\frac{d^2(k-1)(\Gamma-1)^{k-2}(d+\Gamma-1)^{k-2}}{((d+\Gamma-1)^{k-1}-(\Gamma-1)^{k-1})^2}\cdot\frac{\partial\Gamma}{\partial z}, \tag{4.37}$$

and the second derivative is

$$\begin{aligned}&\frac{d^2(k-1)(d+\Gamma-1)^{k-3}(\Gamma-1)^{k-3}}{((d+\Gamma-1)^{k-1}-(\Gamma-1)^{k-1})^2}\\ &\qquad\times\Bigg((d+\Gamma-1)(\Gamma-1)\frac{\partial^2\Gamma}{\partial z^2}+(k-2)(d+2(\Gamma-1))\left(\frac{\partial\Gamma}{\partial z}\right)^2\\ &\qquad-2(k-1)(d+\Gamma-1)(\Gamma-1)\frac{(d+\Gamma-1)^{k-2}-(\Gamma-1)^{k-2}}{(d+\Gamma-1)^{k-1}-(\Gamma-1)^{k-1}}\left(\frac{\partial\Gamma}{\partial z}\right)^2\Bigg)\end{aligned} \tag{4.38}$$

where $\frac{\partial\Gamma}{\partial z}=\frac{e^z}{e^y-1}$. It is straightforward to see that both (4.37) and (4.38) are 0 when $\Gamma=1$, and (4.37) is negative when $\Gamma<1$ and positive when $\Gamma>1$.

To show the second derivative is positive when $\Gamma>1$, it is sufficient to show that

$$(k-2)(d+2(\Gamma-1))-2(k-1)(d+\Gamma-1)(\Gamma-1)\frac{(d+\Gamma-1)^{k-2}-(\Gamma-1)^{k-2}}{(d+\Gamma-1)^{k-1}-(\Gamma-1)^{k-1}}>0$$

when $k\geq 3$, $d\geq 2$, and $\Gamma>1$.

$$\begin{aligned}&(k-2)(d+2(\Gamma-1))-2(k-1)(d+\Gamma-1)(\Gamma-1)\frac{(d+\Gamma-1)^{k-2}-(\Gamma-1)^{k-2}}{(d+\Gamma-1)^{k-1}-(\Gamma-1)^{k-1}}\\ =\;&((k-2)(d+2(\Gamma-1))-2(k-1)(\Gamma-1))\,(d+(\Gamma-1))^{k-1}\\ &+(2(k-1)(d+(\Gamma-1))-(k-2)(d+2(\Gamma-1)))\,(\Gamma-1)^{k-1}\\ =\;&kd(d+(\Gamma-1))^{k-1}+kd(\Gamma-1)^{k-1}-2\left((d+(\Gamma-1))^k-(\Gamma-1)^k\right)\end{aligned}$$

$$\begin{aligned}
&= kd(\Gamma-1)^{k-1} + kd\left(\sum_{i=0}^{k-1}\binom{k-1}{i}d^i(\Gamma-1)^{k-1-i}\right) \\
&\quad -2\left(\sum_{j=0}^{k}\binom{k}{j}d^j(\Gamma-1)^{k-j} - (\Gamma-1)^k\right) \\
&= kd(\Gamma-1)^{k-1} + \sum_{i=0}^{k-1} k\binom{k-1}{i}d^{i+1}(\Gamma-1)^{k-1-i} - 2\sum_{j=1}^{k}\binom{k}{j}d^j(\Gamma-1)^{k-j} \\
&= kd(\Gamma-1)^{k-1} + \sum_{j=1}^{k} k\binom{k-1}{j-1}d^j(\Gamma-1)^{k-j} - 2\sum_{j=1}^{k}\binom{k}{j}d^j(\Gamma-1)^{k-j} \\
&= kd(\Gamma-1)^{k-1} + \sum_{j=1}^{k} d^j(\Gamma-1)^{k-j}\binom{k}{j}(j-2) \\
&= \sum_{j=3}^{k} d^j(\Gamma-1)^{k-j}\binom{k}{j}(j-2) \\
&> 0.
\end{aligned}$$

To get an equation for all possible stationary points in terms of only z and y, we plug (4.36) into (4.35), assume $z \neq y$, and rearrange getting the following equation that has a nice pattern.

$$\frac{z+(d-1)y}{z-y} = \left(\frac{(e^z-1)+(d-1)(e^y-1)}{(e^z-1)-(e^y-1)}\right)^{k-1}. \tag{4.39}$$

4.6.2 Numeric Evidence for $k=3$ and $d=4$ That There Is Only One Maximum in the Interior of the Domain

All maxima, minima, and saddle points will satisfy equation (4.39). One way to show that f has a unique maximum in the interior is to examine the negative Hessian matrix at each of the stationary points. From the second partial derivatives of f, computed in Section 4.5, we have

$$f_{\alpha\alpha} = -\frac{1}{\alpha} - \frac{1}{1-\alpha} + \frac{e^z-1}{e^z-1-z}\frac{\partial z}{\partial\alpha} - \frac{e^y-1}{e^y-1-y}\frac{\partial y}{\partial\alpha}$$

where

$$\begin{aligned}
\frac{\partial z}{\partial \alpha} &= \frac{-z(e^z-1)^2}{\alpha(e^z-1)^2 + krc(e^z(e^z-1-z)-(e^z-1)^2)} \\
\frac{\partial y}{\partial \alpha} &= \frac{y(e^y-1)^2}{(1-\alpha)(e^y-1)^2 + k(1-r)c(e^y(e^y-1-y)-(e^y-1)^2)}.
\end{aligned}$$

As a result, the top left value in the negative Hessian matrix will be

$$\frac{1}{\alpha} + \frac{1}{1-\alpha} - \frac{e^z-1}{e^z-1-z}\frac{\partial z}{\partial \alpha} + \frac{e^y-1}{e^y-1-y}\frac{\partial y}{\partial \alpha}.$$

Using the same analysis as in Section 4.4.3, we can prove that $\frac{\partial z}{\partial \alpha}$ is always negative when $z > 0$ and that $\frac{\partial y}{\partial \alpha}$ is always positive when $y > 0$, and as a result the top left value in the negative Hessian matrix will be positive.

Examining the case where $k = 3$ and $d = 4$, we know that (4.39) gives the location of every stationary point, and because k is odd, for each y, there are exactly two solutions of (4.39). One corresponds to the known local maximum, and the other is a stationary point with $z > y$. If we show that the determinant of the 3×3 negative Hessian matrix is negative at this point, then that point can be not be a local maximum nor a local minimum.

First we limit the domain of y to contain only values that admit a $c < 1$. From (4.25), we have

$$c = \frac{1}{3} \cdot \frac{\alpha}{r} \cdot \frac{(e^z-1)z}{e^z-1-z},$$

from (4.26), we have

$$\alpha = \frac{e^z-1-z}{3(e^y-1-y)+e^z-1-z},$$

and from (4.33) and (4.36), we have

$$r = \frac{(e^z-1)z}{3(e^y-1)y+(e^z-1)z}.$$

Combining these three equations gives

$$c = \frac{1}{3} \cdot \frac{3(e^y-1)y+(e^z-1)z}{3(e^y-1-y)+e^z-1-z}. \qquad (4.40)$$

From (4.25), it is straightforward to see that $y < x < z$, and we can initially bound y with the upper bound for z. From (4.25), the maximum value x can have with $c \leq 1$ is the largest solution to

$$3 = \frac{x(e^x - 1)}{e^x - 1 - x},$$

or $x = 2.1491\ldots$.

Using Mathematica, we start y at 0.001 and increment y by 0.001 until it exceeds 2.500. For each value of y, we use Newton's method to determine the value of z that is the largest solution to (4.39) with $k = 3$ and $d = 4$, and we plug both this y and z value into (4.40) and solve for c. Figure 4.1 plots c as a function of y. Notice that it appears we can further restrict the domain for y because when y is greater than approximately 1.4, c looks to be greater than 1.

Next we compute the value of the determinant of the negative Hessian matrix for f at each y and the corresponding computed z. We again start y at 0.001 and increment y by 0.001 until it exceeds 2.000. Figure 4.2 plots this determinant. Note that the value appears to be negative in the entire plot, supporting our claim that these (y, z) pairs are not local maxima.

4.6.3 Evidence for $k = 3$ and $d = 4$ That There Is No Maximum on the Boundary of the Domain

Our goal is to examine the boundary of the domain for f when $k = 3$ and $d = 4$. The domain is $0 \leq \alpha, r, t \leq 1$, and we will restrict the domain further to the region where $x, y, z > 0$ and f has a real value.

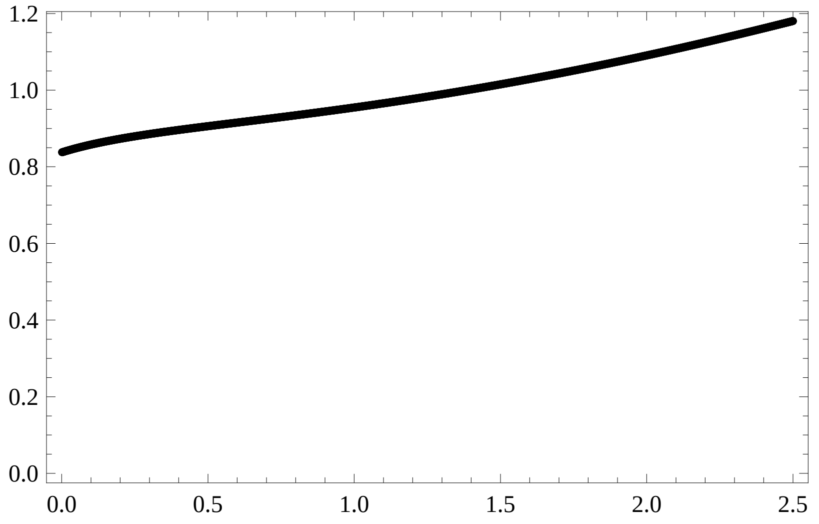

Figure 4.1: A plot of c as a function of y when $k = 3$ and $d = 4$. The largest solution to equation (4.39) gives the corresponding value of z, and then equation (4.40) gives the value of c.

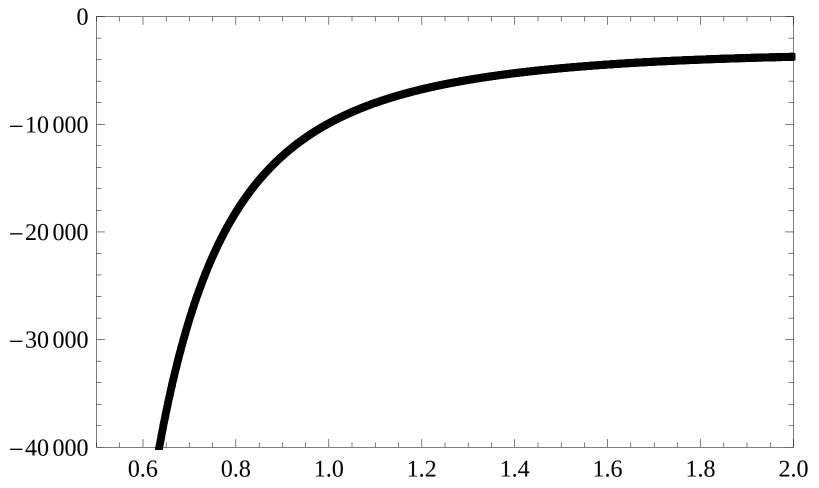

Figure 4.2: A plot of the determinant of the negative Hessian matrix for f when $k = 3$ and $d = 4$ as a function of y where z is the largest solution of (4.39).

From, Section 4.4, the function f restricted to $d=4$ is

$$\begin{aligned} f_{d=4}(\alpha,r,t) &= \ln 4 - c\ln 4 + (1-\alpha)\ln(3) - \alpha\ln\alpha - (1-\alpha)\ln(1-\alpha) \\ &\quad - c(1-3r+2t)\ln(1-3r+2t) - c(3r-3t)\ln(3r-3t) - ct\ln t \\ &\quad - c(1-3r+2t)\ln 9 + c(1-3r+2t)\ln 2 \\ &\quad + r3c\ln r + (1-r)3c\ln(1-r) + \alpha\ln(e^z-1-z) - r3c\ln z \\ &\quad + (1-\alpha)\ln(e^y-1-y) - (1-r)3c\ln y - \ln(e^x-1-x) + 3c\ln x. \end{aligned}$$

We will rewrite $f_{d=4}$ slightly using the equalities

$$\frac{e^x-1-x}{e^x-1} - \frac{x}{3c} = \frac{e^y-1-y}{e^y-1} - \frac{y(1-\alpha)}{3c(1-r)} = \frac{e^z-1-z}{e^z-1} - \frac{z\alpha}{3cr} = 0. \tag{4.41}$$

from (4.2).

$$\begin{aligned} f_{d=4}(\alpha,r,t) &= \ln 4 - c\ln 4 + (1-\alpha)\ln(3) - \alpha\ln\alpha - (1-\alpha)\ln(1-\alpha) \\ &\quad - c(1-3r+2t)\ln(1-3r+2t) - c(3r-3t)\ln(3r-3t) - ct\ln t \\ &\quad - c(1-3r+2t)\ln 9 + c(1-3r+2t)\ln 2 \\ &\quad + r3c\ln r + (1-r)3c\ln(1-r) \\ &\quad + \alpha\left(\ln(e^z-1-z) - \frac{(e^z-1)z}{e^z-1-z}\ln z\right) \\ &\quad + (1-\alpha)\left(\ln(e^y-1-y) - \frac{(e^y-1)y}{e^y-1-y}\ln y\right) \\ &\quad - \ln(e^x-1-x) + \frac{(e^x-1)x}{e^x-1-x}\ln x. \end{aligned}$$

Let

$$\Phi(x) = -\ln(e^x-1-x) + \frac{(e^x-1)x}{e^x-1-x}\ln x.$$

Observation 4.19 *The function $\Phi(x)$ has the following form. When $x \to 0$, the function approaches $\ln 2$, the function has one minimum at $x=1$ with value $-\ln(e-2)$, and the function grows unbounded as $x \to \infty$.*

As a result,

$$\Phi(x) = -\ln(e^x - 1 - x) + \frac{(e^x - 1)x}{e^x - 1 - x} \ln x$$

is always positive, and

$$-\alpha\Phi(z) = \alpha \left(\ln(e^z - 1 - z) - \frac{(e^z - 1)z}{e^z - 1 - z} \ln z \right),$$
$$-(1-\alpha)\Phi(y) = (1-\alpha) \left(\ln(e^y - 1 - y) - \frac{(e^y - 1)y}{e^y - 1 - y} \ln y \right)$$

are always negative. Since the maximum of

$$\begin{aligned}(1-\alpha)\ln(3) - \alpha\ln\alpha - (1-\alpha)\ln(1-\alpha) - c(1-3r+2t)\ln(1-3r+2t) \\ - c(3r-3t)\ln(3r-3t) - ct\ln t - c(1-3r+2t)\ln 9 + c(1-3r+2t)\ln 2 \\ + r3c\ln r + (1-r)3c\ln(1-r)\end{aligned}$$

occurs at the local maximum, the only way another point can be a local maximum is if $\Phi(x)$ dominates the linear combination $-(1-\alpha)\Phi(y) - \alpha\Phi(z)$ by a sufficient amount.

Observation 4.20 *Because the maximum value x can have in $f_{d=4}$ is the largest solution to*

$$3 = \frac{x(e^x - 1)}{e^x - 1 - x},$$

or $x = 2.1491\ldots$, the largest value of $\Phi(x)$ with $x > 1$ is $.60355\ldots$.

The first boundaries we will consider are $\alpha = 0$ and $\alpha = 1$.

The boundary with $\alpha = 0$.

The terms

$$(1-\alpha)\ln 3 - \alpha\ln\alpha - (1-\alpha)\ln(1-\alpha)$$

will all go to 0 as α goes to 0, and this will decrease the value of $f_{d=4}$ by $\ln 4$. From Observations 4.19 and 4.20, the maximum value of $\Phi(x)$ is at most $\ln 2$. As a result, there can not be a maximum when α tends to 0.

The boundary with $\alpha = 1$.

The terms

$$(1-\alpha)\ln 3 - \alpha \ln \alpha - (1-\alpha)\ln(1-\alpha)$$

will decrease from $\ln 4$ to $\ln 3$ as α goes to 1. From Observation 4.19, the z and y terms will reduce the value of $f_{d=4}$ by at least $-\ln(2-e)$. To have a local maximum, the x terms must add back in at least

$$\ln 4 - \ln 3 - \ln(2-e) \approx 0.61857.$$

From Observation 4.20, $\Phi(x)$ can only achieve this amount with $x < 1$. However, as α tends to 1, y tends to ∞, and so $z < x$. This means both $\Phi(z)$ and $\Phi(y)$ are larger than $\Phi(x)$, and so $\Phi(x) - \alpha\Phi(z) - (1-\alpha)\Phi(y) < 0$, and there can not be a maximum when α tends to 1.

The boundaries with $r = \frac{2\alpha}{3c}$ and $1 - r = \frac{2(1-\alpha)}{3c}$.

The next boundaries to test are for r tending to 0 and 1. However, from (4.41), we have

$$\frac{e^z - 1 - z}{z(e^z - 1)} = \frac{\alpha}{3cr},$$

and as

$$\lim_{z\to 0} \frac{e^z - 1 - z}{z(e^z - 1)} = \frac{1}{2},$$

for $z > 0$, we must have $r > \frac{2\alpha}{3c}$, and so we will examine the boundary when r tends to $\frac{2\alpha}{3c}$ from above. The boundary when $1 - r$ tends to $\frac{2(1-\alpha)}{3c}$ is handled analogously.

We will take advantage of the fact that the threshold for the a.s. appearance of a 2-core in a uniformly random hypergraph with edges of size 3 is at clause density $0.818469\ldots$, and the clause density in the 2-core will be greater than for the original hypergraph. This is the reason for the technicality in the Maximum Hypothesis in Section 4.3 that $0.8 < c \leq 1$.

As $3rc$ approaches 2α, and as $0.8 < c \leq 1$, we know that $r < \alpha$. That implies $\frac{1-\alpha}{3c(1-r)} < \frac{1}{3}$, and y, from (4.41), will be much larger than the maximum value for x. In order to dominate the values $\Phi(y)$ and $\Phi(z)$, x would have to tend toward 0, but if $x < 1$ then $c < 0.8$, and so we do not have a maximum.

The case $t = r$.

For the boundary that depends on t, we note that, as specified in Section 4.4, we require $\frac{3r-1}{2} \leq t \leq r$. Because the edges of this boundary are covered in the above sections, we will search for stationary points and determine that these points are not global maxima.

From the derivatives of $f_{d=4}$, (4.3)-(4.5), we have the following equations that will hold for any stationary point.

$$\frac{1-\alpha}{\alpha} = 3\frac{e^y - 1 - y}{e^z - 1 - z} \tag{4.42}$$

$$\left(\frac{1-r}{r}\right)^2 = \frac{9}{2}\left(\frac{y}{z}\right)^3. \tag{4.43}$$

From (4.25),

$$\frac{z}{y} \cdot \frac{e^z - 1}{e^y - 1} \cdot \frac{e^y - 1 - y}{e^z - 1 - z} = \frac{1-\alpha}{\alpha} \cdot \frac{r}{1-r}. \tag{4.44}$$

Combining (4.44) with (4.42) and (4.43) gives

$$\frac{e^z - 1}{e^y - 1} = \sqrt{\frac{2z}{y}}. \tag{4.45}$$

Using (4.45), we find the x, y values for the stationary points, use (4.42) and (4.43) to solve for α and r, and use (4.25) to solve for c and x. We plug these values into $f_{d=4}$ and plot the results in Figure 4.3. As the plot shows, these values appear to be less than $2(1-c)\ln 4$, the maximum for $f_{d=4}$, and can not be maxima.

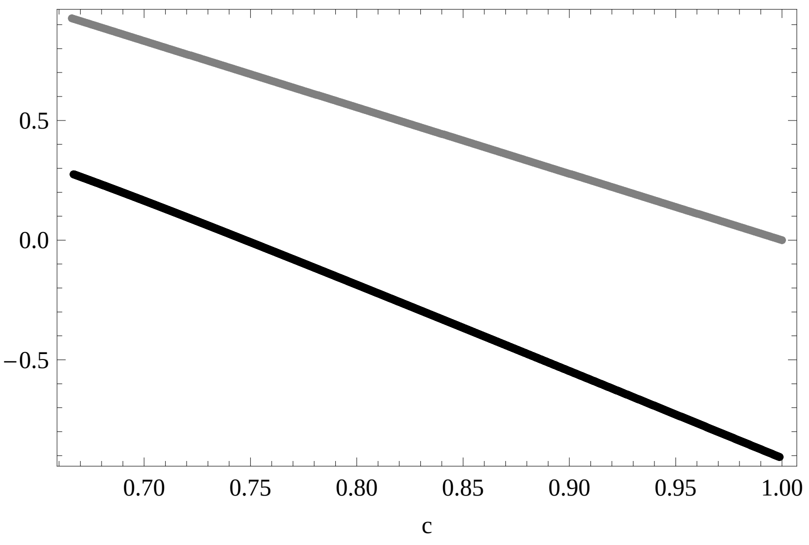

Figure 4.3: The gray line is the maximum of $f_{d=4}$ as a function of c. The black line plots the value of $f_{d=4}$ at the stationary points when $t = r$.

The case $t = \frac{3r-1}{2}$.

From (4.3)-(4.5), we have the following equations that will hold for any stationary point.

$$\frac{1-\alpha}{\alpha} = 3\frac{e^y - 1 - y}{e^z - 1 - z} \tag{4.46}$$
$$\frac{1-r}{r} = \frac{y}{z}\sqrt{\frac{3(1-r)}{3r-1}}.$$

If we solve for r, we get

$$r = \frac{2z^2 \pm z\sqrt{z^2 - 3y^2}}{3(y^2 + z^2)}, \tag{4.47}$$

and combining (4.46) and (4.47) with (4.25) gives two equations for the stationary points.

$$\frac{z}{y} \cdot \frac{e^z - 1}{e^y - 1} = 3\frac{2z^2 + z\sqrt{z^2 - 3y^2}}{3y^2 + z^2 - z\sqrt{z^2 - 3y^2}} \tag{4.48}$$
$$\frac{z}{y} \cdot \frac{e^z - 1}{e^y - 1} = 3\frac{2z^2 - z\sqrt{z^2 - 3y^2}}{3y^2 + z^2 + z\sqrt{z^2 - 3y^2}}. \tag{4.49}$$

Mathematica was unable to find real solutions to (4.49). For (4.48), we find the x, y values for the stationary points, use (4.46) and (4.47) to solve for α and r, and use (4.25)

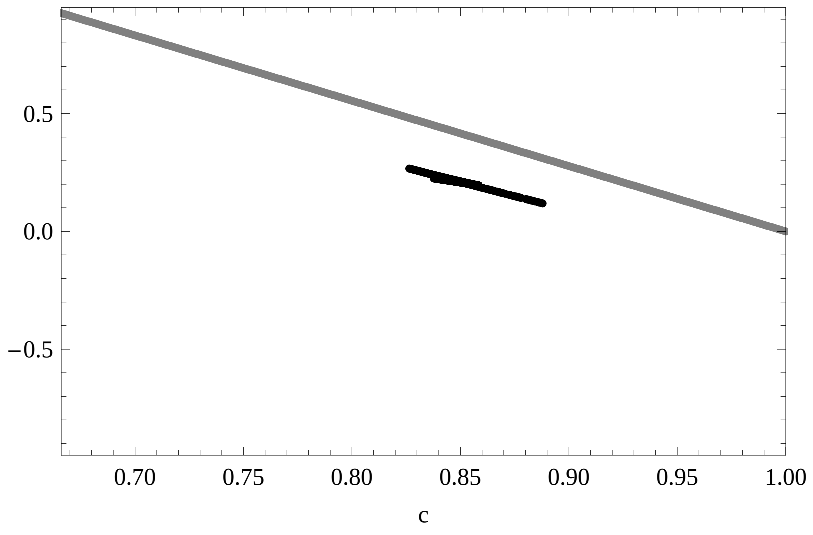

Figure 4.4: The gray line is the maximum of $f_{d=4}$ as a function of c. The black line plots the value of $f_{d=4}$ at the stationary points when $2t = 3r - 1$.

to solve for c and x. We plug these values into f and plot the results in Figure 4.4. The values of $f_{d=4}$ at these stationary points all appear to fall below the line $2(1-c)\ln 4$, the maximum for $f_{d=4}$.

This analysis supports the Maximum Hypothesis, but these results do not constitute a proof. An important continuation of the work in this thesis is to rigorously prove the statements of the Maximum and General Maximum Hypotheses.

Chapter 5

DPLL Behavior on UE-CSP

5.1 Introduction and Main Results

The Davis-Putnam-Logemann-Loveland (DPLL) algorithm [DP60, DLL62] forms the basis of most current complete SAT solvers. The algorithm is a simple backtracking framework. At each step, an unassigned variable v is assigned a value. Any clause that is satisfied by the assignment is removed, v is removed from any clauses in which it occurs, and the constraint on those clauses is appropriately modified. DPLL then recurses on this reduced formula. If a conflict occurs, DPLL backtracks and tries a different value for v.

Because DPLL is a basic framework, there are many possible variations of the algorithm. Current SAT solvers use complex heuristics for choosing the next variable to assign a value, choosing the value to assign, and trimming the search space to prevent DPLL from trying assignments that are known to fail. While the DPLL variations used in practice are too complicated to analyze, we can study simplified variations. Two simple variations discussed in this chapter are DPLL with the unit clause heuristic (DPLL+UC) and DPLL with the generalized unit clause heuristic (DPLL+GUC). In DPLL+UC, if a unit clause (a clause of size 1) exists, the variable from that clause is assigned the appropriate value in order to satisfy the clause. Otherwise, DPLL+UC chooses the next variable that will be assigned a value uniformly randomly from all unassigned variables. In DPLL+GUC, a clause is selected uniformly randomly from the shortest clauses in the formula, and a variable is chosen uniformly randomly from that clause and, in SAT, assigned the appropriate value in order to satisfy that clause. Note that in UE-CSP, we can only choose an appropriate value if the clause is a unit clause, and for larger clauses, the variable is assigned a random value.

As mentioned in Section 2.3.1, bounds on the behavior of both DPLL+UC and DPLL+GUC on 3-SAT are known. Given a random 3-SAT instance with n variables and cn clauses, the result of Chao and Franco [CF86] proves that DPLL+UC will w.u.p.p. take linear time to find a satisfying assignment if $c < \frac{8}{3}$, and Achlioptas, Beame, and Molloy [ABM04b] proves that DPLL+UC will w.u.p.p. require exponential time either to find a satisfying assignment or to verify that the formula is unsatisfiable if $c \geq 3.81$. For DPLL+GUC, from Frieze and Suen [FS96] we know that linear behavior occurs w.u.p.p. when $c \leq 3.003$, and from [ABM04b] we know that exponential behavior occurs w.u.p.p. when $c \geq 3.98$. Note that the upper bounds are below the conjectured satisfiability threshold for 3-SAT. This implies that these DPLL variations are running in exponential time, w.u.p.p., on formulae that are conjectured to be a.s. satisfiable. Applying the same techniques of the above papers gives a similar gap for UE-CSP, and one of the main results of this chapter is that we can take advantage of the uniquely extendible property to close this gap for DPLL+UC on UE-CSP. We can also close the gap for a slight variation of DPLL+GUC.

Theorem 5.1 *Given a random instance of* 3*-UE-CSP with n variables and cn clauses and fixed $d \geq 2$ as input, DPLL+UC will take, w.u.p.p., linear time if $c < \frac{2}{3}$ and exponential time if $c > \frac{2}{3}$.*

A variation of this theorem with the given lower bound but a weaker upper bound is presented in [Con04]. The tight upper bound is implied by Theorem 5.4 which first appears in [CM04]. Note that the bound is below the satisfiability threshold, assuming the Maximum Hypothesis. Again, assuming the Maximum Hypothesis, there are formula that are a.s. satisfiable but DPLL+UC will require exponential time, w.u.p.p., to find a satisfying assignment. A recent paper, Altarelli, Monasson, and Zamponi [AMZ07] also traces the behavior of UC and GUC on k-UE-CSP. That study is non-rigorous, but reaches similar conclusions.

The proof will consist of four steps. From Lemma 5.9 presented in Section 5.2.1, if $c < \frac{2}{3}$, w.u.p.p. DPLL+UC will find a satisfying assignment without backtracking. From Lemma 5.16, if $c > \frac{2}{3}$, w.u.p.p. the unit clause heuristic will guide the algorithm to a subformula on γn variables, $\left(\frac{1}{2} - \epsilon\right) \gamma n$ 2-clauses and $(1+\delta)\gamma n$ 3-clauses where γ, ϵ, and δ are positive constants, ϵ depends on δ, and the subformula is uniformly random on those parameters. By Theorem 5.4 such a formula is a.s. unsatisfiable, and from Lemma 5.17 presented in Section 5.4, DPLL will require, w.u.p.p., $2^{\Omega(\gamma n)}$ steps to backtrack out of this configuration.

We perform a similar analysis for DPLL+GUC, but we do not quite show that DPLL+GUC has a threshold separating exponential and linear time behavior. Instead, we prove in Theorem 5.13 of Section 5.2.2 that a simple variation of DPLL+GUC, when run on a random 3-UE-CSP instance with n variables and cn clauses, will w.u.p.p. find a satisfying assignment in linear time if $c < \Delta$ and will w.u.p.p. run in exponential time if $c > \Delta$, where $\Delta = .75807\ldots$ is the solution of

$$3\Delta - \frac{1}{2}\ln\Delta - \frac{1}{2} - \frac{1}{2}\ln 2 - \frac{1}{2}\ln 3 = 1.$$

This simple variation of DPLL+GUC adds a random coin flip to decide whether the algorithm should look for unit clauses to satisfy or whether the algorithm should deal with 2-clauses. In addition, we add δn random 2-clauses to the instance before running the algorithm, $\delta > 0$ arbitrarily small. These modifications aid the analysis because we can now apply the Lazy Server Lemma of [Ach00] to bound the growth of unit clauses during the execution of the algorithm.

The intuition is that these modifications do not significantly change the behavior of GUC, but we do not have a rigorous proof of this. The additional 2-clauses and delaying the satisfying of unit clauses would only seem to hurt the algorithm. So, we would expect that if this modified DPLL+GUC can solve a problem in linear time w.u.p.p., then so should the standard DPLL+GUC.

On the other hand, if the modifications to DPLL+GUC seem to hurt the algorithm, then the standard DPLL+GUC might be able to find satisfying assignments w.u.p.p. in linear time at a higher clause density than the modified algorithm. However, our intuition suggests this will not be the case. Though we add δn 2-clauses, we can make δ arbitrarily small, and we can make the probability the algorithm satisfies unit clauses arbitrarily close to the expected rate at which unit clauses are generated. This suggests that as n gets large, the behavior of our modified DPLL+GUC will tend toward the behavior of the standard DPLL+GUC.

While these DPLL variations are much simpler than the DPLL algorithms used in practice, studying these variations provides insight into the behavior of the more complicated variants. In particular, most DPLL variations include either a unit clause or generalized unit clause heuristic. Two other common additions to DPLL algorithms are clause learning and restarts. In clause learning, the standard DPLL algorithm with unit clause propagation (all unit clauses are satisfied before testing for a conflict) executes until a conflict occurs. A new clause that identifies the cause of the conflict is added to the set of clauses, and DPLL backtracks. If we allow restarting, DPLL may throw away all variable assignments and start again, but if clause learning is implemented, all learned clauses are retained. The running time of the DPLL algorithm without clause learning is lower bounded by the size of the smallest tree-resolution proof. It is known that tree-resolution can require proofs that are exponentially larger than those for regular resolution [BEGJ00, BSIW03], and regular resolution can require proofs that are exponentially larger than those for general resolution [AJPU02]. Beame, Kautz, and Sabharwal [BKS03] proves that adding clause learning to DPLL can lead to exponentially smaller proofs of unsatisfiability than regular resolution, and adding unlimited restarts and the assumption that an unknown clause is learned on every encountered conflict gives general resolution. This result implies that adding clause learning and restarts will not improve on the exponential bounds of Theorems 5.1 and 5.13. DPLL will not learn a clause until a conflict occurs. From the proof of Theorem 5.1, and similarly Theorem 5.13, when the first conflict occurs, w.u.p.p. the unassigned variables induce a uniformly random subformula that both is a.s. unsatisfiable and w.u.p.p. requires an exponentially long resolution proof of refutation. This implies the following remark.

Remark 5.2 *Adding clause learning to DPLL can give an exponentially faster algorithm, but the exponential lower bounds of Theorems 5.1 and 5.13 still hold even if we supplement clause learning with unlimited restarts.*

We note in Section 2.3.1 that one way to close the gap in the proven behaviors of DPLL+UC and DPLL+GUC on SAT is to prove the $(2+p)$-SAT Conjecture, stated below as Conjecture 5.3. As mentioned in Section 2.2, it is known that for any $\epsilon > 0$, a uniformly random SAT formula with $(1-\epsilon)n$ 2-clauses and λn 3-clauses is a.s. satisfiable if $\lambda \leq \frac{2}{3}$. On the other hand, there exists an $\epsilon > 0$ such that a uniformly random SAT formula with $(1-\epsilon)n$ 2-clauses and λn 3-clauses is a.s. unsatisfiable if $\lambda \geq 2.28$. The $(2+p)$-SAT Conjecture, stated formally below, is that the bound of $\frac{2}{3}$ is tight. In Section 5.3, we prove the third main result of this chapter, the UE-CSP version of the $(2+p)$-SAT Conjecture. The lower bound of the theorem is presented in [Con04], and the upper bound is in [CM04].

Conjecture 5.3 ($(2+p)$-SAT Conjecture [AKKK01, BMW00]) *For every constant $\delta > 0$ there exists a constant $\epsilon > 0$ such that a uniformly random instance of SAT with $(1-\epsilon)n$ 2-clauses and $\left(\frac{2}{3}+\delta\right)n$ 3-clauses is a.s. unsatisfiable.*

Theorem 5.4 *For any constant $\epsilon > 0$ a uniformly random instance of UE-CSP with $\left(\frac{1}{2} - \epsilon\right) n$ 2-clauses, at most $\frac{1}{6}n$ 3-clauses, and no other clauses is w.u.p.p. satisfiable.*

For any constant $\delta > 0$ there exists a constant $\epsilon > 0$ such that a uniformly random instance of UE-CSP with $\left(\frac{1}{2} - \epsilon\right) n$ 2-clauses and $\left(\frac{1}{6} + \delta\right) n$ 3-clauses is a.s. unsatisfiable.

Section 5.1.1 will introduce the $(2+p)$-UE-CSP model, similar to the $(2+p)$-SAT model. In Section 5.2, we apply the same techniques used in the study of SAT to study the behavior of the greedy, non-backtracking algorithms unit clause and generalized unit clause on UE-CSP, and we get analogous results to those known for SAT. In Section 5.3 we prove that, by taking advantage of the uniquely extendible nature of UE-CSP, we can answer the $(2+p)$ Conjecture affirmatively for UE-CSP, and this result tightens the bounds we get in Section 5.2. Finally, Section 5.4 will prove that, analogous to $(2+p)$-SAT, a uniformly random instance of $(2+p)$-UE-CSP w.u.p.p. has exponential resolution complexity.

It is important to note that the proofs of this chapter hold for any constant domain size $d \geq 2$. As a result, all theorems proven here for UE-CSP also apply to XOR-SAT. In particular, Theorems 5.1 and 5.13 give the first proven bounds for the behavior of DPLL on XOR-SAT, and Theorem 5.4 answers the $(2+p)$-SAT Conjecture for XOR-SAT.

This chapter also reveals important differences between SAT and UE-CSP. For one, results that can be proven to hold a.s. in SAT can only be proven to hold w.u.p.p. in UE-CSP. This difference is not a weakness of the techniques, but it is indicative of the nature of UE-CSP and is a consequence of the fact that 2-SAT has a sharp satisfiability threshold while 2-UE-CSP does not. Also, it should be noted that in SAT it is possible to satisfy a clause without assigning values to all of its variables. As a result, DPLL on SAT will remove clauses of various sizes. However, for UE-CSP only singleton clauses will be removed.

5.1.1 The $(2 + p)$-UE-CSP Model

In order to model the subformulae produced during an execution of DPLL, we introduce the random $((2+p), d)$-UE-CSP model, similar to the $(2+p)$-SAT model. In this model, a UE-CSP instance on n variables and m constraints has pm clauses of size 3 and $(1-p)m$ clauses of size 2. To generate a random instance, we first choose $U^{(3,d)}_{n,pm}$, a uniformly random member of the set $\Omega^{(3,d)}_{n,pm}$ of all $(3, d)$-UE-CSP instances with pm 3-clauses on the variables $\{v_1, \ldots, v_n\}$. Then we choose $U^{(2,d)}_{n,(1-p)m}$, a uniformly random member of the set $\Omega^{(2,d)}_{n,(1-p)m}$ of all $(2, d)$-UE-CSP instances with $(1-p)m$ 2-clauses over the same set of variables. The uniformly random member of $((2 + p), d)$-UE-CSP is formed by merging the two clause sets together. As with (k, d)-UE-CSP, when the domain size can be any arbitrary value greater than 1, we drop the d from the notation.

Experiments by Cocco, Monasson, Montanari, and Semerjian [CMMS03] suggest that for each value of p, $0 \le p \le 1$, there is an exact satisfiability threshold for $(2 + p)$-SAT, and there is also an exact threshold for DPLL algorithms on 3-SAT. If the input to the algorithm is a random 3-SAT instance with clause density below the threshold for that DPLL algorithm, the algorithm will find the satisfying assignment in linear time, but if the input has clause density above the threshold, the algorithm will produce a subformula with a clause density that falls on the unsatisfied side of the satisfiability threshold for $(2 + p)$-SAT, and DPLL will take a long time to backtrack out of the subformula. Achlioptas, Beame, and Molloy [ABM04b] proves this algorithm behavior for DPLL+UC and DPLL+GUC using an upper bound on the conjectured $(2 + p)$-SAT threshold, and this chapter will prove analogous behavior for UE-CSP.

Unlike with SAT, $(2 + p)$-UE-CSP will not have a sharp satisfiability threshold for $p < 1$. The reason, similar to that for the 2-UE-CSP threshold of Lemma 4.1, is that with a linear number of 2-clauses, w.u.p.p. the 2-clauses will contain a small number of cycles, and w.u.p.p. each cycle can cause the formula to be unsatisfiable. As a result, the threshold for $(2 + p)$-UE-CSP will distinguish the random formulae that are a.s.

unsatisfiable from the random formulae that are w.u.p.p. satisfiable.

Let c_p be the non-sharp, satisfiability threshold for $(2+p)$-UE-CSP, if it exists. That is, let $c_p(n)$ be the least density at which a uniformly random $(2+p)$-UE-CSP formula on n variables is a.s. unsatisfiable, and assume $c_p = \lim_{n\to\infty} c_p(n)$. To get a trivial upper bound for c_p, we note that both the subformula induced by the 3-clauses and the subformula induced by the 2-clauses must be satisfiable. Thus,

$$c_p \le \min\left\{\frac{c_2^*}{1-p}, \frac{c_3^*}{p}\right\} = \min\left\{\frac{1}{1-p}, \frac{.917935\ldots}{p}\right\}.$$

To get a nontrivial upper bound for c_p, we count the expected number of solutions to a random instance of $((2+p),d)$-UE-CSP. For both $(2,d)$-UE-CSP and $(3,d)$-UE-CSP, a random assignment satisfies each clause with probability $\frac{1}{d}$. Thus, if $\mathcal{S}_n$ is the set of solutions for any $((2+p),d)$-UE-CSP formula on n variables, the expected number of solutions is

$$\mathbf{E}(|\mathcal{S}_n|) = d^n\left(\frac{1}{d}\right)^{cn},$$

and when $c > 1$, $\mathbf{E}(|\mathcal{S}_n|)$ tends to 0 as n tends to infinity. Thus, from Markov's inequality, we get $c_p \le 1$ and the following lemma.

Lemma 5.5 *For any $\epsilon > 0$, a uniformly random UE-CSP instance with $\left(\frac{1}{2}-\epsilon\right)n$ 2-clauses and βn 3-clauses with $\beta > \left(\frac{1}{2}+\epsilon\right)$ is a.s. unsatisfiable.*

This bound on unsatisfiability is not tight, and it will be strengthened in Section 5.3.

5.2 Behavior of Various Non-Backtracking Algorithms

To study the behavior of DPLL+UC and DPLL+GUC on 3-UE-CSP, we will consider both unit clause and generalized unit clause as non-backtracking algorithms. An upper bound on when a stand-alone algorithm that runs in linear time w.u.p.p. finds a satisfying assignment implies that DPLL using that algorithm as a heuristic for choosing the next variable to assign will w.u.p.p. find a satisfying assignment in linear time.

The algorithms considered in this section have the following structure. At each step of the algorithm, a variable is selected and assigned a value. Either the variable is selected uniformly at random from all unassigned variables or a particular clause is identified and a variable is chosen uniformly at random from the variables in that clause. Such algorithms are called “card games” by Achlioptas [Ach01] because we can represent a random formula as a pack of cards with one card for each occurrence of a variable in the formula. The card face records the variable, and the cards are placed face down in m columns where column i gets the same number of cards as there are variables in clause i. The algorithm may select a variable either by naming it or by pointing to a card, and the variable selected is the variable on the card face. All cards that contain the selected variable are turned face up. A key observation for SAT, see e.g. [Ach01], that can be trivially extended to UE-CSP, or any CSP for that matter, is that the distribution of the face down cards is still uniformly random over the unselected variables. This is formalized in the following fact.

Fact 5.6 *Until a card game algorithm backtracks, the subproblem produced at each step by the algorithm is uniformly random. Specifically, the 2-clauses form a uniformly random instance of* 2*-UE-CSP, and the 3-clauses form a uniformly random instance of* 3*-UE-CSP.*

5.2.1 Unit Clause

In this section we model the behavior of unit clause, without backtracking, by a system of differential equations. Let $C_i(t)$ be the number of i-clauses at step t of the algorithm. Note that at each step of the algorithm, an unassigned variable is given a value. Since no backtracking occurs, the number of steps is the same as the number of assigned variables.

From Fact 5.6, if we select a random variable to assign, we expect that variable to occur in $\frac{3C_3(t)}{n-t}$ 3-clauses and $\frac{2C_2(t)}{n-t}$ 2-clauses. As a result, we expect the number of 3-clauses to decrease by $\frac{3C_3(t)}{n-t}$, and since each 3-clause becomes a 2-clause, we expect the number of 2-clauses to change by $\frac{3C_3(t)}{n-t} - \frac{2C_2(t)}{n-t}$. To make the analysis more straightforward, we will not stop the algorithm when a contradiction is reached. Instead, we will have the algorithm continue until all variables are assigned a value, and then we will check for contradictions in the form of null clauses.

To model this behavior as a system of differential equations, let x be the number of variables assigned a value and $c_i(x)$ the number of i-clauses with c_i and x normalized to the range $[0, 1]$, and we have

$$\begin{aligned}\frac{dc_3}{dx} &= -\frac{3c_3(x)}{(1-x)}\\ \frac{dc_2}{dx} &= \frac{3c_3(x)}{(1-x)} - \frac{2c_2(x)}{(1-x)},\end{aligned}$$

and solving the differential equations gives

$$c_3(x) = c_3(0)(1-x)^3 \tag{5.1}$$

$$c_2(x) = (c_2(0) + 3c_3(0)x)(1-x)^2. \tag{5.2}$$

These equations are almost identical to the analogous equations for the behavior of UC on SAT. The only difference is that, for 3-SAT, half of the $\frac{3c_3(x)}{1-x}$ 3-clauses will be satisfied by the assignment to the variable and half will become 2-clauses while for UE-CSP, all will become 2-clauses. Therefore, the justifications found in Achlioptas, Kirousis, Kranakis, and Krizanc [AKKK01] and Achlioptas [Ach01] that use a theorem of Wormald [Wor95] to prove the analogous differential equations describe the a.s. behavior of UC on SAT also imply that the above equations describe the a.s. behavior of UC on UE-CSP. Specifically, for any $\epsilon > 0$ and for $0 \leq t \leq (1-\epsilon)n$, a.s.

$$C_i(t) = c_i(t/n) \cdot n + o(n), \tag{5.3}$$

and therefore,

$$C_3(t) = c_3(0)(1-t/n)^3 \cdot n + \mathrm{o}(n) \tag{5.4}$$

$$C_2(t) = (c_2(0) + 3c_3(0)(t/n))(1-t/n)^2 \cdot n + \mathrm{o}(n). \tag{5.5}$$

The justifications of [AKKK01, Ach01] depend on three properties holding. One is a technicality, that the functions $\frac{3c_3(x)}{(1-x)}$ and $\frac{3c_3(x)}{(1-x)} - \frac{2c_2(x)}{(1-x)}$ satisfy a Lipschitz condition. The second property is that as long as x and $c_i(x)$, $0 \le i \le 3$, stay within some domain, the change in the number of 3-clauses and 2-clauses at each step has constant expectation and is highly concentrated. Specifically, the probability the change in the number of i-clauses at step t is more than $n^{1/5}$ is $\mathrm{o}(n^{-3})$. It is easy to see this property holds because if we let $X_{i,v}$ be number of i-clauses that contain the variable v, then $X_{i,v}$ is a binomial random variable, and we can use a straightforward application of the Chernoff bound. The third property is that, while x and $c_i(x)$ are within some domain, changing the value of t, $C_3(t)$, $C_2(t)$, and $C_1(t)$ by $\mathrm{o}(n)$ only affects the expected change in the value of C_3 and C_2 at step t by $\mathrm{o}(1)$.

Given (5.4) and (5.5), the important observation is that as long as no clause of length 0 is generated, no contradiction is reached, and a clause of length 0 can only be generated if we have more than one clause of length 1. From Fact 5.6, the subformula at each step is uniformly random, and so the probability that, if we satisfy a unit clause at step t, no clause of length 0 is generated at step t is $\left(1 - \frac{1}{n-t}\frac{d-1}{d}\right)^{C_1(t)-1}$. If we run the algorithm for $(1-\epsilon)n$ steps, the probability that no contradiction is generated during all $(1-\epsilon)n$ steps is

$$\prod_{t=1}^{(1-\epsilon)n} \left(1 - \frac{1}{n-t}\frac{d-1}{d}\right)^{a(t)} \text{ where } a(t) = \begin{cases} C_1(t) - 1 & \text{if } C_1(t) > 0 \\ 1 & \text{otherwise} \end{cases},$$

and this probability is lower bounded by

$$\left(1 - \frac{1}{\epsilon n}\frac{d-1}{d}\right)^{\sum_{t=1}^{(1-\epsilon)n} a(t)}.$$

The expected number of clauses of length 1 generated at step t is $2C_2(t)/(n-t)$. So if this density is bounded by $(1-\delta)$ for some $\delta > 0$, the expected number of unit clauses generated at each step will be less than the rate at which unit clauses are satisfied by the algorithm, and a.s. the unit clauses will not accumulate. As noted in [Ach01] the number of unit clauses behaves like the queue size in a stable server system. Therefore, the total number of unit clauses generated during s steps of the algorithm is a.s. less than Ms where M depends only on δ. As a result, we can lower bound the probability that no clause of length 0 is generated during the first $(1-\epsilon)n$ steps by a constant independent of n:

$$\left(1 - \frac{1}{\epsilon n}\frac{d-1}{d}\right)^{(M-1+\epsilon)n} \geq \left(1 - \frac{1}{\epsilon n}\frac{d-1}{d}\right)^{Mn} \geq e^{-\frac{M}{\epsilon}\frac{d-1}{d}}.$$

Therefore, w.u.p.p. there will be no contradictions. In addition, because the expected number of unit clauses generated at each step is less than 1, we can lower bound the probability that no unit clauses are generated at a specific step. These observations are summarized in the following lemma which is a direct extension of a lemma for SAT in both [AKKK01] and [Ach01], and this lemma of [AKKK01, Ach01] is a compilation of results in Chao and Franco [CF90]. Because a straightforward application of the technique used in [CF90] will prove Lemma 5.7, the proof is omitted.

Lemma 5.7 *Fix $\delta, \epsilon > 0$ and let $t_0 = n - \lfloor \epsilon n \rfloor$. If for all $0 \leq t \leq t_0$ a.s. $C_2(t) < \frac{1}{2}(1-\delta)(n-t)$ then w.u.p.p. $C_1(t_0) + C_0(t_0) = 0$.*

We can use the differential equations (5.4) and (5.5) to a.s. trace the first $t_0 = n - \lfloor \epsilon n \rfloor$ steps of UC. If we add the condition that $C_2(t) < \frac{1}{2}(1-\delta)(n-t)$, we can bound the probability that UC fails because by Lemma 5.7, after step t_0 we are left with a formula with ϵn variables, w.u.p.p. no clauses of length 1, and a.s. $C_3(t_0)$ clauses of length 3 and $C_2(t_0)$ clauses of length 2 where

$$\begin{aligned} C_3(t_0) &= c_3(0)\epsilon^3 n + \mathrm{o}(n) \\ C_2(t_0) &= (c_2(0) + 3c_3(0)(1-\epsilon))\epsilon^2 n + \mathrm{o}(n). \end{aligned}$$

For Lemmas 5.8 and 5.9 below, we need to analyze the algorithm to termination, and so we must deal with the final $n - t_0$ steps. We will prove that if we pick ϵ small enough w.u.p.p. the remaining subproblem will be simple enough that UC will always find a satisfying assignment.

We define a cycle in the remaining clauses to be a sequence of l distinct variables $v_1, \ldots, v_l$ and l distinct clauses $e_1, \ldots, e_l$, with $l \geq 2$ such that each pair $v_i, v_{(i \mod l)+1}$ is contained in clause e_i. Because the subformula remaining after t_0 steps is uniformly random, we can switch to the model where we have ϵn vertices and each of the $\binom{\epsilon n}{3}$ possible 3-clauses is added with probability $p_3 = \frac{C_3(t_0)}{\binom{\epsilon n}{3}} + \mathrm{o}(1)$ and each of the $\binom{\epsilon n}{2}$ possible 2-clauses is added with probability $p_2 = \frac{C_2(t_0)}{\binom{\epsilon n}{2}} + \mathrm{o}(1)$. Therefore, the probability that a pair of variables exists in a clause is $p_2 + 1 - (1-p_3)^{\epsilon n - 2}$. Given a sequence of l variables, since we require each pair in the sequence to be in a different clause, the probability that the sequence forms a cycle is the probability that each sequential pair of variables exists in a clause. This latter probability is $\left(p_2 + 1 - (1-p_3)^{\epsilon n-2} - \mathrm{o}(1)\right)^l$, and hence the expected number of cycles of length l is

$$\binom{\epsilon n}{l} \frac{l!}{2l} \left(p_2 + 1 - (1-p_3)^{\epsilon n - 2} + \mathrm{o}(1)\right)^l .$$

Using the fact that $p_2 + 1 - (1-p_3)^{\epsilon n-2} = p_2 + \epsilon n p_3 + \mathrm{o}(1)$, we can find an upper bound on the expected number of cycles of length l, for any constant l, as follows.

$$\begin{aligned}
&\binom{\epsilon n}{l} \frac{l!}{2l} \left(p_2 + 1 - (1-p_3)^{\epsilon n-2}\right)^l \\
&= \binom{\epsilon n}{l} \frac{l!}{2l} \left(p_2 + \epsilon n p_3 + \mathrm{o}(1)\right)^l \\
&= \frac{(\epsilon n)!}{(\epsilon n - l)!} \frac{1}{2l} \\
&\qquad \times \left(\frac{2\left(c_2(0) + 3c_3(0)(1-\epsilon)\right)\epsilon^2 n}{(\epsilon n)^2} + \frac{6c_3(0)\epsilon^4 n^2}{(\epsilon n)^3} + \mathrm{o}(1) \right)^l
\end{aligned}$$

$$\begin{aligned}
&\sim \frac{(\epsilon n)^{\epsilon n+\frac{1}{2}}e^{-\epsilon n}}{(\epsilon n-l)^{\epsilon n-l+\frac{1}{2}}e^{l-\epsilon n}}\frac{1}{2l}\left(\frac{2c_2(0)+6c_3(0)(1-\epsilon)}{n}+\frac{6c_3(0)\epsilon}{n}\right)^l \\
&= \left(\frac{\epsilon n-l}{\epsilon n}\right)^{-\left(\epsilon n-l+\frac{1}{2}\right)}(\epsilon n)^l e^{-l}\frac{1}{2l}\left(\frac{2c_2(0)+6c_3(0)}{n}\right)^l \\
&\leq e^l(\epsilon n)^l e^{-l}\frac{1}{2l}\left(\frac{2c_2(0)+6c_3(0)}{n}\right)^l \\
&= \frac{\left(\epsilon\left(2c_2(0)+6c_3(0)\right)\right)^l}{2l}
\end{aligned}$$

The expected number of cycles of length l, for any constant l, is a constant that does not depend on n. If we choose $\epsilon < \frac{1}{2c_2(0)+6c_3(0)}$ then the expected number of cycles of length l vanishes as l grows large, and the expected total number tends to a constant. Using a straightforward application of the method of moments technique (see [JŁR00]), we can show that if $\epsilon < \frac{1}{2c_2(0)+6c_3(0)}$ the total number of cycles in the formula is asymptotic to a Poisson random variable with constant mean. Therefore, w.u.p.p. there will be no such cycle in the formula.

Consider a formula where there are no such cycles and no unit clauses. UC will always find a satisfying assignment for this formula because during the execution of UC, there will never be more than one unit clause in each connected component of the subformula induced by the unassigned variables. Consider, as a means of contradiction, the first time that two unit clauses appear in the same connected component, and let u_k be the variable that was assigned a value during this step. Let v_1 and v_2 be the two variables that occur in these unit clauses. Because v_1 and v_2 are in the same connected component, they must be connected by a path P of 2- and 3-clauses. Let u_i be the last variable that UC assigned from a clause of size 2 or 3, and let $U = \{u_i, \ldots, u_k\}$ be the set of all variables assigned by UC between the assignment to u_i and the assignment to u_k, including u_i and u_k. Before u_i was assigned a value, there were no unit clauses, and the assignments to each $u_{i+1}, \ldots, u_k$ was forced by the unit clause rule. As a result, prior to the assignment of u_i, v_1 and v_2 must have been connected by a path of 2- and 3-clauses, containing variables from U, and containing no clauses from P. This implies that we had a cycle in the formula.

For the proofs below, we will use the differential equations (5.4) and (5.5) to a.s. trace the first $t_0 = n - \lfloor \epsilon n \rfloor$ steps of UC, for ϵ sufficiently small. Assuming we do not reach a contradiction during the first t_0 steps, then the analysis above shows that w.u.p.p. there will not be a cycle in the subformula induced by the unassigned variables, Lemma 5.7 implies that w.u.p.p. there will also be no unit clauses, and UC will always find a satisfying assignment for such a subformula.

Using this analysis, we can prove the following two key lemmas. Recall that c_p is the satisfiability threshold for $(2+p)$-UE-CSP, if it exists.

Lemma 5.8 *For* $p \leq \frac{1}{4}$, $c_p = \frac{1}{2(1-p)}$.

Proof. From the observation that if the 2-clauses alone are unsatisfiable then the $(2+p)$-UE-CSP formula is unsatisfiable, we know that $c_p \leq \frac{1}{2(1-p)}$. To prove that $c_p = \frac{1}{2(1-p)}$ for $p \leq \frac{1}{4}$, we will show that UC will succeed w.u.p.p. on random formulae with clause density $c < \frac{1}{2(1-p)}$.

Given p, we run UC on a random formula with cp 3-clauses and $c(1-p)n$ 2-clauses. By the justifications above for (5.3), we can plug $c_3(0) = cp$ and $c_2(0) = c(1-p)$ into (5.5) and a.s. after each step t there will be $C_2(t) = \left(c(1-p) + 3cp\frac{t}{n}\right)\left(1 - \frac{t}{n}\right)^2 n + o(n)$ 2-clauses. From Lemma 5.7, if we add the bound $C_2(t) < \frac{1}{2}(1-\delta)(n-t)$, w.u.p.p. UC will reach step t_0 without producing a contradiction and there will be no unit clauses. From the above observation, w.u.p.p. there are no cycles in the remaining clauses, and in this case, UC will find an assignment for the remaining $n - t_0$ unassigned variables. Substituting the value for $C_2(t)$ into the bound gives

$$\left(c(1-p) + 3cp\frac{t}{n}\right)\left(1 - \frac{t}{n}\right)^2 n < \frac{1}{2}(n-t).$$

Letting $x = \frac{t}{n}$ and simplifying yields

$$2c(3px - p + 1)(1-x) < 1, \tag{5.6}$$

and UC will succeed w.u.p.p. if (5.6) holds for all x in the range $[0, 1)$.

Following the same technique as [AKKK01], we note that if $p \leq \frac{1}{4}$, the l.h.s. of (5.6) is a decreasing function of x, and thus the inequality holds iff it holds for $x = 0$, and plugging in $x = 0$ gives

$$c < \frac{1}{2(1-p)}.$$

$\square$

Lemma 5.9 *Let C be a uniformly random instance of 3-UE-CSP with n variables and cn clauses with $c < \frac{2}{3}$. Then w.u.p.p. DPLL+UC will find a satisfying assignment without backtracking.*

Proof. Plug $c_2(0) = 0$ into (5.5), and a.s. after each step t of UC there will be $C_2(t) = \left(3c_3(0)\frac{t}{n}\right)\left(1 - \frac{t}{n}\right)^2 n + o(n)$ 2-clauses. From Lemma 5.7, if $C_2(t) < \frac{1}{2}(1-\delta)(n-t)$ after each step $t \leq t_0$, w.u.p.p. UC will reach step t_0 without producing a conflict, and there will be no unit clauses in the remaining subformula after step t_0. In addition, from the above observation, w.u.p.p. there will be no cycles in the remaining subformula, and UC will always succeed in assigning values to a subformula with no cycles and no unit clauses. Therefore, UC will succeed w.u.p.p. if $3c_3(0)\frac{n}{t}\left(1 - \frac{n}{t}\right) < \frac{1}{2}$. Since the l.h.s. of the inequality has its maximum when $\frac{n}{t} = \frac{1}{2}$, UC will succeed w.u.p.p. if $c_3(0) < \frac{2}{3}$. $\square$

Finally, we use this technique to prove the following lemma.

Lemma 5.10 *Let C be a uniformly random instance of 3-UE-CSP with n variables and $(\frac{8}{9} + \delta)n$ clauses for any $\delta > 0$. Then there exists $\epsilon > 0$ such that w.u.p.p. DPLL with UC will reach a subproblem C' of C that has $n' > \frac{3}{4}n$ variables, $\left(\frac{1}{2} - \epsilon\right) n'$ 2-clauses and $\beta n'$ 3-clauses, $\beta > \left(\frac{1}{2} + \epsilon\right)$, with all such subproblems equally likely.*

Proof. Choose an arbitrary $\delta > 0$, and trace the a.s. behavior of UC on a 3-UE-CSP formula with Δn 3-clauses where $\Delta = \frac{8}{9} + \delta$. We will show that there exist values $x_c < x_u < \frac{1}{4}$ with $x_u > 0$ and such that for any $x' \in (\max\{x_c, 0\}, x_u)$, w.u.p.p. the subformula remaining after $t' = x'n$ steps of UC will have $\left(\frac{1}{2} - \epsilon\right)(n - t')$ 2-clauses for some $\epsilon > 0$ and $\beta(n - t')$ 3-clauses with $\beta > \frac{1}{2} + \epsilon$.

Plug $c_2(0) = 0$ and $c_3(0) = \Delta$ into (5.1) and (5.2) to get

$$c_3(x) = \Delta(1-x)^3 \tag{5.7}$$

$$c_2(x) = 3\Delta x(1-x)^2. \tag{5.8}$$

Solving the equation $c_3(x)+c_2(x) = 1-x$ for x gives the points where, during the a.s. trace of UC, the sum of the number of 2- and 3-clauses equals the number of variables. Using (5.7) and (5.8) for $c_3(x)$ and $c_2(x)$ in this equation and simplifying gives

$$\Delta(1-x)(2x+1) = 1. \tag{5.9}$$

It is straightforward to verify that (5.9) has two real solutions when $\Delta > \frac{8}{9}$. Let x_c be the smaller solution,

$$x_c = \frac{1}{4} - \frac{\sqrt{9\Delta^2 - 8\Delta}}{4\Delta}. \tag{5.10}$$

Later we will use the fact $x = \frac{1}{4}$ maximizes $\Delta(1-x)(2x+1)$.

Solving the equation $\frac{c_2(x)}{1-x} = \frac{1}{2}$ for x gives the points where, during the a.s. trace of UC, the density of the number of 2-clauses is $\frac{1}{2}n$. Plugging in (5.8) and simplifying yields

$$6\Delta x(1-x) = 1. \tag{5.11}$$

It is straightforward to verify that (5.11) has two real solutions when $\Delta > \frac{2}{3}$. Let x_u be the smaller solution,

$$x_u = \frac{1}{2} - \frac{\sqrt{9\Delta^2 - 6\Delta}}{6\Delta}. \tag{5.12}$$

If $\Delta = \frac{8}{9}$ then $x_c = x_u = \frac{1}{4}$. To prove that for $\Delta = \frac{8}{9} + \delta$ we have $x_c < x_u$, we will take the derivatives of (5.9) and (5.11) with respect to Δ and see how x_c and x_u change as we perturb the value of Δ.

$$\frac{dx_u}{d\Delta} = -\frac{1}{2\Delta\sqrt{9\Delta^2 - 6\Delta}} \tag{5.13}$$

$$\frac{dx_c}{d\Delta} = -\frac{1}{\Delta\sqrt{9\Delta^2 - 8\Delta}}. \tag{5.14}$$

It is clear that if $\Delta > \frac{8}{9}$, both derivatives exist, both derivatives are negative, and $\frac{dx_u}{d\Delta} > \frac{dx_c}{d\Delta}$. Therefore, if $\Delta = \frac{8}{9} + \delta$, we have $x_c < x_u < \frac{1}{4}$, and since $c_2(0) = 0$ we know that $x_u > 0$.

For any value $x \in (0, x_u)$, we have $\frac{c_2(x)}{1-x} < \frac{1}{2}$. In addition, from the observation that $x = \frac{1}{4}$ is the location of the maximum of $\Delta(1-x)(2x+1)$, the interval $\left(x_c, \frac{1}{4}\right)$ is a subinterval of the values x for which $c_3(x) + c_2(x) > 1$. Therefore, for any value x' such that $\max\{x_c, 0\} < x' < x_u$, we have $\frac{c_2(x')}{1-x'} = \frac{1}{2} - \epsilon$ for some $\epsilon > 0$ and $c_3(x') + c_2(x') > 1$, and this implies $c_3(x') = \beta$ for some $\beta > \frac{1}{2} + \epsilon$. Therefore, after $t' = x'n$ steps, the subformula will a.s. have $C_2(t') = c_2(x')n + \mathrm{o}(n)$ 2-clauses with $C_2(t') = \left(\frac{1}{2} - \epsilon\right)(n - t')$ and $C_3(t') = c_3(x')n + \mathrm{o}(n)$ 3-clauses with $C_3(t') = \beta(n - t')$.

Because $C_2(t) < \frac{1}{2}(n-t)$ for all $0 \le t \le t'$, by Lemma 5.7 w.u.p.p. no conflict occurs, and by Note 5.6 the subformula reached after t' steps is uniformly random over all such mixed formulae with the same clause densities. □

Lemmas 5.5 and 5.10 as well as Lemma 5.17 presented in Section 5.4 imply that DPLL on a uniformly random instance of 3-UE-CSP with n variables and cn clauses will take linear time to find a solution w.u.p.p. if $c < \frac{2}{3}$ and exponential time w.u.p.p. if $c > \frac{8}{9}$. This result is similar to the analogous result for 3-SAT of [ABM04b], but it is slightly stronger in the sense that the lower bound for exponential behavior for 3-UE-CSP is below the proven satisfiability threshold. The lower bound for exponential behavior for 3-SAT is below the conjectured satisfiability threshold but above the proven lower bound for that threshold, if it exists. On the other hand, Achlioptas, Beame, and Molloy [ABM04a] gives a lower bound for exponential behavior of DPLL on k-SAT, $k \ge 4$, that is below the proven lower bound for the satisfiability threshold.

In addition, Lemmas 5.5 and 5.8 show that a random $(2+p)$-UE-CSP instance with n variables, $\left(\frac{1}{2} - \epsilon\right) n$ 2-clauses and βn 3-clauses is w.u.p.p. satisfiable if $\beta \le \frac{1}{6}$ and w.u.p.p. unsatisfiable if $\beta > \frac{1}{2} + \epsilon$. Again, this result similar to the analogous result for 3-SAT of [AKKK01], but it is slightly weaker in the sense that the $(2+p)$-SAT behavior is proven a.s. to hold.

The $(2+p)$-SAT Conjecture states that the lower bound is actually tight. By taking advantage of the unique extendibility of UE-CSP, we can prove the analogous conjecture for $(2+p)$-UE-CSP. The proof is in Section 5.3.

5.2.1.1 Extending Unit Clause to Large Clause Sizes

As noted in Achlioptas [Ach01], using differential equations to analyze the behavior of UC on 3-SAT can easily be extended to general k-SAT. The same observation holds for k-UE-CSP. The differential equations for k-UE-CSP are:

$$\begin{aligned} \frac{dc_k}{dx} &= -\frac{kc_k(x)}{1-x} \\ &\vdots \\ \frac{dc_i}{dx} &= \frac{(i+1)c_{i+1}(x)}{1-x} - \frac{ic_i(x)}{1-x}, \end{aligned}$$

and solving yields

$$c_i(x) = (1-x)^i \left[\sum_{j=i}^{k} \binom{j}{i} x^{j-i} c_j(0) \right].$$

If we let $c_i(x) = 0$ for all $2 \leq i < k$ and add the bound that $\frac{2c_2(x)}{1-x} < 1$, UC will succeed w.u.p.p. on k-UE-CSP with n variables and cn clauses if $c < \frac{1}{k}\left(\frac{k-1}{k-2}\right)^{k-2}$.

5.2.2 Generalized Unit Clause

In this section, we prove an upper bound on the clause density at which DPLL with a slightly modified variation of GUC will find a satisfying assignment without backtracking. While the result for GUC on 3-SAT was first proven in [FS96], the proof of Lemma 5.11 follows a simplification, presented in [Ach01], using a technique developed in [Ach00], and using a variation of GUC, called GUC*, based on the techniques of [Ach01].

Lemma 5.11 *Let C be a uniformly random instance of 3-UE-CSP with n variables and cn clauses with $c < \Delta$ where $\Delta = .75807\ldots$ is the solution to*

$$3\Delta - \frac{1}{2}\ln\Delta - \frac{1}{2} - \frac{1}{2}\ln 2 - \frac{1}{2}\ln 3 = 1. \tag{5.15}$$

Then if we add δn random 2-clauses to the problem instance, with $\delta > 0$ arbitrarily small, w.u.p.p. DPLL+GUC will find a satisfying assignment without backtracking.*

As discussed in Section 5.1, it seems intuitive that adding additional random 2-clauses will not make the problem easier to solve, and GUC*, defined in Figure 5.1, appears to be a slightly weaker solver than GUC because it does not deal with unit clauses as soon as they appear. We argue, but do not prove, that these modifications only help the analysis, but do not change the behavior of GUC on random 3-UE-CSP.

Proof. As mentioned in Section 5.2.1, one property of unit clause that permitted [AKKK01, Ach01] to use the theorem of [Wor95] to model UC with differential equations is the property that changing the value of t, $C_3(t)$, $C_2(t)$, and $C_1(t)$ by $o(n)$ only affects the expected change of C_3 and C_2 at time t by $o(1)$. However, for GUC, this property no longer holds because the expected change of C_2 at time t depends on whether $C_1(t) = 0$. The solution used in [Ach01] is to analyze a modified version of the algorithm and apply the Lazy-server Lemma of [Ach00]. In the variation GUC*, the algorithm will flip a coin, and with probability q it will satisfy a unit clause or, if no unit clause exists, it will assign a value to a random variable. As a result, the expected change in C_2 at time t depends on q and not on whether $C_1(t) = 0$.

If $n - t = \Theta(n)$, in one round of GUC* the expected number of 3-clauses that become 2-clauses is $\frac{3C_3(t)}{n-t}$. The change in 2-clauses depends on the coin flip. If $U(t) = 1$ or there are no 2-clauses, the expected number of 2-clauses that become unit clauses is $\frac{2C_2(t)}{n-t}$, and if $U(t) = 0$ and there exist 2-clauses, the expected number is $\frac{2(C_2(t)-1)}{n-t} + 1 = \frac{2C_2(t)}{n-t} + 1 + o(1)$. We can simplify this expected number calculation by making the assumption that there will always be 2-clauses present when the algorithm requires one. In fact, for the expected

```
GUC*
(1)  for $t = 1, \ldots, n$
(2)        flip coin $U(t)$ with probability $q$
(3)        if $U(t) = 1$ or if there are no 2-clauses then
(5)              if there are any unit clauses then
(6)                    choose one at random and satisfy it
(7)              otherwise
(8)                    choose a random variable and assign a value to it
(9)        otherwise
(10)             choose a 2-clause at random, choose a random variable
                       from the clause, and assign a value to it
```

Figure 5.1: A modification of GUC used in Lemma 5.11.

change in 2-clauses to be well behaved, we need a linear number of 2-clauses at each step. As in [Ach01], we will add the constraint that $C_2(t) \geq \delta n$ for an arbitrarily small $\delta > 0$. Since the expected number of 2-clauses increases and then decreases, we can ensure this constraint holds a.s. if we start with $2\delta n$ 2-clauses. Clearly, if the formula with the added 2-clauses is satisfiable, then so is the original formula.

As a result, we can describe the algorithm with the following system of differential equations:

$$\frac{dc_3}{dx} = -\frac{3c_3(x)}{1-x} \tag{5.16}$$

$$\frac{dc_2}{dx} = \frac{3c_3(x)}{1-x} - q\frac{2c_2(x)}{1-x} - (1-q)\left(\frac{2c_2(x)}{1-x} + 1\right) \tag{5.17}$$

$$c_2(0) = 2\delta \tag{5.18}$$

where $x = \frac{t}{n}$ and $c_i(x)$ is the number of i-clauses at step $t = xn$ normalized to $[0, 1]$. By the same justifications as [Ach01] for GUC* on SAT, we have $C_i(t) = c_i(x)n + \mathrm{o}(n)$ a.s. for any $\epsilon > 0$ and $0 \le t \le (1 - \epsilon)n$.

Solving (5.16) with initial condition $c_3(0) = \Delta$ yields

$$c_3(x) = 3\Delta(1 - x)^3, \tag{5.19}$$

and using this to simplify (5.17) yields,

$$\frac{dc_2}{dx} = 3\Delta(1 - x)^2 - \frac{2c_2(x)}{1 - x} - 1 + q. \tag{5.20}$$

Note that the rate unit clauses are generated is $\frac{2c_2(x)}{1-x} + 1 - q$, and the rate unit clauses are satisfied is q.

The Lazy-server Lemma of Achlioptas [Ach00] states that if the rate at which the algorithm satisfies unit clauses is greater than the rate unit clauses are generated, then a.s. the number of unit clauses remains bounded throughout the execution of the algorithm. As a result, we have the following lemma for UE-CSP that is analogous to the equivalent lemma of [Ach01] for SAT.

Lemma 5.12 *Let A be any algorithm expressible in the card game that in every step t attempts to satisfy a unit clause with probability $u = u(t, C_2(t), C_3(t))$. If $\delta, \epsilon > 0$ and t_e are such that $t_e \le (1 - \epsilon)n$ and a.s. $C_2(t) < \frac{1}{2}(1 - \delta)(n - t)u$ for all $0 \le t \le t_e$, then there exists $\pi = \pi(\delta, \epsilon) > 0$ such that $\mathbf{Pr}[C_0(t_e) + C_1(t_e) = 0] > \pi$.*

To avoid contradictions, GUC* must attempt to satisfy a unit clause at a rate that is faster than the rate at which they are generated. So, the number of unit clauses will remain bounded if GUC* attempts to satisfy a unit clause with probability $q = \min\left\{(1 + \theta)\left(\frac{c_2(x)}{1-x} + \frac{1}{2}\right), 1\right\}$ for some $\theta > 0$ and if $\frac{c_2(x)}{1-x} + \frac{1}{2} < 1$. As a result, we can substitute $q = (1 + \theta)\left(\frac{c_2(x)}{1-x} + \frac{1}{2}\right)$ into 5.20 to get

$$\frac{dc_2}{dx} = 3\Delta(1 - x)^2 - (1 - \theta)\left(\frac{c_2(x)}{1 - x} - \frac{1}{2}\right). \tag{5.21}$$

To complete the lemma, we solve this differential equation and determine the maximum Δ such that $\frac{c_2(x)}{1-x} < \frac{1}{2}$.

Because we want θ and δ to be arbitrarily small, and because (5.21) is a simple, continuous function, we will use the same technique as [Ach01] and specialize the differential equation by letting $\theta = \delta = 0$. Thus we need to solve the system

$$\frac{dc_2^*}{dx} = 3\Delta(1-x)^2 - \frac{c_2(x)}{1-x} - \frac{1}{2} \tag{5.22}$$

$$c_2^*(0) = 0, \tag{5.23}$$

and solving yields

$$c_2^*(x) = \frac{3}{2}\Delta x^3 - \frac{9}{2}\Delta x^2 + 3\Delta x - \frac{1}{2}x\ln(1-x) + \frac{1}{2}\ln(1-x). \tag{5.24}$$

We need to make certain that for all x in the range $[0,(1-\epsilon)]$, $\frac{2c_2^*(x)}{1-x} < 1$. The derivative of $\frac{2c_2^*(x)}{1-x}$ is $\frac{6\Delta x^2 - 12\Delta x + 6\Delta - 1}{1-x}$. Thus the maximum occurs when $x = 1 \pm (6\Delta)^{-1/2}$. We are interested in the maximum when $x < 1$.

Plugging this value for x into $\frac{2c_2^*(x)}{1-x} = 1$ yields an equation for Δ:

$$3\Delta - \frac{1}{2}\ln\Delta - \frac{1}{2} - \frac{1}{2}\ln 2 - \frac{1}{2}\ln 3 = 1. \tag{5.25}$$

Thus, $\Delta = .75087\ldots$. □

Similar to Theorem 5.1 that gives a threshold separating the linear time and exponential behavior of DPLL+UC on 3-UE-CSP, we can do the same for DPLL+GUC*. The proof exactly follows the proof for DPLL+UC and is sketched in Section 5.5.

Theorem 5.13 *Given a random instance of* 3*-UE-CSP with* n *variables and* cn *clauses as input, if we add* δn *random 2-clauses, for* $\delta > 0$ *arbitrarily small, DPLL+GUC* will take, w.u.p.p., linear time if* $c < \Delta$ *and exponential time if* $c > \Delta$ *where* $\Delta = .75807\ldots$ *is the solution to*

$$3\Delta - \frac{1}{2}\ln\Delta - \frac{1}{2} - \frac{1}{2}\ln 2 - \frac{1}{2}\ln 3 = 1.$$

5.2.3 Other Algorithms for Selecting the Next Variable in DPLL

This chapter limits its analysis to UC and GUC*. Other algorithms have been used for SAT such as the pure literal rule, setting a variable that will satisfy the most clauses, and selecting and satisfying a literal based on its degree and the degree of its complement. However, as mentioned in Section 2.5.2, these algorithms do not apply to XOR-SAT, and by extension to UE-CSP. Implicit in these algorithms is the fact that in SAT, you can satisfy a clause with an assignment to a subset of its variables. In particular, if you set one literal to *true*, the assignment to the rest of the variables in the clause does not matter. This property does not hold in a uniquely extendible CSP.

5.3 Resolving the $(2+p)$ Conjecture for UE-CSP

We now give a proof of Theorem 5.4 which states the UE-CSP version of the $(2+p)$ SAT Conjecture. Namely, Theorem 5.4 states that for a UE-CSP formula with mixture of clauses of size 2 and 3, the maximum clause density at which the unit clause algorithm will w.u.p.p. find a solution is also the maximum clause density at which the formula is w.u.p.p. solvable.

Similar to Section 4.4.1, we reduce a formula F with a mixture of clauses of size 2 and 3 to its 2-core. By the same arguments as the proof of Lemma 4.6, if the 2-core has n' vertices and cn' constraints for $c > 1$, it is a.s. unsatisfiable, and thus F is a.s. unsatisfiable. Cores of random graphs and uniform hypergraphs are well understood, and we can extend these results to non-uniform hypergraphs.

Theorem 5.14 *Let $c_2, c_3 \geq 0$. Let x be the largest solution to*

$$x = (1-e^{-x})^2 3c_3 + (1-e^{-x})2c_2. \tag{5.26}$$

If $x > 0$, then a uniformly random hypergraph with $c_2 n$ 2-edges, $c_3 n$ 3-edges and no *other edges a.s. has a 2-core with $\alpha(c_2, c_3)n + o(n)$ vertices, $\beta_2(c_2, c_3)n + o(n)$ 2-edges and $\beta_3(c_2, c_3)n + o(n)$ 3-edges where $\alpha(c_2, c_3) = 1 - e^{-x} - xe^{-x}$, $\beta_2(c_2, c_3) = c_2(1 - e^{-x})^2$, and $\beta_3(c_2, c_3) = c_3(1 - e^{-x})^3$.*

We will not give the proof of Theorem 5.14 here. Instead, we will prove a more general version in Chapter 6. Theorem 6.1 gives the a.s. size of the r-core of a non-uniform hypergraph for $r \geq 2$, and Theorem 5.14 is a straightforward restriction to the case when $r = 2$ and we only have edges of size 2 and 3.

We restate Theorem 5.4.

Theorem 5.4 *For any constant $\epsilon > 0$ a uniformly random instance of UE-CSP with $\left(\frac{1}{2} - \epsilon\right)n$ 2-clauses, at most $\frac{1}{6}n$ 3-clauses, and no other clauses is w.u.p.p. satisfiable.*

For any constant $\delta > 0$ there exists a constant $\epsilon > 0$ such that a uniformly random instance of UE-CSP with $\left(\frac{1}{2} - \epsilon\right)n$ 2-clauses and $\left(\frac{1}{6} + \delta\right)n$ 3-clauses is a.s. unsatisfiable.

Proof. From Lemma 5.8, if $p \leq \frac{1}{4}$, a uniformly random instance of $(2+p)$-UE-CSP is satisfiable w.u.p.p. if and only if the 2-clauses alone are satisfiable w.u.p.p.. If we have $\left(\frac{1}{2} - \epsilon\right)n$ 2-clauses, then $p \leq \frac{1}{4}$ corresponds to adding up to $\frac{1}{6}n$ 3-clauses.

Now, consider a random UE-SAT formula F on n variables with $\left(\frac{1}{2} - \epsilon\right)n$ constraints of size 2 and $\left(\frac{1}{6} + \delta\right)n$ constraints of size 3 for some $\delta, \epsilon = \epsilon(\delta) > 0$ where δ is arbitrary and ϵ will be chosen later.

Take the underlying hypergraph H of F, and assume H has a 2-core H' with αn vertices and βn hyperedges. Consider the subformula F' of F that corresponds to H'. By the same argument as Fact 4.5, F' is uniformly random conditional on the number of variables, constraints of size 2 and constraints of size 3, and if we choose an assignment, that assignment satisfies each constraint of F', regardless of the size of that constraint, with probability $\frac{1}{d}$ where d is the domain size. Thus by the same argument as the proof of Lemma 4.6,

$$\begin{aligned}\mathbf{E}(\#\text{ of satisfying assignments}) &= d^{\alpha n}\left(\frac{1}{d}\right)^{\beta n} \\ &= \mathrm{o}(1) \text{ if } \beta > \alpha.\end{aligned}$$

Thus, if $\beta > \alpha$, F' is a.s. unsatisfiable and so F is a.s. unsatisfiable.

Now we prove F a.s. has a 2-core with more edges than vertices by applying Theorem 5.14 with $c_2 = \frac{1}{2} - \epsilon$ and $c_3 = \frac{1}{6} + \delta$. Lemma 5.15 below proves that for all $\delta > 0$ there exists an $\epsilon > 0$ such that the x of Theorem 5.14 is positive and $\beta = \beta_2(c_2, c_3) + \beta_3(c_2, c_3) > \alpha(c_2, c_3) = \alpha$. Thus, we pick an ϵ which satisfies Lemma 5.15 and complete the proof. □

Lemma 5.15 *For any $\delta > 0$, there exists $\epsilon > 0$ such that the largest solution to*

$$x = (1 - e^{-x})^2 3\left(\frac{1}{6} + \delta\right) + (1 - e^{-x})2\left(\frac{1}{2} - \epsilon\right) \tag{5.27}$$

is greater than 0 and

$$1 - e^{-x} - xe^{-x} < \left(\frac{1}{2} - \epsilon\right)(1 - e^{-x})^2 + \left(\frac{1}{6} + \delta\right)(1 - e^{-x})^3. \tag{5.28}$$

Proof. We fix an arbitrary $\delta > 0$, and the proof will have two steps. First, we find a value x_δ such that all $x > x_\delta$, if we plug δ and x into (5.27) and solve for ϵ, these values will also satisfy (5.28). Second, we prove that there exists an $\epsilon > 0$ such that plugging δ and ϵ into (5.27) yields an x such that $x > x_\delta$.

Solving (5.27) for $(1 - 2\epsilon)$ gives

$$(1 - 2\epsilon) = \frac{2x - (1 - e^{-x})^2(1 + 6\delta)}{2(1 - e^{-x})}, \tag{5.29}$$

and solving (5.28) for $(1 - 2\epsilon)$ gives

$$(1 - 2\epsilon) > \frac{6(1 - e^{-x} - xe^{-x}) - (1 - e^{-x})^3(1 + 6\delta)}{3(1 - e^{-x})^2}. \tag{5.30}$$

To find x_δ, set the r.h.s.'s of (5.29) and (5.30) equal to each other and solve for δ:

$$\delta = \frac{x + xe^{-x} + 2e^{-x} - 2}{(1 - e^{-x})^3} - \frac{1}{6}. \tag{5.31}$$

Consider the r.h.s. of (5.31). Its derivative is positive if $x > 0$, it tends to 0 as x tends to 0, and it tends to infinity as x tends to infinity. Therefore, for any $\delta > 0$, there is an $x > 0$ such that (5.31) holds. For our fixed δ, we will denote as x_δ the positive value of x that satisfies (5.31).

Note that, for fixed δ, x_δ is the point at which the r.h.s.'s of (5.29) and (5.30) are equal. Also note that for all $x > 0$, the r.h.s. of (5.30) grows slower than the r.h.s. of (5.29). Therefore for all $x > x_\delta$, the $(1-2\epsilon)$ value from the equality (5.29) always satisfies the inequality (5.30). As a result, for any δ and $x > x_\delta$, both (5.27) and (5.28) hold.

Now prove that for any $\delta > 0$, there exists $\epsilon > 0$ such that $x > x_\delta$. From (5.27) we have

$$\delta = \frac{x - (1 - e^{-x})(1 - 2\epsilon)}{3(1 - e^{-x})^2} - \frac{1}{6}. \tag{5.32}$$

Consider the δ of (5.31) as a function of x, denote this function $\delta_{5.31}(x)$, and consider the δ of (5.32) as a function of x and ϵ, denote this function $\delta_{5.32}(x, \epsilon)$. Note that $x_\delta = \delta_{5.31}^{-1}(\delta)$, and let $x_0 = \lim_{\epsilon \to 0} \delta_{5.32}^{-1}(\delta, \epsilon)$. Examining derivatives and limits, we see that for any $x > 0$, $\lim_{e \to 0} \delta_{5.32}(x, \epsilon) < \delta_{5.31}(x)$, and thus for any $\delta > 0$, $x_0 > x_\delta$. Therefore, for any $\delta > 0$, there exists $\epsilon > 0$ such that $x > x_\delta$. $\square$

Finally, we can use this lemma along with the results of Section 5.2 to prove that on random instances of 3-UE-CSP drawn from above the thresholds of Lemmas 5.9 and 5.11, UC will produce uniformly random subformulae with n' variables, $\left(\frac{1}{2} - \epsilon\right)$ 2-clauses, and a linear number of 3-clauses, and the subformulae are a.s. unsatisfiable.

Lemma 5.16 *Let C be a uniformly random instance of* 3*-UE-CSP with n variables and cn clauses. If $c > \frac{2}{3}$, then executing the UC algorithm on C will w.u.p.p. produce a uniformly random subformula on n' variables with $\left(\frac{1}{2} - \epsilon\right) n'$ 2-clauses and $\left(\frac{1}{6} + \delta\right) n'$ 3-clauses where δ and ϵ are positive constants that satisfy Lemma 5.15.*

Proof. Let $\frac{c_3(x)}{1-x} = \frac{1}{6} + \delta$, and let $\frac{c_2(x)}{1-x} = \frac{1}{2} - \epsilon$. If we plug these values into (5.1) and (5.2), set $c_2(0) = 0$, and solve for $c_3(0)$, we get $c_3(0) = \frac{2}{3} \cdot \frac{(1+3\delta-\epsilon)^2}{1+6\delta}$. As δ and ϵ tend to 0, $c_3(0)$ will tend to $\frac{2}{3}$.

For any $\delta > 0$, we set $\epsilon = \epsilon(\delta) > 0$ so that δ and ϵ satisfy (5.27) and (5.28) of Lemma 5.15, and for any $c > \frac{2}{3}$, we can find an appropriate δ so that $c = \frac{2}{3} \cdot \frac{(1+3\delta-\epsilon)^2}{1+6\delta}$. As a result, if we run UC on a uniformly random instance of 3-UE-CSP with n variables and cn clauses, UC will reach a subformula with $\left(\frac{1}{6} + \delta\right) n$ 3-clauses and $\left(\frac{1}{2} - \epsilon\right) n$ 2-clauses such that δ and ϵ satisfy (5.27) and (5.28). By Fact 5.6 such a formula is uniformly random. ☐

A similar lemma will hold for GUC* if we start with $c > \Delta$, where Δ is given by (5.15), and we add δn 2-clauses for some arbitrarily small $\delta > 0$. Then, if we plug $\frac{c_3(x)}{1-x} = \frac{1}{6} + \delta$ and $\frac{c_2^*(x)}{1-x} = \frac{1}{2} - \epsilon$ into (5.19) and (5.24) and solve for $c_3(0)$, we get

$$1 - 2\epsilon = 3\Delta - \frac{1}{2} - 3\delta - \frac{1}{2}\ln 2 - \frac{1}{2}\ln 3 + \ln\left(\frac{\sqrt{\Delta(1+6\delta)}}{\Delta}\right),$$

and letting δ and ϵ approach 0 gives the desired result.

5.4 Resolution Lower Bound for $(2+p)$-UE-CSP

The final step to prove Theorem 5.1 is the following lemma.

Lemma 5.17 *For any $\Delta, \epsilon > 0$, DPLL will require $2^{\Omega(n)}$ steps w.u.p.p. to backtrack out of a uniformly random UE-CSP instance with $\left(\frac{1}{2} - \epsilon\right) n$ 2-clauses and Δn 3-clauses, if that instance is unsatisfiable.*

The trace of an execution of DPLL on an unsatisfiable formula can be converted into a tree-like resolution proof of the same size, and tree-like resolution proofs are at least as large as general resolution proofs. In fact, Bonet, Esteban, Galesi, and Johansen [BEGJ01] and Ben-Sasson, Impagliazzo, and Wigderson [BSIW03] show that tree-like resolution proofs can be exponentially larger. Therefore, Lemma 5.17 follows from the following lemma.

Lemma 5.18 *For any $\Delta, \epsilon > 0$, the resolution complexity of a uniformly random instance of $(2+p)$-UE-CSP with n variables, $\left(\frac{1}{2}-\epsilon\right)n$ 2-clauses and Δn 3-clauses, is w.u.p.p. $2^{\Theta(n)}$.*

Resolution complexity is the smallest size of a resolution proof of unsatisfiability. From techniques developed in Ben-Sasson and Wigderson [BSW01] and Mitchell [Mit02], the shortest resolution proof of unsatisfiability for a CSP has exponential size, a.s., if there exists constants $\alpha, \zeta > 0$ such that a.s. the following three conditions hold.

1. Every subproblem on at most αn variables is satisfiable.
2. Every subproblem on v variables where $\frac{1}{2}\alpha n \leq v \leq \alpha n$ has at least ζn variables of degree at most 1.
3. If x is a variable of degree at most 1 in a CSP f then, letting f' be the subproblem obtained by removing x and its constraint, any satisfying assignment of f' can be extended to a satisfying assignment of f by assigning some value to x.

Note that the third condition is trivially true for UE-CSP. However the first condition does not hold a.s. for a random UE-CSP formula with $\left(\frac{1}{2}-\epsilon\right)n$ 2-clauses because, as first observed in [ER60], the number of constant length cycles induced by the 2-clauses has a Poisson distribution with a constant mean. As a result, w.u.p.p. the formula will have a small unsatisfiable cycle. On the other hand, w.u.p.p. the formula will have no cycles in the 2-clauses and hence no unsatisfiable cycle. As a result, Lemma 5.18 reduces to the following lemma.

Lemma 5.19 *For any $\Delta, \epsilon > 0$, consider a random UE-CSP problem $\mathcal{C}$ on n variables with Δn 3-clauses and $(\frac{1}{2}-\epsilon)n$ 2-clauses where every such formula is equally likely. A.s. either $\mathcal{C}$ has a cycle in the 2-clauses or the following two conditions hold:*

(a) every subformula on at most αn variables is satisfiable, and

(b) every subformula on v variables where $\frac{1}{2}\alpha n \leq v \leq \alpha n$ has at least ζn variables of degree at most 1.

The proof of this lemma closely follows a similar one from Molloy and Salavatipour [MS07].

Proof. Consider any $(2+p)$-UE-CSP problem $\mathcal{C}$ and its underlying hypergraph H. A *pendant path* of H is a path of 2-edges whose internal vertices each have degree 2 and do not lie in any 3-edge of H. Trivially, a single vertex is a pendant path of length 0.

For any $r \geq 1$, a Y_r *configuration* consists of:

- r pendant paths and
- a collection of t_2 additional 2-edges and t_3 additional 3-edges whose vertices are all endpoints of the r pendant paths for some t_2, t_3 with

$$\frac{3}{2}t_2 + 3t_3 \geq \frac{2}{3}r_0 + \frac{5}{3}r_1 \tag{5.33}$$

where r_0 is the number of pendant paths of length 0 and $r_1 = r - r_0$.

Let $\mathcal{P}$ be a set of r pendant paths of H such that (i) every vertex of H appears on exactly one path and (ii) $\mathcal{P}$ is minimal in the sense that it is impossible to form a collection of $r-1$ paths satisfying (i) by adding a 2-edge from H to $\mathcal{P}$.

If C' is a minimally unsatisfiable subformula of $\mathcal{C}$, then C' must be connected and have no vertices of degree at most 1. In addition, the lemma statement assumes $\mathcal{C}$, and therefore C', has no cycle in the 2-clauses. Lemma 5.20 says that such a C' must have a Y_r configuration for some $r \geq 1$. Lemma 5.21 says that there a.s. cannot be a Y_r configuration on at most αn variables for any $r \geq 1$, and so $\mathcal{C}$ a.s. has no unsatisfiable formula on at most αn variables.

Consider any subformula F on v variables where $\frac{1}{2}\alpha n \leq v \leq \alpha n$. Consider a minimal set of pendant paths of the underlying hypergraph of F such that every variable of F

appears on exactly one path. Let r be the number of paths in this set. The proof of Lemma 5.20 below shows that if F has at most $\frac{r}{3}$ variables of degree at most 1 then F has a Y_r configuration. Since a.s. every such formula does not have a Y_r configuration, a.s. for each such F, there exists an $r \geq 1$ such that F has at least $\frac{r}{3}$ variables of degree at most 1 and a collection of r pendant paths that contain all its variables.

Now we show that r must be $\Theta(n)$. Let G be the underlying hypergraph of F. By Lemma 5.22 below, for every constant $\theta > 0$, G a.s. has at most $2ne^2\theta^{-\theta}$ pendant paths of length θ. Since any path of length more than θ contains a path of length exactly θ, G a.s. has at most $2ne^2\theta^{1-\theta}$ vertices on pendant paths of length at least θ. Pick θ so that $2e^2\theta^{1-\theta} < \frac{\alpha}{4}$. Thus, at least $\frac{\alpha}{4}n$ variables of G lie on paths in $\mathcal{P}$ of length less than θ. Therefore, $r > \frac{\alpha}{4\theta}n$, and so G has at least ζn vertices of degree at most 1 for $\zeta = \frac{\alpha}{12\theta}$. $\square$

The following two lemmas closely follow lemmas from [MS07].

Lemma 5.20 *For each H there exists an $r \geq 1$ such that if H has at most $\frac{r}{3}$ vertices of degree at most 1 and no cycles in the 2-edges then H has a Y_r configuration.*

Proof. Let $\mathcal{P}$ be a minimal set of pendant paths of H such that every vertex of H appears on exactly one path. Let r be the number of paths in $\mathcal{P}$, let r_0 be the number of paths of length 0, and let $r_1 = r - r_0$.

We call the edges of $\mathcal{P}$ *path edges* and the other edges of H *non-path edges.* Note that every non-path edge contains only vertices that are endpoints of the paths in $\mathcal{P}$. Let t_2 be the number of non-path 2-edges, and let t_3 be the number of (non-path) 3-edges. We will prove H has a Y_r configuration by proving $\frac{3}{2}t_2 + 3t_3 \geq \frac{2}{3}r_0 + \frac{5}{3}r_1$.

We define a set X to contain exactly those vertices that are an endpoint of a path of $\mathcal{P}$. Thus, $|X| = 2r_1 + r_0$. We form a graph G with vertex set X, and the edges of G are the 2-edges of H that do not lie on a path of $\mathcal{P}$. Note that $t_2 = |E(G)|$.

Let l_1 be the number of components of G with exactly one vertex, and let l_2 be the number of components with exactly two vertices. The remaining components have size at least 3, and thus these components contain $|X| - l_1 - 2l_2$ vertices and at least $\frac{2}{3}(|X| - l_1 - 2l_2)$ edges. Therefore, $t_2 \geq l_2 + \frac{2}{3}(|X| - l_1 - 2l_2)$, and rearranging gives $\frac{3}{2}t_2 + l_1 + \frac{1}{2}l_2 \geq |X| = r_0 + 2r_1$.

Now note that every vertex that had degree 0 in G must either be in a 3-edge or have degree at most 1 in H. Also note that every component of G that has size 2 must have at least one vertex that is either in a 3-edge or has degree at most 1 in H. Otherwise, the two vertices are either the two endpoints of the same path in $\mathcal{P}$ which would form a cycle in the 2-edges of H, or endpoints of different paths in $\mathcal{P}$ which would violate the minimality of $\mathcal{P}$. This yields $l_1 + l_2 \leq 3t_3 + s$ where s is the number of vertices of degree at most 1 in H. Thus,

$$r_0 + 2r_1 \leq \frac{3}{2}t_2 + l_1 + \frac{1}{2}l_2 \leq \frac{3}{2}t_2 + l_1 + l_2 \leq \frac{3}{2}t_2 + 3t_3 + s.$$

Since $s \leq \frac{r}{3}$, H has a Y_r configuration. □

Lemma 5.21 *For any $\Delta, \epsilon > 0$, consider a random hypergraph H on n vertices with Δn 3-edges and $(\frac{1}{2} - \epsilon)n$ 2-edges where every such graph is equally likely. There is some constant $\alpha > 0$ such that a.s. H has no Y_r configuration for any $r < \alpha n$.*

Proof. Fix an $r < \alpha n$ and compute the expected number of Y_r configurations. Consider any list of 2-edges $e_1, \ldots, e_k$. The probability that they all appear in H is

$$\begin{aligned}
\frac{\binom{\binom{n}{2}-k}{(\frac{1}{2}-\epsilon)n-k}}{\binom{\binom{n}{2}}{(\frac{1}{2}-\epsilon)n}} &= \frac{\left(\binom{n}{2}-k\right)!\left(\left(\frac{1}{2}-\epsilon\right)n\right)!}{\binom{n}{2}!\left(\left(\frac{1}{2}-\epsilon\right)n-k\right)!} \\
&= \left(\frac{\left(\frac{1}{2}-\epsilon\right)n}{\binom{n}{2}}\right)\cdots\left(\frac{\left(\frac{1}{2}-\epsilon\right)n-k}{\binom{n}{2}-k}\right) \\
&\leq \left(\frac{\left(\frac{1}{2}-\epsilon\right)n}{\binom{n}{2}}\right)^k \\
&= \left(\frac{2\left(\frac{1}{2}-\epsilon\right)}{n-1}\right)^k \\
&< \left(\frac{2\left(\frac{1}{2}-\epsilon'\right)}{n}\right)^k
\end{aligned}$$

for some $0 < \epsilon' < \epsilon$.

As before, we let X be the set of vertices that are endpoints of the pendant paths of the Y_r configuration. Let r_0 be the number of paths of length 0, and let $r_1 = r - r_0$. We will use t_2 and t_3 to represent the number of 2- and 3-edges that are not on a pendant path of the Y_r configuration but whose vertices are all endpoints of the r pendant paths.

There are at most $\binom{n}{r} n^{r_1}$ choices for the endpoints of the r paths. Suppose the number of 2-edges in the paths are $l_1, \dots, l_r$, and let $L = l_1 + \cdots + l_r$. Then there are n^{L-r} choices for the interior vertices of the paths. We multiply by the probability that all L of these edges appear and that there are t_2 other 2-edges and t_3 3-edges on the endpoints. First, assume that t_2 and t_3 are both at least $\frac{r}{100}$. This gives an upper bound of

$$\begin{aligned}
&\sum_{l_1,\dots,l_r \geq 0} \binom{n}{r} n^{r_1} n^{L-r} \left(\frac{2(\frac{1}{2}-\epsilon')}{n}\right)^L \binom{(\frac{1}{2}-\epsilon)n}{t_2} \binom{\Delta n}{t_3} \left(\frac{|X|}{n}\right)^{2t_2+3t_3} \\
\leq\ & \left(\frac{ne}{r}\right)^r n^{r_1 - r} \left(\frac{\frac{1}{2}ne}{t_2}\right)^{t_2} \left(\frac{\Delta ne}{t_3}\right)^{t_3} \left(\frac{2r}{n}\right)^{2t_2+3t_3} \sum_{l_1,\dots,l_r} (1-2\epsilon')^L \\
\leq\ & \left(\frac{r}{n}\right)^{t_2+2t_3-r_1} e^{t_2+t_3+r} \Delta^{t_3} 2^{t_2+3t_3} 100^{t_2+t_3} \left(\sum_{l \geq 0} (1-2\epsilon')^l\right)^r \qquad (5.34) \\
\leq\ & \left(\frac{r}{n}\right)^{t_2+2t_3-r_1} \zeta^{3t_2+6t_3+2r} \\
\leq\ & \left(\frac{\gamma_1 r}{n}\right)^{t_2+2t_3-r_1} \qquad (5.35) \\
\leq\ & \left(\frac{\gamma_1 r}{n}\right)^{r_1/9} \qquad (5.36)
\end{aligned}$$

for some $\zeta, \gamma_1 > 0$. Inequality (5.34) follows because $t_2, t_3 \geq \frac{r}{100}$. Multiplying (5.33) by $\frac{2}{3}$ gives $t_2 + 2t_3 - \frac{4}{9}r_0 - \frac{10}{9}r_1 \geq 0$, and (5.35) follows because

$$\zeta^{3t_2+6t_3+2r} \leq \zeta^{3t_2+6t_3+2r+45\left(t_2+2t_3-\frac{4}{9}r_0-\frac{10}{9}r_1\right)} \leq \left(\zeta^{48}\right)^{t_2+2t_3-r_1}.$$

To get (5.36), we note that $t_2 + 2t_3 - r_1 = \frac{2}{3}(\frac{3}{2}t_2 + 3t_3) - r_1$, and from (5.33),

$$\frac{2}{3}(\frac{3}{2}t_2 + 3t_3) - r_1 \geq \frac{2}{3}(\frac{2}{3}r_0 + \frac{5}{3}r_1) - r_1 \geq \frac{2}{3}(\frac{5}{3}r_1) - r_1 = \frac{r_1}{9}.$$

If $t_2 \leq \frac{r}{100}$ then from (5.33), $t_3 \geq (\frac{2}{9} - \frac{1}{200})r_0 + (\frac{5}{9} - \frac{1}{200})r_1$. For such a t_2, compute the expected number of collections of r pendant paths along with t_3 3-clauses on their endpoints. As above, the expected number is upper bounded with:

$$\left(\frac{ne}{r}\right)^r n^{r_1-r}\left(\frac{e\Delta n}{t_3}\right)^{t_3}\left(\frac{|X|}{n}\right)^{3t_3}\left(\sum_{l\geq 0}(1-2\epsilon')^l\right)^r < \left(\frac{\gamma_2 r}{n}\right)^{r_1/10}$$

for some $\gamma_2 > 0$.

If $t_3 \leq \frac{r}{100}$ then from (5.33), $t_2 \geq (\frac{4}{9}-\frac{2}{100})r_0 + (\frac{10}{9}-\frac{2}{100})r_1$. For such a t_3, and similar to above, the expected number of collections of r pendant paths and t_2 2-edges on the endpoints is upper bounded by:

$$\left(\frac{ne}{r}\right)^r n^{r_1-r}\left(\frac{\frac{1}{2}ne}{t_2}\right)^{t_2}\left(\frac{|X|}{n}\right)^{2t_2}\left(\sum_{l\geq 0}(1-2\epsilon')^l\right)^r < \left(\frac{\gamma_3 r}{n}\right)^{r_1/11}$$

for some $\gamma_3 > 0$.

Let $\gamma = \max\{\gamma_1, \gamma_2, \gamma_3\}$. As there are $\mathrm{O}(r)$ choices for t_2, t_3, it suffices to show that

$$\sum_{r=1}^{\alpha n} r\left(\frac{\gamma r}{n}\right)^{r_1/11} = \mathrm{o}(1).$$

The first $\log n$ terms of this sum add up to at most $\mathrm{O}\left(\frac{\log n}{n^{1/11}}\right)$, and if $\alpha < \frac{1}{2\gamma}$ then the rest add up to at most $\sum_{i\geq \log n} i\left(\frac{1}{2}\right)^{i/11} = \mathrm{o}(1)$. □

The proof of this final lemma is an exercise in the second moment method on random graphs.

Lemma 5.22 *For any constants $\epsilon > 0$, $\zeta > 0$, and $\theta > 0$, a uniformly random graph with n vertices and $\left(\frac{1}{2}-\epsilon\right)n$ edges has a.s. at most $(1+\zeta)ne^2\theta^{-\theta}$ pendant paths of length θ.*

Proof. Using a well known property of random graphs, we can consider a model with n vertices and each of the $\binom{n}{2}$ edges existing independently with probability $p < \frac{1}{n}$.

Let X be the number of pendant paths of length θ. The expected value of X is bounded above by the number of choices for the θ vertices, the probability that each edge on the path exists, the probability there is no edge from the interior path vertices to the rest of the graph, and the probability the path is induced:

$$\begin{aligned}
\mathbf{E}(X) &\leq \binom{n}{\theta} p^{\theta-1}(1-p)^{(n-\theta)(\theta-2)}(1-p)^{\binom{\theta-1}{2}} \\
&\sim \left(\frac{ne}{\theta}\right)^{\theta}\left(\frac{1}{n}\right)^{\theta-1} e^{2-\theta} \\
&= ne^2\theta^{-\theta}.
\end{aligned}$$

Using the second moment method, we show the expected number is highly concentrated about its mean by summing over all sets of θ vertices that intersect with a given path multiplied by the probability that the intersecting set is also a pendant path. In the calculations below, k is the number of the θ vertices that intersect with the given path.

$$\begin{aligned}
\mathbf{E}(X^2) &= \mathbf{E}(X)\left[1+\sum_{k=1}^{\theta-1} 2\binom{n-\theta}{k} p^k(1-p)^{(n-\theta-k+1)k-1}(1-p)^{\binom{k}{2}}\right. \\
&\qquad \left. + \binom{n-\theta}{\theta} p^{\theta-1}(1-p)^{(n-2\theta+2)(\theta-2)}(1-p)^{\binom{\theta-1}{2}}\right] \\
&\sim \mathbf{E}(X)\left[1+\sum_{k=1}^{\theta-1} 2\left(\frac{ne}{k}\right)^k\left(\frac{1}{n}\right)^k e^{-k} + \mathbf{E}(X)\right] \\
&\sim \mathbf{E}(X)^2(1+\mathrm{o}(1)).
\end{aligned}$$

So by Chebyshev's Inequality, the probability that for any $\zeta > 0$, $X > (1+\zeta)\mathbf{E}(X)$ is $\mathrm{o}(1)$. □

5.5 The Proof of Theorem 5.1

Finally, we close this chapter with a proof of Theorem 5.1. The proof of Theorem 5.13 follows from the same analysis replacing the upper bound of UC with the upper bound of GUC* sketched after the proof of Lemma 5.16.

Theorem 5.1 *Given a random instance of* 3-*UE-CSP with* n *variables and* cn *clauses as input, DPLL+UC will take, w.u.p.p., linear time if* $c < \frac{2}{3}$ *and exponential time if* $c > \frac{2}{3}$.

Proof. The proof consists of four steps. From Lemma 5.9 presented in Section 5.2.1, if C is a random instance of 3-UE-CSP with n variables and cn clauses with $c < \frac{2}{3}$, then w.u.p.p. DPLL+UC will find a satisfying assignment without backtracking.

From Lemma 5.16 if $c > \frac{2}{3}$, executing the unit clause algorithm on C will w.u.p.p. produce a uniformly random subformula on n' variables with $\left(\frac{1}{2} - \epsilon\right) n'$ 2-clauses and $\left(\frac{1}{6} + \delta\right) n'$ 3-clauses where δ and ϵ are positive constants such that the largest solution to

$$x = (1 - e^{-x})^2 3 \left(\frac{1}{6} + \delta\right) + (1 - e^{-x}) 2 \left(\frac{1}{2} - \epsilon\right)$$

is greater than 0 and

$$\begin{aligned}1 - e^{-x} - xe^{-x} <& \\ &\left(\frac{1}{2} - \epsilon\right)(1 - e^{-x})^2 + \left(\frac{1}{6} + \delta\right)(1 - e^{-x})^3.\end{aligned}$$

From the proof of Theorem 5.4, we know that such a subformula has more edges than vertices and is a.s. unsatisfiable.

From Lemma 5.17 presented in Section 5.4, DPLL will require, w.u.p.p., $2^{\Omega(\gamma n)}$ steps to backtrack out of any random UE-CSP instance with $\left(\frac{1}{2} - \epsilon\right) n'$ 2-clauses and a linear number of 3-clauses. $\square$

Chapter 6

The Size of the Core for Non-Uniform Hypergraphs

In order to prove Theorem 5.4, we need to reduce a formula with a mixture of clauses of size 2 and 3 to its 2-core. Cores of graphs, and therefore of formulae, are well understood for graphs and uniform hypergraphs. The first paper to identify the threshold for the appearance of a k-core in a random graph and give its size was Pittel, Spencer and Wormald [PSW96]. Since then many papers have extended these results to random uniform hypergraphs and found alternative models and techniques for examining cores. These papers include Molloy [Mol05], Kim [Kim06], Cain and Wormald [CW06], Riodan [Rio07], and Darling and Norris [DN]. Still more papers examine cores for graphs with a given degree sequence. This list includes Cooper [Coo04], Fernholz and Ramachandran [FR03, FR04], and Jansen and Łuczak [JŁ06]. In this chapter, we simply extend the theorem of Molloy [Mol05] on r-cores of random uniform hypergraphs to random non-uniform hypergraphs, $r \geq 2$.

Theorem 6.1 *Let $r \geq 2$ and $k \geq 3$, for each i such that $2 \leq i \leq k$, let $c_i \geq 0$, and let x*

be the largest solution to

$$x = \sum_{i=2}^{k} \left(1 - e^{-x} \sum_{d=0}^{r-2} \frac{x^d}{d!}\right)^{i-1} ic_i. \tag{6.1}$$

If $x > 0$ *and*

$$\sum_{i=2}^{k} i(i-1) \left(1 - e^{-x} \sum_{d=0}^{r-2} \frac{x^d}{d!}\right)^{i-2} c_i e^{-x} \frac{x^{r-2}}{(r-2)!} < 1, \tag{6.2}$$

then a uniformly random hypergraph with $c_i n$ *i-edges for each* $2 \leq i \leq k$ *and no other edges a.s. has an* r*-core with* $\alpha(c_2, \ldots, c_k)n + \mathrm{o}(n)$ *vertices and* $\beta_i(c_2, \ldots, c_k)n + \mathrm{o}(n)$ *i-edges for each* $2 \leq i \leq k$ *where*

$$\alpha(c_2, \ldots, c_k) = 1 - e^{-x} \sum_{d=0}^{r-1} \frac{x^d}{d!},$$

and

$$\beta_i(c_2, \ldots, c_k) = c_i \left(1 - e^{-x} \sum_{d=0}^{r-2} \frac{x^d}{d!}\right)^{i}.$$

Note that the left hand side of (6.2) is the derivative of the right hand side of (6.1), and this derivative can never be greater than 1 at the largest solution to (6.1). So (6.2) can only fail if the derivative equals 1. This corresponds to all the clause densities aligning at exactly a threshold for the appearance of a core. In this case, the theorem does not guarantee the existence of a core in a hypergraph with those edge densities $c_2, \ldots, c_k$. However, increasing any c_i by an arbitrarily small positive ϵ will not change the sign of the largest solution to (6.1), and it will force the largest solution to (6.1) to satisfy inequality (6.2). So the theorem will guarantee that a random hypergraph, with this small change to c_i, a.s. has an r-core.

The proof of this theorem follows the same technique as the proof of the equivalent theorem for r-cores of uniform hypergraphs in [Mol05]. The above theorem is slightly weaker than the results of [Mol05] because [Mol05] actually proves the location of a sharp threshold for the appearance of a 2-core. In our case, the existence of the linear number of 2-edges implies that the appearance of a 2-core in mixed hypergraphs has a coarse threshold. The proof will analyze the behavior of the CORE procedure, defined in Section 2.5.1 for CSPs, on a non-uniform hypergraph.

Proof. The proof will give the a.s. size of an r-core by analyzing the behavior of a function that takes a hypergraph H as input and outputs the r-core of H.

> **CORE:** While the hypergraph has any vertices of degree less than r, choose an arbitrary such vertex and delete it along with all hyperedges containing it.

To simplify analysis, we will actually analyze two equivalent variations. The first variation, CORE1, removes all vertices of degree at most $r-1$ simultaneously in each round of the algorithm. The second variation, CORE2, removes one vertex of degree at most $r-1$ in each round, but it chooses that vertex uniformly at random from all vertices of degree at most $r-1$. Note that the order in which the vertices are removed in CORE does not affect the output of the function, and as a result, CORE, CORE1, and CORE2 will produce identical outputs when run on the same input hypergraph H.

The proof idea is to first use CORE1 to reduce H until we are left with a subgraph of H that has the desired size and only a small number of vertices of degree at most $r-1$. We then run CORE2 on this output to remove these last vertices of degree at most $r-1$.

We define the following recursion similar to a recursion in [Mol05]:

$$\begin{aligned}\rho_0 &= 1\\ \rho_j &= \mathbf{Pr}\left(Z\left(\sum_{i=2}^{k} i\rho_{j-1}^{i-1}c_i\right) \geq r-1\right),\end{aligned}$$

and define

$$\lambda_j = \mathbf{Pr}\left(Z\left(\sum_{i=2}^{k} i\rho_{j-1}^{i-1}c_i\right) \geq r\right)$$

where $Z(x)$ is a Poisson random variable with mean x. We then use the following lemma.

Lemma 6.2 *For any constant t, the probability that a vertex v survives after t rounds of CORE1 is $\lambda_t + \mathrm{o}(1)$.*

Proof. Let η be the set of vertices within distance t of v. Let E_1 be the event that $|\eta| < \log^2 n$ and the vertices of η induce a hypertree.

Claim 6.3 $\mathbf{Pr}(E_1) = 1 - \mathrm{o}(1)$.

Proof. Note that the expected number of i-edges incident to a vertex v is ic_i. As a result, the expected number of neighbors of v is $\sum_{i=2}^{k} i(i-1)c_i$. So, the expected number of vertices at distance exactly t of v is at most $\left(\sum_{i=2}^{k} i(i-1)c_i\right)^t$, and for all t this expected number is at most $\log n$. From Markov's inequality, the probability that there are at least $\log^2 n$ vertices within distance t from v is $\mathrm{o}(1)$.

To prove that the vertices at distance at most t from v a.s. induce a hypertree, start at v and do a breadth first traversal of the hypergraph exposing only those edges that belong to this breadth first tree rooted at v. Continue the breadth first traversal until it reaches a depth t from v. Now expose any additional edges between these vertices of the breadth first tree. The probability that there is an edge between any two vertices is

$$\sum_{i=2}^{k} \left(1 - \left(1 - \frac{c_i n}{\binom{n}{i}}\right)^{\binom{n-2}{i-2}}\right) = \Theta\left(n^{-1}\right).$$

As the total number of vertices in the breadth first tree is a.s. less than $\log^2 n$, the expected number of edges between any pair of these vertices is upper bounded by

$$\binom{log^2 n}{2} \Theta\left(n^{-1}\right) = \mathrm{o}(1),$$

and by Markov's inequality, the probability that there is at least one edge in addition to the edges of the breadth first tree is $\mathrm{o}(1)$.

Therefore E_1 holds with probability $1 - \mathrm{o}(1)$. $\square$

If E_1 holds then η induces a hypertree in H that is rooted at v. For any vertex u at distance j from v, define a *child edge* of u of size i to be a hyperedge of the form $(u, x_1, \ldots, x_{i-1})$ where each $x_1, \ldots, x_{i-1}$ is at distance $j+1$ from v. Note that if E_1 holds and $u \neq v$ then exactly one hyperedge containing u is not a child edge of u.

Now consider the variation of CORE, CORE_t, where in round j, for $1 \leq j \leq t-1$, only vertices at distance $t-j$ of v are considered, and a vertex is removed from H if it has no remaining child edge in H. Note that CORE_t will not consider vertices at distance at least t from H. If E_1 holds, then v remains after t rounds of CORE_t if and only if it remains after t rounds of CORE1.

Now we prove by induction on j, $0 \leq j \leq t-1$, that for any fixed α and β and any set of vertices $u_1, \ldots, u_\alpha, w_1, \ldots, w_\beta$ all at distance $t-j$ from v, the probability that the u's all survive and the w's all do not survive is $\rho_j^\alpha(1-\rho_j)^\beta + o(1)$.

For $j = 0$, nothing is removed, and the assertion holds. For $j \geq 1$, we will expose the vertices at distance $t-j$ from v by doing a breadth first traversal from v. Let η_j be the number of vertices exposed by this search. We expose the child edges from each u_i and w_i. We will say that a vertex *wins* if at least $r-1$ of its child edges survive. Note that if a vertex wins then it will survive, but the converse may not be true. However, if E_1 holds, then a vertex wins if and only if it survives.

If $|\eta_j| \geq \log^2 n$, we will set up the following experiment. The reason for the experiment is to make the probability of winning roughly the same regardless of whether E_1 holds. Then can bound the probability a vertex survives with the probability the vertex wins plus or minus the probability E_1 fails. We create $|\eta_j| - \log^2 n$ "special" vertices. For each "special" vertex we flip a coin, and the vertex survives with probability ρ_j. We create a set containing all vertices of H not in η_j and all of our "special" vertices. For each u_i and w_i, we consider every subset of size $i-1$ from this set such that the subset contains at least 1 "special" vertex. Then we flip a coin and with probability $\frac{c_i n}{\binom{n}{i}}$ we add that subset to the child edges of the vertex.

Let n' be the number of vertices not in η_j, including any "special" vertices. If $|\eta_j| < \log^2 n$, $n' = n - |\eta_j|$, and otherwise $n' = n - \log^2 n$.

Now, consider a vertex u from the set $u_1, \ldots, u_\alpha, w_1, \ldots, w_\beta$. Run CORE_t for $j-1$ rounds, and let μ be the expected number of child edges of u that survived CORE_t. This number is equal to the expected number of child edges containing u multiplied by the probability those edges survive, and the probability an edge survives is the probability the set of vertices, except u, in that edge survive. Assuming each edge survives independently, applying the induction hypothesis gives,

$$\mu = \sum_{i=2}^{k} \binom{n'}{i-1} \frac{c_i n}{\binom{n}{i}} \rho_{j-1}^{i-1}.$$

Because $n' = n - \mathrm{o}(n)$, this yields

$$\begin{aligned}\mu &= \sum_{i=2}^{k} \frac{n^{i-1}}{(i-1)!} \frac{c_i n}{\frac{n^i}{i!}} \rho_{j-1}^{i-1} + \mathrm{o}(1)\\ &= \sum_{i=2}^{k} i c_i \rho_{j-1}^{i-1} + \mathrm{o}(1).\end{aligned}$$

However, each edge does not survive independently. One child vertex may appear in more than one child edge, but the probability this happens is o(1). Also, the probability a child vertex survives is not independent of whether the other child vertices survive. However, the induction hypothesis implies that this dependence contributes only o(1) to the probabilities. If we were to compute the probability stated in the induction hypothesis, we would get a large sum over all possibilities and each term of this sum will be of the same form as the induction hypothesis: this set of vertices survives and that does not. In addition, by the same argument on the child vertices and the fact that the probability one child vertex is in more than one child edge is o(1), we see that the probability u wins affects the probability the other u_i and w_i vertices win by o(1).

Now we show that the distribution of the number of child edges of u that survive after j rounds of CORE$_t$ is asymptotic to a Poisson variable, and to do this we use the following lemma from Janson, Łuczak, and Ruciński [JŁR00].

Lemma 6.4 (Corollary 6.8 of [JŁR00]) *Let $S_n = \sum_{\alpha \in A_n} I_{n,\alpha}$ be sums of indicator variables $I_{n,\alpha}$. If $\mu \geq 0$ is such that, as $n \to \infty$,*

$$\sum_{\alpha_1,\ldots,\alpha_l} \mathbf{Pr}(I_{n,\alpha_1} = \cdots = I_{n,\alpha_l} = 1) \to \mu^l, \tag{6.3}$$

for every $l \geq 1$ where the sum is over distinct indices, then S_n is asymptotic to a Poisson random variable with mean μ.

For each i, we will apply the lemma separately to the set of i-edges by defining A_n to be the set of i-edges that are possible child edges of u. Since u is at distance j from v, A_n is the set of all subsets of i vertices containing u but not containing a vertex from η_j. Let $I_{n,\alpha}$ be an indicator variable that is 1 if and only if edge α is a child edge of u after j iterations of CORE_t.

The total number of ordered subsets of size l of possible child i-edges is

$$l!\binom{\binom{n'}{i-1}}{l},$$

and each i-edge exists in H with probability $\frac{i!c_i+\mathrm{o}(1)}{n^{i-1}}$. From the argument above concerning survival independence and overlapping edges, each i-edge survives j rounds of CORE_t with probability $\rho_{j-1}^{i-1}+\mathrm{o}(1)$. Thus, the sum (6.3) tends to $\left(ic_i\rho_{j-1}^{i-1}+\mathrm{o}(1)\right)^l$, and by Lemma 6.4, the number of i-edges is asymptotic to a Poisson random variable with mean $ic_i\rho_{j-1}^{i-1}+\mathrm{o}(1)$.

Since the sum of $k-1$ Poisson variables with means $\mu_2,\ldots,\mu_k$ is a Poisson random variable with mean $\mu_2+\cdots+\mu_k$, the distribution of the number of child edges of u that survive j rounds of CORE_t is asymptotic to a Poisson random variable with mean $\mu=\sum_{i=2}^{k} ic_i\rho_{j-1}^{i-1}+\mathrm{o}(1)$.

Because $n'=n-\mathrm{o}(n)$, we can bound the probability that u has at least $r-1$ child edges by $\mathbf{Pr}(Z(\mu)\geq r-1)+\mathrm{o}(1)=\rho_j+\mathrm{o}(1)$.

Finally, we note that

$$\mathbf{Pr}(u \text{ wins})\times\mathbf{Pr}(E_1 \text{ holds})\leq\mathbf{Pr}(u \text{ survives})\leq\mathbf{Pr}(u \text{ wins}),$$

and so $\mathbf{Pr}(u \text{ survives})=\rho_j+\mathrm{o}(1)$. A similar analysis for the rest of the vertices in the set $u_1,\ldots,u_\alpha,w_1,\ldots,w_\beta$ completes the induction.

Vertex v survives the final round of CORE_t if and only if at least 2 edges incident to v remain after $t-1$ rounds. By the same analysis, the expected number of edges incident to v that survive $t-1$ rounds of CORE$_t$ is

$$\sum_{i=2}^{k} \binom{n-1}{i-1} \frac{c_i n}{\binom{n}{i}} \rho_{t-1}^{i-1} = \sum_{i=2}^{k} i c_i \rho_{t-1}^{i-1} + \mathrm{o}(1),$$

and the number of such edges is asymptotic to a Poisson random variable. Therefore, conditional on E_1 holding, the probability v survives t rounds of CORE$_t$ is $\lambda_t + \mathrm{o}(1)$. Since $\mathbf{Pr}(E_1) = 1 - \mathrm{o}(1)$, v survives t rounds of CORE1 with probability $\lambda_t + \mathrm{o}(1)$. $\square$

For any integer $t > 0$, let H_t be the subhypergraph of H that remains after t rounds of CORE1. Following the technique of [Mol05], we prove that the size of H_t is highly concentrated about its expected size.

Lemma 6.5 *For any constant t, a.s. $||H_t| - \lambda_t n| = \mathrm{o}(n)$.*

Proof. The proof will use the second moment method and closely follows the proof of Goerdt and Molloy [GM03] for a similar result on random regular graphs.

For a vertex i, let X_i be the indicator variable that is 1 if vertex i survives t rounds of CORE1 and 0 otherwise. Then $|H_t| = X = \sum_{i=1}^{n} X_i$.

$$\begin{aligned}
\mathbf{Var}(X) &= \mathbf{E}(X^2) - (\mathbf{E}(X))^2 \\
&= \mathbf{E}\left(\sum_{i,j} X_i X_j\right) - (\mathbf{E}(X))^2 \\
&= \mathbf{E}\left(\sum_{i} X_i^2 + \sum_{i \neq j} X_i X_j\right) - (\mathbf{E}(X))^2 \\
&= \mathbf{E}\left(\sum_{i} X_i^2\right) + \sum_{i \neq j} \mathbf{E}\left(X_i X_j\right) - (\mathbf{E}(X))^2 && \text{by linearity of expectation.}
\end{aligned}$$

So,

$$\begin{aligned}
\mathbf{Var}(X) &= \mathbf{E}(X) + \sum_{i \neq j} \mathbf{E}\left(X_i X_j\right) - (\mathbf{E}(X))^2 \\
&= \mathbf{E}(X) + n(n-1)\mathbf{E}\left(X_u X_v\right) - (\mathbf{E}(X))^2 && \text{by symmetry} \\
&= \mathbf{E}(X) + n(n-1)\mathbf{Pr}(X_u X_v = 1) - (\mathbf{E}(X))^2
\end{aligned}$$

where u and v are arbitrary vertices.

Let η_u and η_v be the set of vertices within t of u and v, respectively. Similar to event E_1 above, let E_u be the event that $|\eta_u| < \log^2 n$ and the vertices of η_u induce a hypertree. Likewise, define E_v similarly, and we will assume both E_u and E_v hold.

$$\begin{aligned}\mathbf{Pr}(X_uX_v = 1) = \mathbf{Pr}(X_uX_v = 1 \mid \eta_u \cap \eta_v = \emptyset)\mathbf{Pr}(\eta_u \cap \eta_v = \emptyset)\\ + \mathbf{Pr}(X_uX_v = 1 \mid \eta_u \cap \eta_v \neq \emptyset)\mathbf{Pr}(\eta_u \cap \eta_v \neq \emptyset).\end{aligned}$$

Note that if $\eta_u \cap \eta_v = \emptyset$ then the probability that u survives is independent from the probability that v survives. To compute $\mathbf{Pr}(\eta_u \cap \eta_v \neq \emptyset)$, expose η_u. Now expose each vertex of η_v one at a time. The probability that we expose an element of η_u is at most

$$1 - \left(1 - \frac{|\eta_u|}{n - |\eta_u| - |\eta_v|}\right)^{|\eta_v|}.$$

If E_u and E_v hold, then $|\eta_u|, |\eta_v| < \log^2 n$, and thus this probability is $o(1)$. By Lemma 6.2, we have

$$\begin{aligned}\mathbf{Pr}(X_uX_v = 1) &= (\lambda_n + o(1))^2(1 - o(1)) + \mathbf{Pr}(X_uX_v = 1 \mid \eta_u \cap \eta_v \neq \emptyset) \times o(1)\\ &= \lambda_n^2 + o(1).\end{aligned}$$

This plus the fact that E_u and E_v fail to hold with probability $o(1)$ gives

$$\begin{aligned}\mathbf{Var}(X) &\leq \mathbf{E}(X) + n(n-1)\left(\lambda_n^2 + o(1)\right) - (\mathbf{E}(X))^2\\ &\leq n(\lambda_n + o(1)) + n^2\left(\lambda_n^2 + o(1)\right) - (n(\lambda_n + o(1)))^2\\ &= o\left(n^2\right).\end{aligned}$$

To complete the proof, we show that there exists a function $f(n) = o(n)$ such that

$$\mathbf{Pr}(||H_t| - \lambda_t n| \geq f(n)) = o(1).$$

Since $\mathbf{Var}(X) = o\left(n^2\right)$, we have $\sqrt{\mathbf{Var}(X)} = \frac{n}{\omega(n)}$ where $\lim_{n\to\infty}\omega(n) = \infty$. Let

$$f(n) = \sqrt{\mathbf{Var}(X)\omega(n)}.$$

By Chebychev's Inequality,

$$\mathbf{Pr}(|X - \mathbf{E}(X)| \geq f(n)) \leq \frac{1}{\omega(n)} = \mathrm{o}(1).$$

Recalling that $X = |H_t|$ and $\mathbf{E}(X) = \lambda_t n$ completes the proof. □

Lemma 6.6 *For any $\delta > 0$ and for x defined in (6.1), there exists $I = I(\delta, x)$ such that if $x > 0$ then*

(a) if we run CORE1 for I rounds, a.s. $\alpha(c_2, \ldots, c_k)n < |H_I| < \alpha(c_2, \ldots, c_k)n + \delta n$, and

(b) if after iteration I we run CORE2 until all vertices of degree at most $r-1$ are removed, a.s. CORE2 will remove at most δn vertices after iteration I.

The proof again follows the technique of [Mol05].

Proof. First we show that the sequence $\rho_0, \rho_1, \ldots$ is non-increasing. Note that $\rho_1 \leq \rho_0$ by definition, and if $\rho_j \leq \rho_{j-1}$ then $\rho_{j+1} \leq \rho_j$ because if the expected value of a Poisson random variable does not increase then neither can the probability that the variable is greater than $r-1$. Let $\rho = \lim_{t\to\infty} \rho_t$. Since each ρ_t is non-negative and the sequence $\rho_0, \rho_1, \ldots$ is non-increasing, this limit exists. Furthermore, it must satisfy the largest solution to

$$\rho = \mathbf{Pr}\left(Z\left(\sum_{i=2}^{k} i\rho^{i-1}c_i\right) \geq r-1\right). \tag{6.4}$$

Let $x = \sum_{i=2}^{k} i\rho^{i-1}c_i$, and by substituting we get

$$\begin{aligned} x &= \sum_{i=2}^{k} i\left(\mathbf{Pr}(Z(x) \geq r-1)\right)^{i-1} c_i \\ &= \sum_{i=2}^{k} i\left(1 - e^{-x}\sum_{d=0}^{r-2}\frac{x^d}{d!}\right)^{i-1} c_i. \end{aligned} \tag{6.5}$$

Assume (6.5) has a positive solution for x. (6.5) may have more than one solution for x; let x_1 be the largest solution. Consider the function

$$\tau(\rho) = \mathbf{Pr}\left(Z\left(\sum_{i=2}^{k} i\rho^{i-1}c_i\right) \geq r-1\right).$$

It is straightforward to see that increasing any c_i will increase the value of $\tau(\rho)$ for any $\rho > 0$. As a result, the largest solution to (6.4) will increase for any increase in a c_i. It is straightforward to see that x_1 is the only solution to (6.5) that will increase on any increase to a c_i. Therefore, ρ satisfies $x_1 = \sum_{i=2}^{k} i\rho^{i-1}c_i$, and $\lim_{t\to\infty}\lambda_t = \mathbf{Pr}\left(Z(x_1) \geq r\right) = \alpha(c_2, \dots, c_k) > 0$. Therefore, for I sufficiently large in terms of x and δ, we have $\alpha(c_2, \dots, c_k) < \lambda_I < \alpha(c_2, \dots, c_k) + \delta$. This plus Lemma 6.5 proves part (a).

To prove part (b), we follow the technique of [Mol05] and expose the vertices of H_I and their degrees.

Claim 6.7 *Every hypergraph with that vertex set and degree sequence is equally likely to be H_I.*

The proof of Claim 6.7 follows exactly the same logic as the proof of Fact 4.5 in Section 4.4.1 and is omitted.

We will draw H_I uniformly randomly from the configuration model. The configuration model is defined by Bollobás [Bol80] and based on Bender and Canfield [BC78]. Independently, Wormald [Wor78, Wor81a, Wor81b] defines an equivalent model in a slightly different form that is based on the more general model of Békéssy, Békéssy, and Komlós [BBK72]. To generate a uniformly random element of this model, we make $d(v)$ copies of vertex v where $d(v)$ is the degree of v. Then we take a uniformly random partition of all the copies into sets of sizes 2 through k, each set corresponding to a hyperedge on the vertices whose copies are in the set. It is possible that this technique will form a multigraph. For each i, using the techniques from [Bol80, Wor78, Wor81a, Wor81b] for graphs and from [CFMR96] for hypergraphs, we know that we can lower bound with a positive constant the probability the i-edges alone do not induce a multigraph. The only other way to form a multigraph is if some j-edge is a subset of some i-edge with $i > j$. However, there are $\binom{i}{j}$ sets of variables of size j that are contained in each i-edge, and the probability that none of the j-edges are one of these at most $\binom{i}{j}c_i n$ sets is at least

$$\left(1-\frac{\binom{i}{j}c_i n}{\binom{n}{j}}\right)^{c_j n},$$

and as n tends to infinity this probability tends to 1 if $j > 2$, and the probability tends to $e^{-j!\binom{i}{j}c_2c_3}$ if $j = 2$. Therefore, we can bound the probability that the resulting hypergraph is a simple hypergraph by a constant, and this yields that results holding a.s. for the random configuration also hold a.s. for H_I.

For each $d, I \geq 0$, let $\gamma_{d,I}n$ be the number of vertices of degree d in H_I. For any vertex v, the probability that $v \in H_I$ and the degree of v in H_I is less than r is equal to the probability that v is in H_I but not H_{I+1}. By Lemma 6.2, a.s. the number of such vertices is at most $(\lambda_I - \lambda_{I+1})n + \mathrm{o}(n)$. Since $\lim_{i\to\infty}\lambda_i$ exists, we can make $(\gamma_{0,I} + \gamma_{1,I})$ arbitrarily small by taking I sufficiently large. For any $d \geq r$, since $\lim_{i\to\infty}\rho_i = \rho$, the expected number of edges incident to a vertex in H_I approaches $\sum_{i=2}^{k} ic_i\rho^{i-1} + \mathrm{o}(1)$ and since the number of incident edges is asymptotic to a Poisson random variable, a.s.

$$\gamma_{d,I} = \mathbf{Pr}\left(Z\left(\sum_{i=2}^{k} i\rho_I^{i-1}c_i\right) = d\right) + \mathrm{o}(1).$$

By taking I sufficiently large, we can make this value arbitrarily close to

$$\gamma'_d = e^{-x}\frac{x^d}{d!}.$$

Now consider the procedure CORE2 that starts with the graph H_I resulting from I rounds of CORE1. CORE2 will then remove one vertex chosen uniformly at random from all vertices of degree at most $r-1$. We let $X_0 = \left(\sum_{i=0}^{r-1}\gamma_{i,I}\right)n$, i.e. the number of vertices of degree less than r in H_I, and for each $j > 0$, we let X_j be the number of such vertices remaining after the jth iteration of CORE2. Similar to [Mol05], we will show that there is some ψ such that $0 < \psi < \delta$ and such that a.s. the sequence $X_0, X_1, X_2, \ldots$ will drift to $X_j = 0$ for some $j \leq \psi n$. This implies that CORE2 will halt at the jth iteration and will have found a 2-core after removing at most $\psi n < \delta n$ vertices.

During the first iteration of CORE2, we delete at most $r-1$ hyperedges. Expose the vertices in those hyperedges. Each has degree exactly r with probability $\frac{r\gamma_{r,I}}{\sum_{d\geq 1} d\gamma_{d,I}} + \mathrm{o}(1)$. By taking I sufficiently large so X_0 will be arbitrarily small, we can make this probability arbitrarily close to

$$\frac{r\gamma'_r}{\sum_{d\geq r} d\gamma'_d} = \frac{re^{-x}\frac{x^r}{r!}}{\sum_{d\geq r} de^{-x}\frac{x^d}{d!}} = \frac{\frac{x^r}{(r-1)!}}{\sum_{d\geq r}\frac{x^d}{(d-1)!}} = \frac{\frac{x^{r-1}}{(r-1)!}}{\sum_{d\geq r-1}\frac{x^d}{d!}} = \frac{\frac{x^{r-1}}{(r-1)!}}{e^x - \sum_{d=0}^{r-2}\frac{x^d}{d!}}. \tag{6.6}$$

The next step is to show that this probability multiplied by the expected number of neighbors of a vertex v of degree less than r is less than 1. We can calculate the expected number of neighbors of v as follows. Fix an edge of size i, for I sufficiently large and for each vertex in that edge, the probability that vertex survives I rounds of CORE1 can be made arbitrarily close to $\rho = (1 - e^{-x}\sum_{d=1}^{r-2}\frac{x^d}{d!})$, and the probability the edge survives is ρ^i, the probability each vertex of the edge survives. Therefore, after I rounds of CORE, there are $\rho^i c_i + \mathrm{o}(1)$ i-edges that have survived, for each $i \in \{2, \ldots, k\}$. By definition of the configuration model, given a vertex v if we choose a uniformly random edge incident to v, the probability that edge has size i is equal to

$$\frac{i \times \{\text{the number of } i \text{ edges}\}}{\{\text{the sum of degrees in the graph}\}},$$

and the expected number of neighbors of v is

$$(\text{degree of } v) \times \left(\sum_{i=2}^{k}(i-1) \times \mathbf{Pr}(v \text{ is in a i-edge})\right).$$

If the degree of v is at most $r-1$, the expected number of neighbors is at most

$$\begin{aligned}
&(r-1)\frac{\sum_{i=2}^{k}(i-1)i\rho^i c_i}{\sum_{i=2}^{k} i\rho^i c_i} \\
&= (r-1)\frac{\sum_{i=2}^{k}(i-1)i\rho^{i-2} c_i}{\sum_{i=2}^{k} i\rho^{i-2} c_i} \\
&= (r-1)\frac{\sum_{i=2}^{k}(i-1)i\left(1 - e^{-x}\sum_{d=0}^{r-2}\frac{x^d}{d!}\right)^{i-2} c_i}{\sum_{i=2}^{k} i\left(1 - e^{-x}\sum_{d=0}^{r-2}\frac{x^d}{d!}\right)^{i-2} c_i}.
\end{aligned} \tag{6.7}$$

Expanding on the technique of [Mol05], from (6.2),

$$\begin{aligned}
1 &> \sum_{i=2}^{k}(i-1)i\left(1-e^{-x}\sum_{d=0}^{r-2}\frac{x^d}{d!}\right)^{i-2} c_i e^{-x}\frac{x^{r-2}}{(r-2)!} \\
&= \frac{\sum_{i=2}^{k}(i-1)i\left(1-e^{-x}\sum_{d=0}^{r-2}\frac{x^d}{d!}\right)^{i-2} c_i}{\sum_{i=2}^{k} i\left(1-e^{-x}\sum_{d=0}^{r-2}\frac{x^d}{d!}\right)^{i-2} c_i} \cdot \frac{\frac{x^{r-2}}{(r-2)!}\sum_{i=2}^{k} i\left(1-e^{-x}\sum_{d=0}^{r-2}\frac{x^d}{d!}\right)^{i-2} c_i}{e^x} \\
&= \frac{\sum_{i=2}^{k}(i-1)i\left(1-e^{-x}\sum_{d=0}^{r-2}\frac{x^d}{d!}\right)^{i-2} c_i}{\sum_{i=2}^{k} i\left(1-e^{-x}\sum_{d=0}^{r-2}\frac{x^d}{d!}\right)^{i-2} c_i} \cdot \frac{\frac{x^{r-2}}{(r-2)!}\sum_{i=2}^{k} i\left(1-e^{-x}\sum_{d=0}^{r-2}\frac{x^d}{d!}\right)^{i-1} c_i}{e^x\left(1-e^{-x}\sum_{d=0}^{r-2}\frac{x^d}{d!}\right)} \\
&= (r-1)\frac{\sum_{i=2}^{k}(i-1)i\left(1-e^{-x}\sum_{d=0}^{r-2}\frac{x^d}{d!}\right)^{i-2} c_i}{\sum_{i=2}^{k} i\left(1-e^{-x}\sum_{d=0}^{r-2}\frac{x^d}{d!}\right)^{i-2} c_i} \cdot \frac{\frac{x^{r-1}}{(r-1)!}}{\left(e^x-\sum_{d=0}^{r-2}\frac{x^d}{d!}\right)}. \qquad (6.8)
\end{aligned}$$

Note that (6.8) is exactly (6.7) multiplied by (6.6). As a result, there is some $\kappa = \kappa(x) > 0$ such that for sufficiently large constant I, the expected number of vertices of degree r that are adjacent to each vertex of degree less than r is at most $(1-\kappa)$.

Even if an adversary can deterministically choose the degrees of the at most $(k-1)(r-1)j$ vertices whose degrees are altered during the first j iterations of CORE2, for each d the number of vertices of degree d will be arbitrarily close to the range $\gamma_d' n \pm (k-1)(r-1)j$. In particular for I sufficiently large, the probability a vertex is degree r after j iterations is arbitrarily close to

$$\frac{2\gamma_2 \pm 2\frac{(k-1)(r-1)j}{n}}{\sum_{d\geq r} d\gamma_d'} = \frac{\frac{x^{r-1}}{(r-1)!}}{e^x-\sum_{d=0}^{r-1}\frac{x^d}{d!}} \pm \frac{2(k-1)(r-1)j}{\left(e^x-\sum_{d=0}^{r-1}\frac{x^d}{d!}\right)e^{-x}xn}.$$

Therefore, if

$$\psi \leq \frac{\kappa x^2}{4(k-1)(r-1)^2\sum_{i=0}^{k}(i-1)i\left(1-e^{-x}\sum_{d=0}^{r-2}\frac{x^d}{d!}\right)^{i-2} c_i}, \qquad (6.9)$$

then for each $j < \psi n$, even if the adversary can deterministically choose the degrees of the at most $(k-1)(r-1)j$ vertices whose degrees are altered during the first j iterations, the expected number of degree r vertices whose degree decreases during iteration $j+1$ is at most $\left(1-\frac{\kappa}{2}\right)$.

When CORE2 deletes a vertex v, a.s. the degree of each neighbor of v will decrease by 1. For the degree of a neighbor y of v to decrease by more than 1, y must be contained in more than one edge incident to x. Since the number of edges incident to v is asymptotic to a Poisson distribution with constant expectation, the probability that some variable y occurs more than once in this neighborhood is $\mathrm{O}\left(\frac{1}{n}\right)$. Therefore $\mathbf{E}(X_1 - X_0) \le -\kappa$ for some $\kappa > 0$, and furthermore, for each $j < \psi n$, if $X_j > 0$ then $\mathbf{E}(X_{j+1} - X_j) \le -\frac{\kappa}{2}$. We pick I large enough such that $\lambda_I - \lambda_{I+1} < \frac{\psi\kappa}{8}$, $\psi < \delta$, and (6.9) hold.

By Lemma 6.5, a.s. $X_0 < \left(\frac{\psi\kappa}{8}\right) n + \mathrm{o}(n)$. In particular, we have a.s. $X_0 < \left(\frac{\psi\kappa}{4}\right) n$. The final step of the proof is to show that a.s. the sequence $X_0, X_1, X_2, \ldots$ drifts to $X_j = 0$ for some $j \le \psi n$. At this point CORE2 will halt and output a k-core, removing at most $\psi n < \delta n$ vertices in the process. This proves part (b).

Following the ideas of [Mol05], we prove that $X_0, X_1, \ldots$ drifts to zero quickly by coupling it to another random sequence: $Y_0, Y_1, \ldots$. For each $i \in \{0, \ldots, (r-1)(k-1)-1\}$, let

$$\xi_i = \max_{j < \psi n}\{X_{j+1} - X_j = i \mid X_j > 0\},$$

and let

$$\xi_{-1} = 1 - \sum_{i=0}^{(r-1)(k-1)-1} \xi_i.$$

By the definition of ψ, we have

$$\sum i\xi_i \le -\frac{\kappa}{2}.$$

Define $Y_0 = X_0$, and for each $i \in \{-1, \ldots, (r-1)(k-1)-1\}$, $Y_{j+1} = Y_j + i$ with probability ξ_i. We can couple the sequences so that for each $j < \psi n$, if $X_j > 0$ and $X_{j+1} - X_j \ne -1$ then $Y_{j+1} - Y_j = X_{j+1} - X_j$. Thus for each $j \le \psi n$, $X_i \le \max\{Y_j, 0\}$. Note that

$$\mathbf{E}(Y_{\psi n}) \le Y_0 - \left(\frac{\kappa}{2}\right)\psi n \le -\left(\frac{\psi\kappa}{4}\right) n.$$

Since changing any one choice from the sequence $Y_0, Y_1, \ldots$ can affect $Y_{\psi n}$ by at most $(r-1)(k-1)$, Lemma 1.3, Azuma's Inequality, implies that

$$\mathbf{Pr}\left(Y_{\psi n} \geq -\frac{\psi\kappa n}{8}\right) \leq e^{-\left(\frac{\psi\kappa n}{8}\right)^2/2\psi n(r-1)(k-1)} = e^{-\psi\kappa^2 n/128(k-1)^2(r-1)^2} = \mathrm{o}(1),$$

since ψ, κ, $r-1$, and $k-1$ are all positive constants. Thus a.s. $Y_{\psi n} \leq -\left(\frac{\psi\kappa}{8}\right)n$ and a.s. $X_{\psi n} = 0$. $\square$

Therefore, the probability a vertex survives CORE is $\lim_{t\to\infty}\lambda_t = \alpha(c_2, c_3)$. Since $\rho = \lim_{t\to\infty}\rho_t$ is the probability a vertex in a given edge survives CORE, we have $\beta_2(c_2, c_3) = c_2\rho^2$ and $\beta_3(c_2, c_3) = c_3\rho^3$ as desired. $\square$

Chapter 7

Conclusion

This thesis studied several threshold phenomena in random constraint problems including the satisfiability threshold and the thresholds for algorithm behavior. We defined a new family of constraint satisfaction problems that appears to be very similar to SAT. If the Maximum Hypothesis is proven, then $(3, d)$-UE-CSP will become the first problem that is known to have all of the following properties: it is NP-complete, a uniformly random instance with n variables and cn clauses, $c > 0$, a.s. has high resolution complexity, its random model has a known exact satisfiability threshold, and the satisfiability threshold occurs when the number of clauses is linear in the number of variables. Therefore, an important task is to resolve the Maximum Hypothesis, Hypothesis 4.2, or to find an alternative proof of the satisfiability threshold for $(3, d)$-UE-CSP.

The reasons for studying uniquely extendible constraints are many. First is that uniquely extendible constraints have a nice connection to the well studied combinatorial structures of quasigroups, Latin squares, and generalized Latin squares. Second is that extending our knowledge of uniquely extendible constraints improves our understanding of XOR-SAT. Third is that comparing the behavior of random UE-CSP with random SAT improves our understanding of both problems. While this thesis does not improve on any of the known results for random SAT, it does suggest areas for additional research.

The first such area is resolving the Satisfiability Threshold Conjecture, Conjecture 1.1. The differences between SAT and UE-CSP seem to relate to the differences between directed and undirected graphs. For 2-SAT and 2-UE-CSP, the relation is obvious because to find the 2-UE-CSP satisfiability threshold location, we translate a random 2-UE-CSP instance into an undirected graph and find the threshold that indicates the a.s. appearance of a cycle in the graph. For 2-SAT, we find the satisfiability threshold location by translating a random problem instance into a directed graph, and we find the clause density at which a strongly connected component a.s. appears in the directed graph. A strongly connected component is a directed analog of an undirected cycle because both are the minimum structures required for a vertex to be connected to itself by a non-empty vertex disjoint path. For larger clause sizes, a similar relationship may hold. To find the satisfiability threshold for k-UE-CSP, we translate a random k-UE-CSP instance into a k-uniform hypergraph and the satisfiability threshold for k-UE-CSP occurs at the same point as the threshold for when the 2-core of the k-uniform hypergraph a.s. has as many hyperedges as vertices. It may be that a similar structure using some notion of directed hyperedge also holds for k-SAT.

A second topic is to resolve the $(2+p)$ Threshold Conjecture for SAT, Conjecture 5.3. If the $(2+p)$ Threshold Conjecture holds for SAT, this will lend strong evidence that SAT and UE-CSP have very similar behavior in the random model. If the $(2 + p)$ Threshold Conjecture does not hold for SAT then the proof should illuminate key differences between random SAT and random UE-CSP and lead to a better understanding of both problems. However, it may be that we will not resolve the $(2 + p)$ Threshold Conjecture for SAT until we have a tight upper bound on the satisfiability threshold for 3-SAT.

A third area to explore concerns the behavior of survey propagation. While most of the current research is focused on proving survey propagation behavior for SAT, studying the behavior of survey propagation on XOR-SAT, or UE-CSP in general, may lead to some important results. Survey propagation currently fails on XOR-SAT above the threshold for the 2-core. The primary reason survey propagation behavior is different in the two problems is that SAT has an asymmetry in its solutions that XOR-SAT does not: exactly one variable in each clause must be set appropriately. However, another difference in the two problems is that once a 2-core appears in XOR-SAT, a.s. every variable that occurs in the 2-core is constrained in each solution cluster. On the other hand, for SAT we expect a large number of variables from the 2-core to remain unconstrained in a solution cluster. However, evidence for SAT suggests that as the clause density approaches the satisfiability threshold, the size of a solution cluster decreases as the number of variables from the 2-core that remain unconstrained in the cluster decreases. In particular, it is proven for k-SAT, with large k, as the clause density approaches the conjectured satisfiability threshold, the solution clusters become arbitrarily small and maximally far apart [ART06]. This result suggests that if the number of constrained versus unconstrained variables plays a role in the success of survey propagation, then if we are unable to improve the behavior of survey propagation on XOR-SAT, we may be unsuccessful in getting survey propagation to succeed w.u.p.p. on SAT up to the satisfiability threshold. On the other hand, it may be that the asymmetry of SAT is enough to find solutions even when the cluster sizes are arbitrarily small.

A fourth area for research is to prove a bound on the behavior of a random walk algorithm on XOR-SAT. Such a proof may likely lead to a better bound for random walk on SAT. The current SAT bound is based on the pure literal rule, and to improve it we need some technique that does not depend on always finding pure literals.

A fifth topic is to explore the threshold behavior of UE-CSP when we restrict the number of constraints or the properties of those constraints that may be assigned to a clause. If we consider a k-SAT clause to induce a constraint on its k variables, the number of possible constraints that can be placed on a set of k variables is exponential in k while the number of possible (k, d)-UE-CSP constraints is exponential in both k and d. The current random model has all possible uniquely extendible constraints occurring with equal probability. However, only a small number of totally symmetric, medial constraints are actually required to have an NP-complete problem. It may be interesting to see how the threshold behavior changes as the number of constraints is restricted.

Bibliography

[ABM04a] Dimitris Achlioptas, Paul Beame, and Michael Molloy. Exponential bounds for DPLL below the satisfiability threshold. In *Proceedings of the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 132–133, 2004.

[ABM04b] Dimitris Achlioptas, Paul Beame, and Michael Molloy. A sharp threshold in proof complexity yields lower bounds for satisfiability search. *Journal of Computer and System Sciences*, 68:238–268, 2004.

[ABS03] Mikhail Alekhnovich and Eli Ben-Sasson. Linear upper bounds for random walk on small density random 3-CNFs. In *Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science*, pages 352–361, 2003.

[Ach00] Dimitris Achlioptas. Setting two variables at a time yields a new lower bound for random 3-SAT. In *Proceedings of the Thirty-Second Annual ACM Symposium of Theory of Computing*, pages 28–37, 2000.

[Ach01] Dimitris Achlioptas. A survey of lower bounds for random 3-SAT via differential equations. *Theoretical Computer Science*, 256(1–2):159–185, 2001.

[ACIM01] Dimitris Achlioptas, Arthur Chtcherba, Gabriel Istrate, and Cristopher Moore. The phase transition in 1-in-k SAT and NAE 3-SAT. In *Proceed-*

ings of the Twelfth Annual ACM-SIAM Symposium on Discrete Algorithms, pages 721–722, 2001.

[AJPU02] Michael Alekhnovich, Jan Johannsen, Toniann Pitassi, and Alasdair Urquhart. An exponential separation between regular and general resolution. In *Proceedings of the Thirty-Fourth Annual ACM Symposium of Theory of Computing*, pages 448–456, 2002.

[AKKK01] Dimitris Achlioptas, Lefteris M. Kirousis, Evangelos Kranakis, and Danny Krizanc. Rigorous results for random $(2 + p)$-SAT. *Theoretical Computer Science*, 265(1–2):109–129, 2001.

[AKV04] Albert Atserias, Phokion G. Kolaitis, and Moshe Y. Vardi. Constraint propagation as a proof system. In *Proceedings of the Tenth International Conference on Principles and Practice of Constraint Programming*, 2004.

[AM06] Dimitris Achlioptas and Cristopher Moore. Random k-SAT: Two moments suffice to cross a sharp threshold. *SIAM Journal of Computing*, 36(3):740–762, 2006.

[AMK$^+$01] Dimitris Achlioptas, Michael S. O. Molloy, Lefteris M. Kirousis, Yannis C. Stamatious, Evangelos Kranakis, and Danny Krizanc. Random constraint satisfaction: A more accurate picture. *Constraints*, 6(4):329–344, 2001.

[AMZ07] Fabrizio Altarelli, Rémi Monasson, and Francesco Zamponi. Relationship between clustering and algorithmic phase transitions in the random k-XORSAT model and its NP-complete extensions. In *Proceedings of the International Workshop on Statistical-Mechanical Informatics*, 2007. arXiv:cs.CC:0709.0367v2 [cs.CC].

[AP04] Dimitris Achlioptas and Yuval Peres. The threshold for random k-SAT is $2^k \log 2 - \mathrm{O}(k)$. *Journal of the American Mathematical Society*, 17(4):947–973, 2004.

[Apo74] Tom M. Apostol. *Mathematical Analysis: A Modern Approach to Advanced Calculus*. Addison-Wesley, 1957 and 1974.

[APT79] Bengt Aspvall, Michael F. Plass, and Robert Endre Tarjan. A linear-time algorithm for testing the truth of certain quantified boolean formulas. *Information Processing Letters*, 8(3):121–123, 1979.

[ART06] Dimitris Achlioptas and Federico Ricci-Tersenghi. On the solution-space geometry of random constraint satisfaction problems. In *Proceedings of the 38th Annual ACM Symposium on Theory of Computing*, pages 130–139, 2006.

[AS00] Dimitris Achlioptas and Gregory B. Sorkin. Optimal myopic algorithms for random 3-sat. In *Proceedings of the 41th Annual IEEE Symposium on Foundations of Computer Science*, pages 590–600, 2000.

[Azu67] Kazuoki Azuma. Weighted sums of certain dependent random variables. *Tohoku Mathematical Journal*, 19:357–367, 1967.

[BBC$^+$01] Béla Bollobás, Christian Borgs, Jennifer T. Chayes, Jeong Han Kim, and David B. Wilson. The scaling window of the 2-SAT transition. *Random Structures and Algorithms*, 18(3):201–256, 2001.

[BBK72] A. Békéssy, P. Békéssy, and J. Komlós. Asymptotic enumeration of regular matrices. *Studia Scientiarium Mathematicarum Hungarica*, 7:343–353, 1972.

[BC78] Edward A. Bender and E. Rodney Canfield. The asymptotic number of labeled graphs with given degree sequences. *Journal of Combinatorial Theory, Series A*, 24(3):296–307, 1978.

[BEGJ00] Maria Luisa Bonet, Juan Luis Esteban, Nicola Galesi, and Jan Johanssen. On the relative complexity of resolution refinements and cutting planes proof systems. *SIAM Journal on Computing*, 30(5):1462–1484, 2000.

[BEGJ01] Maria Luisa Bonet, Juan Luis Esteban, Nicola Galesi, and Jan Johannsen. On the relative complexity of resolution refinements and cutting planes proof systems. *SIAM Journal on Computing*, 30(5):1462–1484, 2001.

[BFU93] Andrei Z. Broder, Alan M. Frieze, and Eli Upfal. On the satisfiability and maximum satisfiability of random 3-CNF formulas. In *Proceedings of the Fourth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 322–330, 1993.

[BIP05] Stefan Boettcher, Gabriel Istrate, and Allon G. Percus. Spines of random constraint satisfaction problems: Definition and connection with computational complexity. *Annals of Mathematics and Artificial Intelligence*, 44(4):353–372, 2005.

[BKS03] Paul Beame, Henry Kautz, and Ashish Sabharwal. Understanding the power of clause learning. In *International Joint Conference of Artificial Intelligence (IJCAI'03)*, pages 1194–1201, August 2003.

[BKW97] Johannes Blömer, Richard Karp, and Emo Welzl. The rank of sparce random matrices over finite fields. *Random Structures and Algorithms*, 10:407–419, 1997.

[BMW00] Giulio Biroli, Rémi Monasson, and Martin Weigt. A variational description of the ground state structure in random unsatisfiability problems. *The European Physical Journal B*, 14:551–568, 2000.

[BMZ05] A. Braunstein, M. Mézard, and R. Zecchina. Survey propagation: An algorithm for satisfiability. *Random Structures and Algorithms*, 27(2):201–226, 2005.

[BO78] C. Bender and S. Orsag. *Advanced Mathematical Methods for Scientists and Engineers*. McGraw-Hill, 1978.

[Bol79] B. Bollobás. *Graph Theory: An Introductory Course*. Springer-Verlag, 1979.

[Bol80] B. Bollobás. A probabilistic proof of an asymptotic formula for the number of labelled regular graphs. *European Journal of Combinatorics*, 1:311–316, 1980.

[BPZ78] R. A. Bailey, D. A. Preece, and P. J. Zemroch. Totally symmetric latin squares and cubes. *Utilitas Mathematical*, 14:161–170, 1978.

[Bru44] Richard H. Bruck. Some results in the theory of quasigroups. *Transactions of the American Mathematical Society*, 55:19–52, 1944.

[BS04] Daniel Le Berre and Laurent Simon. Fifty-five solvers in vancouver: The SAT 2004 competition. In *Proceedings of the Seventh International Conference on Theory and Applications of Satisfiability Testing*, pages 231–344, 2004.

[BSIW03] Eli Ben-Sasson, Russell Impagliazzo, and Avi Wigderson. Near optimal separation of tree-like and general resolution. *Combinatorica*, 24(4):585–603, 2003.

[BSW01] Eli Ben-Sasson and Avi Wigderson. Short proofs are narrow - resolution made simple. *Journal of the ACM*, 48(2):149–169, 2001.

[CA96] James M. Crawford and Larry D. Auton. Experimental results on the crossover point in random 3-SAT. *Artificial Intelligence*, 81(1–2):31–57, 1996.

[Cal96] Neil J. Calkin. Dependent sets of constant weight vectors in GF(q). *Random Structures and Algorithms*, 9(1–2):49–53, 1996.

[CD03] Nadia Creignou and Hervé Daudé. Generalized satisfiability problems: Minimal elements and phase transitions. *Theoretical Computer Science*, 302(1–3):417–430, June 2003.

[CF86] Min-Te Chao and John Franco. Probabilistic analysis of two heuristics for the 3-satisfiability problem. *SIAM Journal of Computing*, 15(4):1106–1118, 1986.

[CF90] Min-Te Chao and John Franco. Probabilistic analysis of a generalization of the unit clause literal selection heuristic for the k-satisfiability problem. *Information Science*, 51(3):289–314, 1990.

[CFMR96] Colin Cooper, Alan Frieze, Michael Molloy, and Bruce Reed. Perfect matchings in random r-regular, s-uniform hypergraphs. *Combinatorics, Probability, and Computing*, 5:1–14, 1996.

[CKS01] Nadia Creignou, Sanjeev Khanna, and Madhu Sudan. *Complexity Classifications of Boolean Constraint Satisfaction Problems*. SIAM Monographs on Discrete Mathematics and Applications. Society for Industrial and Applied Mathematics, 2001.

[CKT91] Peter Cheeseman, Bob Kanefsky, and William M. Taylor. Where the REALLY hard problems are. In *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence*, pages 331–337, 1991.

[CM04] Harold Connamacher and Michael Molloy. The exact satisfiability threshold for a potentially intractable random constraint satisfaction problem. In *Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science*, pages 590–599, 2004.

[CMMS03] S. Cocco, R. Monasson, A. Montanari, and G. Semerjian. Approximate analysis of search algorithms with "physical" methods. arXiv:cs/0302003v1 [cs.CC], 2003.

[Con04] Harold Connamacher. A constraint satisfaction problem that seems hard for DPLL. In *Proceedings of the Seventh International Conference on Theory and Applications of Satisfiability Testing*, pages 3–11, 2004.

[Coo00] C. Cooper. On the rank of random matrices. *Random Structures and Algorithms*, 16(2):209–232, 2000.

[Coo04] Colin Cooper. The cores of random hypergraphs with a given degree sequence. *Random Structures and Algorithms*, 25(4):353–375, 2004.

[CR92] V. Chvátal and B. Reed. Mick gets some (the odds are on his side). In *Proceedings of the 33rd Annual IEEE Symposium on Foundations of Computer Science*, pages 620–627. IEEE, 1992.

[CS88] Vaček Chvátal and Endre Szemerédi. Many hard examples for resolution. *Journal of the ACM*, 35(4):759–768, 1988.

[CW06] Julie Cain and Nicholas Wormald. Encores on cores. *The Electronic Journal of Combinatorics*, 13(1), 2006. R81.

[dB70] N. G. de Bruijn. *Asymptotic Methods in Analysis.* North-Holland, 1958 and 1970. Dover reprint, 1981.

[DB97] O. Dubois and Y. Boufkhad. A general upper bound for the satisfiability threshold of random r-SAT formulae. *Journal of Algorithms*, 24(2):395–420, 1997.

[DBM03] Olivier Dubois, Yacine Boufkhad, and Jacques Mandler. Typical random 3-SAT formulae and the satisfiability threshold. Technical Report TR03-007, Electronic Colloquium on Computational Complexity, 2003.

[DFM03] Martin Dyer, Alan Frieze, and Michael Molloy. A probabilistic analysis of randomly generated binary constraint satisfaction problems. *Theoretical Computer Science*, 290(3):1815–1828, January 2003.

[DK74] J. Dénes and A. D. Keedwell. *Latin Squares and their Applications*. Academic Press, 1974.

[DLL62] Martin Davis, George Logemann, and Donald Loveland. A machine program for theorem-proving. *Communications of the ACM*, 5(7):394–397, 1962.

[DLR77] Arthur Dempster, Nan Laird, and Donald Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, 39(1):1–39, 1977.

[DM02] Olivier Dubois and Jacques Mandler. The 3-XORSAT threshold. In *Proceedings of the 43rd Annual IEEE Symposium on Foundations of Computer Science*, pages 769–778, 2002.

[DMMZ05] Hervé Daudé, Marc Mézard, Thierry Mora, and Riccardo Zecchina. Pairs of SAT assignment in random boolean formulae. arXiv:cond-mat/0506053v3 [cond-mat.dis-nn], 2005.

[DN] R. Darling and J. R. Norris. Cores and cycles in random hypergraphs. In preparation.

[DP60] Martin Davis and Hilary Putnam. A computing procedure for quantification theory. *Journal of the ACM*, 7(3):201–215, 1960.

[ER59] Paul Erdös and Alfred Rényi. On random graphs, I. *Publicationes Mathematicae Debrecen*, 6:290–297, 1959. Reprinted in *Paul Erd os: The Art of Counting*. Joel Spencer, editor, The MIT Press, 1973.

[ER60] Paul Erdös and Alfred Rényi. On the evolution of random graphs. *Publications of the Mathematical Institute of the Hungarian Academy of Sciences*, 5:17–61, 1960. Reprinted in *Paul Erd os: The Art of Counting*. Joel Spencer, editor, The MIT Press, 1973.

[FdlV92] W. Fernandez de la Vega. On random 2-SAT. Manuscript, 1992.

[Fla03] Abraham D. Flaxman. A sharp threshold for a random constraint satisfaction problem. *Discrete Mathematics*, 285(1–3):301–305, August 2003.

[FLRTZ01] Silvio Franz, Michele Leone, Federico Ricci-Tersenghi, and Riccardo Zecchina. Exact solutions for diluted spin glasses and optimization problems. *Physical Review Letters*, 87(12):127209, 2001.

[FM03] Alan M. Frieze and Michael Molloy. The satisfiability threshold for randomly generated binary constraint satisfaction problems. In *Proceedings of the 7th International Workshop on Randomization and Approximation Techniques in Computer Science*, 2003.

[FP83] John Franco and Marvin Paull. Probabilistic analysis of the davis putnam procedure for solving the satisfiability problem. *Discrete Applied Mathematics*, 5(1):77–87, 1983.

[FR03] Daniel Fernholz and Vijaya Ramachandran. The giant k-core of a random graph with a specified degree sequence. Manuscript, 2003.

[FR04] Daniel Fernholz and Vijaya Ramachandran. Cores and connectivity in sparse random graphs. Technical Report TR04-13, University of Texas Computer Science, 2004.

[Fri99] Ehud Friedgut. Sharp thresholds of graph properties, and the k-SAT problem. *Journal of the American Mathematical Society*, 12(4):1017–1054, 1999.

[FS96] Alan Frieze and Stephen Suen. Analysis of two simple heuristics on a random instance of k-SAT. *Journal of Algorithms*, 20(2):312–355, 1996.

[FW01] Alan Frieze and Nicholas C. Wormald. Random k-SAT: A tight threshold for moderately growing k. In *Proceedings of the Fifth International Symposium on the Theory and Applications of Satisfiability Testing (SAT 2002)*, pages 1–6, 2001.

[Gal77] Zvi Galil. On the complexity of regular resolution and the Davis-Putnam procedure. *Theoretical Computer Science*, 4(1):23–46, 1977.

[GFSB04] Carla P. Gomes, Cèsar Fernández, Bart Selman, and Christian Bessière. Statistical regimes across constrainedness regions. In *Proceedings of the Tenth International Conference on Principles and Practice of Constraint Programming*, 2004.

[GJ79] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman and Company, 1979.

[GM03] Andreas Goerdt and Mike Molloy. Analysis of edge deletion processes on faulty random regular graphs. *Theoretical Computer Science*, 297(1–3):241–260, 2003.

[GMP$^+$01] Ian P. Gent, Ewan Macintyre, Patrick Prosser, Barbara M. Smith, and Toby Walsh. Random constraint satisfaction: Flaws and structure. *Constraints*, 6(4):345–372, 2001.

[Goe96] Andreas Goerdt. A threshold for unsatisfiability. *Journal of Computer and System Sciences*, 53(3):469–486, 1996.

[Hen94] F. Hennecart. Stirling distributions and stirling numbers of the second kind. computational problems and statistics. *Kybernetika*, 30(3):279–288, 1994.

[Her06] P. R. Herwig. Decomposing satisfiability problems. Master's thesis, Delft University of Technology, 2006.

[HM06] Eric I. Hsu and Sheila A. McIlraith. Characterizing propagation methods for boolean satisfiability. In *Proceedings of the Ninth International Conference on Theory and Applications of Satisfiability Testing*, pages 325–338, 2006.

[HS03] Mohammad Taghi Hajiaghayi and Gregory B. Sorkin. The satisfiability threshold of random 3-SAT is at least 3.52. arXiv:math/0310193v2 [math.CO], 2003.

[HvM06] Marijn J. H. Heule and Hans van Maaren. March_dl: Adding adaptive heuristics and a new branching strategy. *Journal on Satisfiability, Boolean Modeling, and Computation*, 2:47–59, 2006.

[HvM07] Marijn Heule and Hans van Maaren. Effective incorporation of double look-ahead procedures. In *Proceedings of the Tenth International Conference on Theory and Applications of Satisfiability Testing*, pages 258–271, 2007.

[Int04] Yannet Interian. Approximation algorithm for random MAX-kSAT. In *Proceedings of the Seventh International Conference on Theory and Applications of Satisfiability Testing*, pages 64–68, 2004.

[JKŁP93] Svante Janson, Donald E. Knuth, Tomasz Łuczak, and Boris Pittel. The birth of the giant component. *Random Structures and Algorithms*, 4(3):231–358, 1993.

[JŁ06] Svante Janson and Malwina J. Łuczak. A simple solution to the k-core problem. *Random Structures and Algorithms*, 30(1–2):50–62, 2006.

[JŁR00] Svante Janson, Tomasz Łuczak, and Andrzej Ruciński. *Random Graphs.* Wiley-Interscience Series in Discrete Mathematics and Optimization. John Wiley & Sons, 2000.

[JMS04] Haixia Jia, Cris Moore, and Bart Selman. From spin glasses to hard satisfiable formulas. In *Proceedings of the Seventh International Conference on Theory and Applications of Satisfiability Testing*, pages 12–19, 2004.

[JSV00] Svante Janson, Yannis C. Stamatiou, and Malvina Vamvakari. Bounding the unsatisfiability threshold of random 3-sat. *Random Structures and Algorithms*, 17(2):103–116, 2000. Erratum in *Random Structures and Algorithms*, 18(1): 99–100, 2001.

[Kim06] Jeong Han Kim. The poisson cloning model for random graphs. research.microsoft.com/theory/jehkim/papers/PCMRG.pdf, 2006.

[KK01] Jeff Kahn and János Komlós. Singularity probabilities for random matrices over finite fields. *Combinatorics, Probability and Computing*, 10(2):137–157, 2001.

[KKKS98] Lefteris M. Kirousis, Evangelos Kranakis, Danny Krizanc, and Yannis C. Stamatiou. Approximating the unsatisfiability threshold of random formulas. *Random Structures and Algorithms*, 12:253–269, 1998.

[KKL02] Alexis C. Kaporis, Lefteris M. Kirousis, and Efthimios G. Lalas. The probabilistic analysis of a greedy satisfiability algorithm. In *Proceedings of the 10th Annual European Symposium on Algorithms*, pages 574–585. Springer-Verlag, 2002.

[KKL03] Alexis C. Kaporis, Lefteris M. Kirousis, and Efthimios G. Lalas. Selecting complementary pairs of literals. In *Electronic Notes in Discrete Mathematics*, volume 16. Elsevier, 2003.

[KKS$^+$01] Alexis C. Kaporis, Lefteris M. Kirousis, Yannis C. Stamatious, Malvina Vamvakari, and Michele Zito. Coupon collectors, q-binomial coefficients and the unsatisfiability threshold. In *Theoretical Computer Science, 7th Italian Conference, ICTCS 2001*, Lecture Notes in Computer Science, pages 328–338. Springer, 2001.

[KŁ02] Michal Karoński and Tomasz Łuczak. The phase transition in a random hypergraph. *Journal of Computational and Applied Mathematics*, 142(1):125–135, 2002.

[KMPS95] Anil Kamath, Rajeev Motwani, Krishna Palem, and Paul Spirakis. Tail bounds for occupancy and the satisfiability threshold conjecture. *Random Structures and Algorithms*, 7:59–80, 1995.

[KS94] Scott Kirkpatrick and Bart Selman. Critical behavior in the satisfiability of random boolean expressions. *Science*, 264(5163):1297–1232, 1994.

[LSB05] Matthew D. T. Lewis, Tobias Schubert, and Bernd W. Becker. Speedup techniques utilizes in modern SAT solvers: An analysis in the MIRA environment. In *Proceedings of the Eighth International Conference on Theory and Applications of Satisfiability Testing*, pages 437–443, 2005.

[Łuc90] Tomasz Łuczak. On the equivalence of two basic models of random graphs. In Michal Karonski, Jerzy Jaworski, and Andrzej Rucinski, editors, *Proceedings of Random Graphs '87*, pages 151–158. John Wiley & Sons, 1990.

[Łuc91] Tomasz Łuczak. Size and connectivity of the k-core of a random graph. *Discrete Mathematics*, 91(1):61–68, 1991.

[MdlV95] A. El Maftouhi and Wenceslas Fernandez de la Vega. On random 3-SAT. *Combinatorics, Probability & Computing*, 4:189–195, 1995.

[Meh91] Madan Lal Mehta. *Random Matrices.* Academic Press, 2nd edition, 1991.

[Min02] Mark Minichiello. Solving MAX-XOR-SAT problems with stochastic local search. Masters in Mathematics 3rd Year Project Report, 2002.

[Mit02] David G. Mitchell. Resolution complexity of random constraints. In *Principles and Practices of Constraint Programming – CP 2002*, pages 295–309, 2002.

[MM06] Thierry Mora and Marc Mézard. Geometric organization of solutions to random linear boolean equations. arXiv:cond-mat/0609099v1 [cond-mat.dis-nn], 2006.

[MMW07] Eliza Maneva, Elchanan Mossel, and Martin J. Wainwright. A new look at survey propagation and its generalizations. *Journal of the ACM*, 54(4), 2007.

[MMZ01] Olivier C. Martin, Rémi Monasson, and Riccardo Zecchina. Statistical mechanics methods and phase transitions in optimization problems. *Theoretical Computer Science*, 265(1–2):3–67, 2001.

[MMZ05] Marc Mézard, Thierry Mora, and Riccardo Zecchina. Clustering of solutions in the random satisfiability problem. *Physical Review Letters*, 94(19):197205, 2005.

[MMZ06] Stephan Mertens, Marc M'ezard, and Riccardo Zecchina. Threshold values of random k-SAT from the cavity method. *Ramdom Structures and Algorithms*, 28(3):340–373, 2006.

[Mol] Michael Molloy. When does the giant component bring unsatisfiability? To appear in *Combinatorica*.

[Mol01] Michael Molloy. Thresholds for colourability and satisfiability for random graphs and boolean formulae. In J. Hirschfield, editor, *Surveys in Combinatorics*, pages 165–197. Cambridge University Press, 2001.

[Mol02] Michael Molloy. Models and threshold for random constraint satisfaction problems. In *Proceedings of the Thirty-Fourth Annual ACM Symposium of Theory of Computing*, pages 209–217, 2002.

[Mol05] Michael Molloy. Cores in random hypergraphs and boolean formulas. *Random Structures and Algorithms*, 27(1):124–135, 2005.

[MPV86] Marc Mézard, Giorgio Parisi, and Miguel Angel Virasoro. SK model: The replica solution without replicas. *Europhysics Letters*, 1:77, 1986.

[MPV87] Marc Mézard, Giorgio Parisi, and Miguel Angel Virasoro. *Spin Glass Theory and Beyond*, volume 9 of *World Scientific Lecture Notes in Physics*. World Scientific, 1987.

[MPZ02] M. Mézard, G. Parisi, and R. Zecchina. Analytic and algorithmic solution of random satisfiability problems. *Science*, 297:812–815, 2002.

[MRTZ03] M. Mézard, F. Ricci-Tersenghi, and R. Zecchina. Two solutions to diluted p-spin models and XORSAT problems. *Journal of Statistical Physics*, 111(3–4):505–533, 2003.

[MS07] Michael Molloy and Mohammad Salavatipour. The resolution complexity of random constraint satisfaction problems. *SIAM Journal of Computing*, 37(3):895–922, 2007.

[MWHC96] B. S. Majewski, N. C. Wormald, G. Havas, and Z. J. Czech. A family of perfect hashing methods. *The Computer Journal*, 39(6):547–554, 1996.

[MZ96] Rémi Monasson and Riccardo Zecchina. The entropy of the K-satisfiability problem. *Physical Review Letters*, 76(21):3881–3885, 1996.

[MZ97] Rémi Monasson and Riccardo Zecchina. Statistical mechanics of the random K-satisfiability problem. *Physical Review E*, 56(2):1357–1370, 1997.

[MZ02] Marc Mézard and Riccardo Zecchina. Random K-satisfiability problem: From an analytic solution to an efficient algorithm. *Physical Review E*, 66(5), 2002. 056126.

[MZK$^+$96] Rémi Monasson, Riccardo Zecchina, Scott Kirkpatrick, Bart Selman, and Lidor Troyansky. Phase transitions and search cost in the 2+p–SAT problem. In *4th Workshop on Physics and Computation*, 1996.

[MZK$^+$99] Rémi Monasson, Riccardo Zecchina, Scott Kirkpatrick, Bart Selman, and Lidor Troyansky. $2 + p$-SAT: Relation of typical-case complexity to the nature of the phase transition. *Random Structures and Algorithms*, 15(3–4):414–435, October–December 1999.

[NDSLB04] Eugene Nidelman, Alex Devkar, Yoav Shoham, and Kevin Leyton-Brown. Understanding random SAT: Beyond the clauses-to-variables ratio. In *Pro-*

ceedings of the Tenth International Conference on Principles and Practice of Constraint Programming, 2004.

[Par02] Andrew J. Parkes. Scaling properties of pure random walk on random 3-SAT. In *CP '02: Proceedings of the 8th International Conference on Principles and Practice of Constraint Programming*, pages 708–713. Springer-Verlag, 2002.

[Pea82] Judea Pearl. Reverend Bayes on inference engines: A distributed hierarchical approach. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 133–136, 1982.

[PSW96] Boris Pittel, Joel Spencer, and Nicholas Wormald. Sudden emergence of a giant k-core in a random graph. *Journal of Combinatorial Theory, Series B*, 67(1):111–151, 1996.

[Rio07] Oliver Riordan. The k-core and branching processes. arXiv:math/0511093v2 [math.CO], 2007.

[Sch78] Thomas J. Schaefer. The complexity of satisfiability problems. In *Proceedings of the Tenth Annual ACM Symposium on Theory of Computing*, pages 216–226, 1978.

[SM03] Guilhem Semerjian and Rémi Monasson. Relaxation and metastability in a local search procedure for the random satisfiability problem. *Physical Review E*, 67:066103, 2003.

[Smi01] Barbara M. Smith. Constructing an asymptotic phase transition in random binary constraint satisfaction problems. *Theoretical Computer Science*, 265(1–2):265–283, August 2001.

[SML96] Bart Selman, David G. Mitchell, and Hector J. Levesque. Generating hard satisfiability problems. *Artificial Intelligence*, 81(1–2):17–29, 1996.

[SS85] Jeanette P. Schmidt and Eli Shamir. Component structure in the evolution of random hypergraphs. *Combinatorica*, 5(1):81–94, 1985.

[Tal01] Michel Talagrand. Rigorous results for mean field models for spin glasses. *Theoretical Computer Science*, 265(1–2):69–77, 2001.

[Tal03a] Michel Talagrand. On the meaning of Parisi's functional order parameter. *Comptes Rendus Mathematique*, 337(9):625–628, 2003.

[Tal03b] Michel Talagrand. Self organization in the low temperature region of a spin glass model. *Reviews in Mathematical Physics*, 15(1):1–78, 2003.

[Wor78] Nicholas C. Wormald. *Some Problems in the Enumeration of Labelled Graphs*. PhD thesis, University of Newcastle, 1978.

[Wor81a] Nicholas C. Wormald. The asymptotic connectivity of labelled regular graphs. *Journal of Combinatorial Theory, Series B*, 31:156–167, 1981.

[Wor81b] Nicholas C. Wormald. The asymptotic distribution of short cycles in random regular graphs. *Journal of Combinatorial Theory, Series B*, 31:168–182, 1981.

[Wor95] Nicholas C. Wormald. Differential equations for random processes and random graphs. *Annals of Applied Probability*, 5(4):1217–1235, 1995.

[XL00] Ke Xu and Wei Li. Exact phase transitions in random constraint satisfaction problems. *Journal of Artificial Intelligence Research*, 12:93–103, 2000.

[XL06] Ke Xu and Wei Li. Many hard examples in exact phase transitions. *Theoretical Computer Science*, 355(3):291–302, 2006.