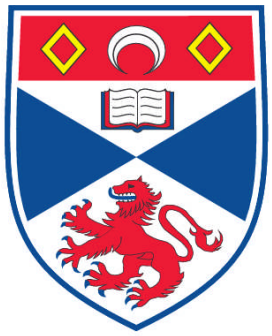
NUMERICAL EVIDENCE FOR PHASE TRANSITIONS OF NP-COMPLETE PROBLEMS FOR INSTANCES DRAWN FROM LÉVY-STABLE DISTRIBUTIONS

Abram Connelly

A Thesis Submitted for the Degree of PhD
at the
University of St. Andrews



2011

Numerical Evidence for Phase Transitions of NP-Complete Problems for Instances Drawn from Lévy-Stable Distributions

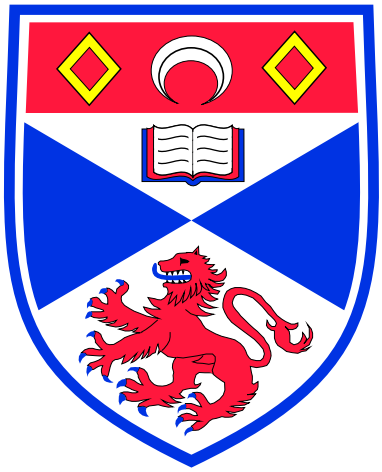
Thesis by

Abram Connelly

In Partial Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy



University of St Andrews

School of Computer Science

2011

Abstract

Random NP-Complete problems have come under study as an important tool used in the analysis of optimization algorithms and help in our understanding of how to properly address issues of computational intractability.

In this thesis, the Number Partition Problem and the Hamiltonian Cycle Problem are taken as representative NP-Complete classes. Numerical evidence is presented for a phase transition in the probability of solution when a modified Lévy-Stable distribution is used in instance creation for each. Numerical evidence is presented that show hard random instances exist near the critical threshold for the Hamiltonian Cycle problem. A choice of order parameter for the Number Partition Problem's phase transition is also given.

Finding Hamiltonian Cycles in Erdös-Rényi random graphs is well known to have almost sure polynomial time algorithms, even near the critical threshold. To the author's knowledge, the graph ensemble presented is the first candidate, without specific graph structure built in, to generate graphs whose Hamiltonicity is intrinsically hard to determine. Random graphs are chosen via their degree sequence generated from a discretized form of Lévy-Stable distributions. Graphs chosen from this distribution still show a phase transition and appear to have a pickup in search cost for the algorithms considered. Search cost is highly dependent on the particular algorithm used and the graph ensemble is presented only as a potential graph ensemble to generate intrinsically hard graphs that are difficult to test for Hamiltonicity.

Number Partition Problem instances are created by choosing each element in the list from a modified Lévy-Stable distribution. The Number Partition Problem has no known good approximation algorithms and so only numerical evidence to show the phase transition is provided without considerable focus on pickup in search cost for the solvers used. The failure of current approximation algorithms and potential candidate approximation algorithms are discussed.

Declaration

I, Abram Connelly, hereby certify that this thesis, which is approximately 43387 words in length, has been written by me, that it is the record of work carried out by me and that it has not been submitted in any previous application for a higher degree.

Date ..... Signature of candidate .....

I was admitted as a research student in September 2005 and as a candidate for the degree of Doctor of Philosophy in June 2011; the higher study for which this is a record was carried out in the University of St Andrews between 2005 and 2011.

Date ..... Signature of candidate .....

I hereby certify that the candidate has fulfilled the conditions of the Resolution and Regulations appropriate for the degree of Doctor of Philosophy in the University of St Andrews and that the candidate is qualified to submit this thesis in application for that degree.

Date ..... Signature of supervisor .....

In submitting this thesis to the University of St Andrews I understand that I am giving permission for it to be made available for use in accordance with the regulations of the University Library for the time being in force, subject to any copyright vested in the work not being affected thereby. I also understand that the title and the abstract will be published, and that a copy of the work may be made and supplied to any bona fide library or research worker, that my thesis will be electronically accessible for personal or research use unless exempt by award of an embargo as requested below,

and that the library has the right to migrate my thesis into new electronic forms as required to ensure continued access to the thesis. I have obtained any third-party copyright permissions that may be required in order to allow such access and migration, or have requested the appropriate embargo below.

The following is an agreed request by candidate and supervisor regarding the electronic publication of this thesis: Access to printed copy and electronic publication of thesis through the University of St Andrews.

Date ..... Signature of candidate .....

Date ..... Signature of supervisor .....

Copyright

Acknowledgments

I would like to thank supervisor, Ian Gent, for his support.

I would like to thank Josh Barratt, Rama Hoetzlein, Chris Hooley, Ron Maimon, Gary Sinclair, Nick Taylor and Andy Wood for many helpful discussions.

Finally, I would like to thank my family, Mom, Dad, Aaron, Karen and Ben for their support and love.

Contents

List of Figures

Chapter 1

Introduction

Non-Deterministic Polynomial Time complete problems, or NP-Complete problems, encapsulate our notion of a particular kind of computation. Because of their generality and their reducibility they encompass many commonly encountered problems, from perfectly balancing a set of weights on a scale to determining if there is a complete tour through a graph network. The reducibility provides a facility so that a general solution for one translates to a solution to all. The converse is also true and a difficulty in one translates to a difficulty in all.

A problem belongs to the class of NP if, given a solution, one can verify the validity in polynomial time. This gives us an efficient certificate of authenticity: we only ask for problems whose solution we can verify efficiently. We call a class of problems NP-Complete if a general solution for this class of problem would solve all problems in NP. Finding a complete cycle in a graph network (the Hamiltonian Cycle Problem) and perfectly balancing weights on a scale (the Number Partition Problem) are both examples of NP-Complete problems that, given a general solution to either would imply a solution to every NP-Complete problem. The NP class is one of many residing on the complexity hierarchy and the reader is referred to [52], [90] and [72] for more details.

The famous P vs. NP conjecture asks whether NP-Complete problems have an efficient algorithm to solve them ([35], [48]) where the commonly held belief is that they are distinct ([1]). One avenue of investigation into the nature of the difference between these two classes is to look at random instance generation. Given a class of NP-Complete problems, look at an instance drawn from some distribution and analyze run-times of algorithms within that purview. While this might not provide a proof as to why the two classes are different, it might provide insight as to how they are different.

This analysis of algorithms provides the motivation for looking at random NP-Complete problems.

In the past two decades, so-called phase transitions of random instances of NP-Complete problems have come under consideration ([51], [29], [66]). In this context, the phase transition is a rapid change in the existence of a solution when instances are created from a parametrized random ensemble as the parameter is varied. The idea is that the parameter encapsulates a microscopic quantity of the ensemble that, when varied, creates a macroscopic change in the system. In this case the macroscopic property is the probability of a solution existing. For example, there is a phase transition in the probability of finding a Hamiltonian Cycle in random graphs, or Erdös-Rényi random graphs, when the probability of edge creation is varied from 0 to 1 ([69], [93]). For the Number Partition Problem the parameter is the ratio of number of elements in the list to the average bit size of each element ([53], [23]). The critical parameter depends on the NP-Complete class under investigation and it is not always clear what the proper parametrization is ([77], [86]).

The phase transition is a rapid change in probability from a region where the probability of a solution is almost surely 0 to a region where the probability of a solution is almost surely 1. The exact nature of the transition between these two phases will be discussed later in Chapter 3. If a problem instance is created in the region where the probability of a solution occurring is almost surely 1, we might guess that the ability to find this solution would be easy. For many NP-Complete problems, this is the case ([86], [29], [105]). From the other end, we might guess that the ability of determining that no solution can exist in the region of almost sure 0 probability is also easily determined. For example, this is the case for the Hamiltonian Cycle Problem on Erdös-Rényi random graphs past the critical threshold ([18], [105], [19]). Given this anecdotal evidence, we might guess that all NP-Complete problem instances are easy when they are drawn in the almost sure probability 1 or 0 region and that hard problems exist at or near the transition point. This was the view popularized by [29], [60], and [61].

Since then, the issue has become more complex. The Hamiltonian Cycle Problem on Erdös-Rényi random graphs, which had been reported in [29] to have an exponential increase in search cost near the critical threshold, turned out to have provably polynomial time algorithms ([19], [105]) and the exponential increase in search cost noticed was due only to a poor search algorithm choice. For 3-SAT, progress has been made with algorithms that find solutions very near the critical threshold, even in very large systems ([25]).

Some NP-Complete problems appear to remain difficult even well in the almost sure probability

1 or 0 region. Current state of the art algorithms for the Number Partition Problem do not fare much better than random chance nearly everywhere in the probability space for problems of even intermediate size ([53], [21], [22]).

In one extreme there is the Hamiltonian Cycle Problem which, in general, is thought to be exponentially difficult but for which we have difficulty finding random instances whose search cost is worse than polynomial. In the other extreme, there is the Number Partition Problem where, even into the region where there are almost surely an exponential number of solutions, our best algorithms have difficulty finding even one.

This thesis focuses on addressing how random graphs can be created so as to create intrinsically difficult instances for the Hamiltonian Cycle Problem. Approximation algorithms are discussed for the Number Partition Problem and while the author's investigation for better search methods has met with failure, a potential class of approximation algorithms is proposed that could prove fruitful. The Hamiltonian Cycle Problem and the Number Partition Problem are taken as representative NP-Complete classes with the hope that statements about either will translate to all problems within this complexity class.

In both cases a different probability distribution, one that has previously been neglected, is used to create random instances. Numerical evidence is presented to show that a phase transition also occurs from these previously unconsidered distributions. Numerical evidence is provided to suggest that graphs whose degree sequence is generated from the family of Lévy-Stable distributions are intrinsically difficult to determine Hamiltonicity. A slightly modified algorithm used in Culberson and Vandegriend's paper [105] for small graphs and a randomized algorithm proposed by Pósa [93] for large graphs are used to generate run time analysis. As run times are highly dependent on the algorithms used, the class of graphs are presented only as a candidate to generate intrinsically hard graphs whose Hamiltonicity is difficult to determine. Finally, the inability of current algorithms to solve the Number Partition Problems in the region where there is almost surely a solution is discussed.

1.1 Lévy-Stable Distributions

Instance generation for the random Hamiltonian Cycle Problem and the random Number Partition Problem rely on the family of Lévy-Stable distributions and so will be discussed before proceeding

further.

The family of Lévy-Stable probability distributions are, under some very general conditions, the convergent distribution of summing independent and identically distributed random variables ([88], [46]). In general, the class of Lévy-Stable distributions do not admit closed form solutions and are most often defined in terms of their characteristic function:

$$\phi(x) = \exp(-|\gamma t|^{\alpha}(1 - i\beta \text{sign}(t) \tan(\pi\alpha/2)) + i\delta t)$$

where sign(t) returns the sign of t, α is the critical exponent, β is the skew, γ is the scale and δ is the shift. The probability density function can be given by the Fourier Transform of its characteristic function:

$$p(x) = \frac{1}{2\pi} \int_{-\infty}^{\infty} dt \ \exp(-itx - |\gamma t|^{\alpha}(1 - i\beta sign(t) tan(\pi\alpha/2)) + i\delta t)$$

Only for a few values of α does the Lévy-Stable distribution reduce to a closed form solution [1]. One can view α as a parameter to set the degree of the power law tail of the distribution. γ is analogous to the variance term for the Normal distribution, though this analogy is not exact ([88]). In Nolan's ([88]) notation, the above corresponds to a distribution chosen from the $S(\alpha, \beta, \gamma, \delta; 0)$ class of Lévy-Stable distributions. For simplicity's sake, only the class of symmetric Lévy-Stable distributions is considered i.e. $\beta = 0$, $\delta = 0$:

$$p(x) = \frac{1}{2\pi} \int_{-\infty}^{\infty} dt \exp(-itx - \gamma^{\alpha}|t|^{\alpha})$$

A salient feature is their power law tails. From Nolan ([88]), for $x \gg 0$:

$$P(X > x) \sim \gamma^{\alpha} c_{\alpha}(1 + \beta) x^{-\alpha}$$

$$p(x) \sim \alpha\gamma^{\alpha} c_{\alpha}(1 + \beta) x^{-(\alpha+1)}$$

Here, X is the Lévy-Stable random variable drawn from $S(\alpha, \beta, \gamma, \delta; 0)$, $p(\cdot)$ is the probability distribution function, $P(\cdot)$ is the cumulative distribution function and $c_{\alpha} = \sin(\pi\alpha/2)\gamma(\alpha)/\pi$. For this

[1] for example, a Lévy distribution follows if $\alpha = \frac{1}{2}$, a Cauchy distribution for $\alpha = 1$, and a Normal distribution for $\alpha = 2$. See Nolan ([88]) for more details

reason, Lévy-Stable distributions are often called "fat-tailed" distributions, with the α parameter setting the 'corpulence' of the tail of the distribution.

An attractive feature of the Lévy-Stable distributions is their stability: Should the sum of independent and identically distributed (i.i.d.) random variables (r.v.'s) converge, then they converge to one parametrization in the Lévy-Stable family ([88], [46], [108]). If one starts out with i.i.d. r.v.'s drawn from a Lévy-Stable distribution to begin with, then the resulting sum will be Lévy-Stable with the same parametrization save for a rescaling factor. This can be stated as (again, using Nolan's ([88]) notation):

$$X_k \in S(\alpha, \beta, \gamma, \delta; 0), \quad k \in [0 \dots n-1]$$

$$X_0 + X_1 + \cdots + X_{n-1} \sim S(\alpha, \beta, n^{1/\alpha}\gamma, \delta_*; 0)$$

$$\delta_* = \begin{cases} n\delta + \gamma\beta \tan(\pi\alpha/2)(n^{1/\alpha} - n), & \alpha \neq 1 \\ n\delta + 2\gamma\beta n \ln(n)/\pi, & \alpha = 1 \end{cases}$$

As a reminder, the sum of finite varianced i.i.d. r.v.'s converges to a Normal Distribution. Once the finite variance condition is dropped, the Gaussian is no longer the convergent distribution and one of the other in the family of Lévy-Stable distributions is the limiting case for sums of stable random variables. Under some very general conditions, we are assured that for the sum of r.v.'s with differing underlying distributions, one in the class of Lévy-Stable distributions is likely to result ([46]).

When needed, the GNU Scientific Library (GSL) is used as it provides functions for simulating Lévy-Stable random variables.

1.2 The Hamiltonian Cycle Problem

A Hamiltonian Cycle is a graph traversal that passes through each vertex exactly once and ends next to where it started. Given a graph $G = (V, E)$, of vertices V and edges E, $|V| = n$, $|E| = m$, the Hamiltonian Cycle Problem asks if a path, $P = (v_0, v_1, ..., v_{n-1})$, exists such that $v_i \neq v_j$ for $i \neq j$ and with $(v_i, v_{i+1}), (v_{n-1}, v_0) \in E$ for $0 \leq i < n-1$. Figure 1.1 gives an example of a Hamiltonian cycle on a small graph, with bold lines representing the complete circuit through the graph. Unless otherwise stated, all graphs are assumed to be simple graphs: Undirected edges, no self loops and

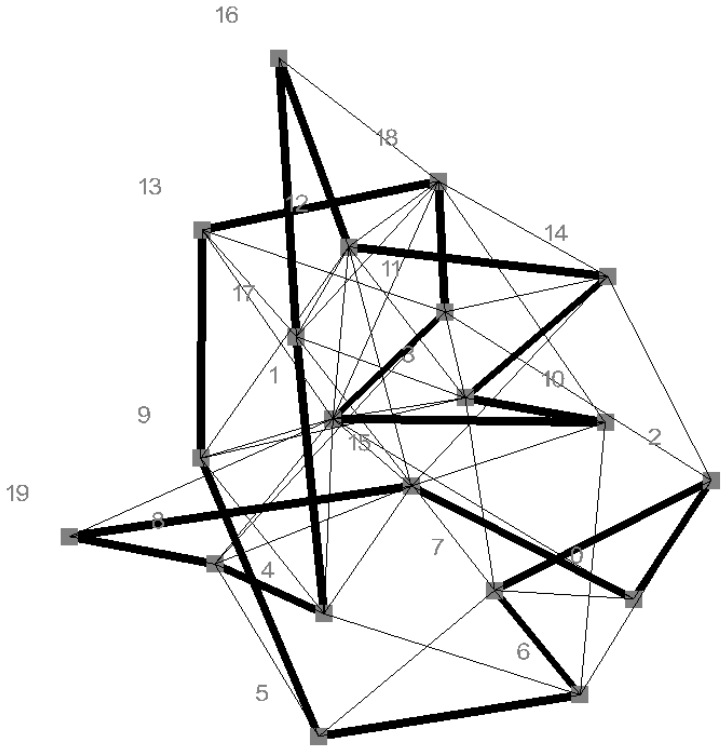


Figure 1.1: An example Hamiltonian Cycle in a small 20 vertex graph. The Hamiltonian Cycle is highlighted in black.

no multiple edges.

What is usually meant by a random graph is a graph constructed by adjoining vertex v to vertex w with a certain probability p, independent of all other connections. Alternatively, one could fix the number of edges and then choose connections independently at random. These two methods are slightly different but for properties of Hamiltonian Cycles, under some lax conditions, give essentially the same results in the sense that the graph ensemble with the number of edges fixed, $G_{N,M}$, can be replaced by the graph ensemble with a fixed edge probability, $G_{N,p}$ with $p = M/\binom{N}{2}$ ([18], Theorem 2.2). In the literature, some results are for graph ensembles chosen from $G_{N,M}$ while others are for graph ensembles chosen from $G_{N,p}$. In what follows, either of the two conventions will be used as is convenient. Hereafter, random graphs chosen in this way will be referred to as Erdös-Rényi random graphs unless otherwise specified.

In this thesis, random graphs are constructed by choosing a degree sequence from modified Lévy-Stable distributions. For simplicity, symmetric Lévy-Stable distributed random variables are chosen

which are then absolute valued and truncated to produce the degree sequence for graph generation. Graph construction and feasibility is an issue and will be discussed further in chapter 2 and 4. Briefly, given each vertex's degree, the "Erased Configuration Model" ([85]) of graph construction is employed by executing a best effort connection scheme to pair vertices until each vertex's degree schedule is filled. The reader is referred to Chapter 3 where the Erased Configuration Model is explained in more detail.

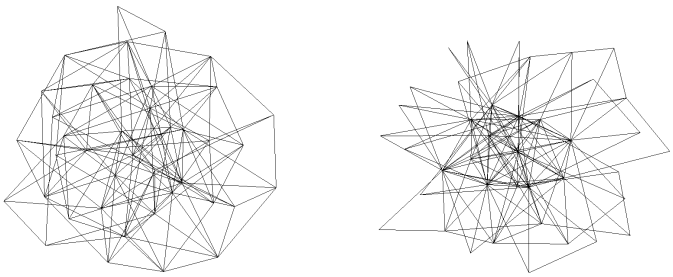


Figure 1.2: An example of an Erdös-Rényi graph with 40 vertices on the left and a graph with 40 vertices whose degree sequence was generated via a truncated Lévy-Stable distribution on the right. The Erdös-Rényi graph on the left was generated with an edge probability of $p = 0.1957$ and has 148 edges. The graph on the right was generated from a truncated symmetric Lévy-Stable distributed degree sequence with parameters $\alpha = 0.5$, $\gamma = 3$, has minimum degree 2, maximum degree 20 and 148 edges. Notice that even though the edge and vertex count is exactly the same, the right most graph has a much different qualitative look to it: there are many more hub-like vertices with lower degree vertices around it creating "spokes". The Erdös-Rényi random graph has a degree sequence which does not vary as wildly and gives it a homogeneous look.

Figure 1.2 gives an example of an Erdös-Rényi graph compared with a power law degree distributed graph. Erdös-Rényi random graphs have their degree sequences that tends towards a Poisson distribution ([18], Chap. 3.1) whereas power law degree distributed graphs have a degree distribution that has a much wider variation. This gives Erdös-Rényi graphs a more homogeneous distribution of degrees for the vertices in the graph, whereas the power law degree distributed graphs tend to be much more inhomogeneous. Given a graph whose degree distribution is chosen from one of the family of Lévy-Stable probability distributions whose critical exponent is in the range of 0 to 2, the second moment diverges as larger graphs are considered. This gives power law degree distributed graphs their qualitative difference to Erdös-Rényi graphs.

1.3 The Number Partition Problem

One formulation of the Number Partition Problem is, given n integers, $a_k \in \mathbb{N}$, $0 \leq k < n$, find a partition of the list into two bins such that $|\sum_{k=0}^{n-1} \sigma_k a_k| \leq 1$, where $\sigma_k \in \{-1, 1\}$. For example, if the list was (81, 435, 87, 96, 17, 483, 278, 388, 363, 56), then a perfect partition of this list would be:

$$-81 - 435 - 87 + 96 + 17 - 483 + 278 + 388 + 363 - 56 = 0$$

The random Number Partition Problem is usually defined as choosing each element in the list from some range uniformly at random. For example, the above numbers were chosen from a range of $[1 \ldots 512]$ uniformly and at random.

In this thesis, random Number Partition Problem instance elements are chosen from a modified Lévy-Stable distribution. As mentioned in the previous section, sums of independent and identically distributed (i.i.d.) Lévy-Stable random variables (r.v.'s) are, up to a rescaling factor, a Lévy-Stable r.v. with the same parametrization. Instance elements are restricted to $\mathbb{N}$. For simplicity, random variables are chosen from a symmetric Lévy-Stable distribution which are then absolute valued and truncated to force integral values for the list elements.

The stability property of the Lévy-Stable distributions allows for easier analysis and will be used in heuristics to estimate the order parameter for the phase transition of the Number Partition Problem for elements chosen from this distribution.

1.4 Random NP-Complete Problems and Phase Transitions

From a physics and statistics perspective, complexity and criticality is a field concerned with how large systems behave when governed by simple rules ([9], [62]). Complexity in this context should not be confused with Computational Complexity but rather as emergent behavior from the repeated application of simple rules. Sometimes these systems will exhibit a so-called phase transition. This is a rapid change from one state to another. The region between these two states, where observables are scale-free, is called the critical region. A system is said to be critical when in this critical region ([30]).

A canonical example of a system displaying complexity and criticality is the percolation model ([99]). Start with a two dimensional square lattice where each cell is initially blank. Fill in each square with some probability p. We restrict our attention to finite lattices and look at the probability of the left wall being connected to the right wall by a contiguous path of filled cells as we increase the size of the lattice. Below the critical point a spanning cluster almost surely does not exist, whereas above the critical point this spanning cluster almost surely does exist. Very near or at the critical point, the system is critical, where cluster sizes do not exhibit a length scale, characterized by a power law distribution in cluster size.

It has been fairly recently noticed that choosing NP-Complete instances at random from a particular parametrized distribution has much the same behavior as these complex and critical systems. Fu and Anderson [51] thought to use statistical mechanical methods in investigating combinatorial optimization problems. Mitchell, Selman and Levesque [84] discovered the phase transition for random k-SAT. Culberson and Vandegriend [105] give a good overview and numerical simulation of the phase transition of the Hamiltonian Cycle Problem in Erdös-Rényi random graphs. There is a large body of work on the phase transition of the appearance of a Hamiltonian Cycle in Erdös-Rényi random graphs and the reader is referred to Bollobás's reference [18]. Gent and Walsh [53] were first to notice the phase transition for the Number Partition Problem which was later refined by Mertens [79] and analytically solved by Borgs, Chayes and Pittel [23]. See Martin, Monasson and Zecchina [77] for a survey on statistical mechanical methods in use for NP-Complete problems.

Much like the probability of finding the spanning cluster in the percolation model, the probability of finding a solution to a random instance of an NP-Complete problem undergoes a rapid transition. There is a region of insolubility, where instances drawn below the critical point almost surely do not have a solution. This then goes to a region of almost sure solubility, where instances drawn above the critical region almost surely have a solution. In between, there is a rapid transition whose form is similar to the percolation transition ([66]).

Many NP-Complete problems, properly formulated, exhibit phase transition behavior with respect to the probability of finding a solution. Often the parametrization is not obvious and still requires insight into how to formulate and analyze the problem properly ([77]).

It was thought that difficult instances resided in the critical region ([29]). For the Hamiltonian cycle problem, where random graphs are Erdös-Rényi, this is provably not true. Bollobás, Fenner and Frieze [19] give an almost sure polynomial time algorithm amongst others (See Angluin and Valiant

[8] and Shamir [96]). See Bollobás's reference [18] for further details and results. To the author's knowledge, graphs whose degree sequences are chosen from a Lévy-Stable distribution are the only candidates for random graph ensembles whose Hamiltonicity is intrinsically hard to determine.

Erdös-Rényi graphs have an average diameter of $\sim \ln N$ whereas power law degree distributed graphs have average diameter of $\sim \ln \ln N$ or smaller ([34], [101], [102], [103]). This comparatively large diameter gives Erdös-Rényi graphs a "tree-like" structure ([81]). This local tree like structure of the Erdös-Rényi random graphs could be one factor that allows for random backtracking to be so successful. Power law degree distributed random graphs are considered as a candidate to find intrinsically hard instances of graphs where Hamiltonicity determination is difficult, as they do not suffer from either the homogeneity of degree distribution or the tree-like structure that Erdös-Rényi graphs exhibit.

1.5 Contributions

Understanding when random instances of NP-Complete problems become hard helps us in our understanding of the P vs NP conjecture, if not in why, then in how. It also allows us to understand how to better cope with NP-Complete problems in general. If, for example, a set of instances of NP-Complete problems were to fall in the region where there is almost surely a solution, very much further after the phase transition occurs, we might expect instances drawn from this region to be easily soluble by some algorithm. This might help us in developing better algorithms suited to problems drawn from a particular region of the distribution that are better able to exploit features found there.

In this thesis, numerical experiments are presented to give evidence for a class of random graphs in which determining Hamiltonicity is potentially intrinsically hard. This is novel in that previously hard graph instances created for the Hamiltonian Cycle Problem were specifically designed to foil particular search algorithms without attempting to create intrinsically hard graphs ([105]). It should be noted that this is only a candidate probability distribution for generating graphs whose Hamiltonicity is difficult to detect. Until there is a theory for why it would be the case that these graphs are intrinsically hard or not, caution should be exercised when making claims of locations of hard instances of the Hamiltonian Cycle problems.

The Erdös-Rényi graph model assumes a fixed probability for each potential edge between the n

vertices. This results in a degree distribution that is Poissonian. The finite variance of the degree distribution makes the degree variation small, creating "tree-like" structures ([81]). This might be one of the reasons why many algorithms using relatively simple heuristics work so well in determining Hamiltonicity. It is a fine balance to create graphs that have enough large degree vertices to destroy the local tree like behavior but have enough low degree vertices to not make path selection trivial.

As an attempt to circumvent the tree-like structure of the Erdös-Rényi random graph model, random graphs are constructed by choosing a degree distribution from a slightly modified Lévy-Stable distribution. The power law tails of the Lévy-Stable distribution give the degree distribution diverging second moments which destroys the tree-like structure that Erdös-Rényi random graphs possess ([101], [102], [103]). The Lévy-Stable distributions are attractive as a choice as, under some general conditions, they are the convergent distribution of sums of independent and identically distributed random variables [2]. This limiting behavior means that we need not worry too much about the underlying processes involved in instance creation as, under very general restrictions, we can be reasonably assured that the limiting behavior is Lévy-Stable.

Erdös-Rényi graphs are locally 'tree-like' ([81]), in that there is a low probability of finding a path that loops back of length less than $\ln(N)$. The diverging second moment introduces "hubs" in graphs whose degree distribution is chosen to be Lévy-Stable which reduces the average diameter of the graph from $\sim \ln(N)$ seen in Erdös-Rényi random graphs to $\sim \ln\ln(N)$ or smaller ([101], [102], [103]). This property might be one of the more important reasons why paths through the graph will loop back in on themselves at an early stage rather than being allowed to continue unfettered. The diameter alone does not characterize a graph as a complete graph will have a low diameter but is trivially Hamiltonian. Instead, it is the low graph diameter compared with the edge density.

Numerical evidence of a phase transition is presented. Numerical evidence is also presented that shows an increase in computational search cost associated with finding a Hamiltonian Cycle near the phase transition. Search cost is highly dependent on the algorithms employed and the numerical evidence is presented only to suggest a potential class of intrinsically hard graphs whose Hamiltonicity is difficult to determine.

A slightly modified version of the algorithm presented by Culberson and Vandegriend [105] is used. The version of Culberson and Vandegriend's program was implemented independently and does not include articulation point checking though adds some other features specifically designed to help

[2]Note that only when the variance is finite for the sum of independent and identically distributed random variables is the converging distribution Normal. When the finite variance condition is relaxed and random variables with infinite second moment are allowed, the limiting distribution is, under very general conditions, Lévy-Stable.

with the class of graph under question. The reader is referred to Chapter 3 for a more in depth review of the algorithm and the additions.

Numerical evidence is also presented of the analogous probability distribution for the Number Partition Problem (NPP) and to show that NPP instances whose elements are drawn from this distribution also display a phase transition. There is a lack of good approximation algorithms for the NPP, even in the region where a solution almost surely exists, and so the primary consideration is on describing the phase transition without giving undue focus on the pickup in search cost for the algorithms used. The failure of current algorithms to solve the Number Partition Problem, even in the region where a solution almost surely exists, is also discussed and a candidate class of algorithms is presented that holds promise for solving large NPP instances in the region of the phase transition where a solution is almost sure to exist.

1.6 Outline

Chapter 2 will review previous work in the field of random graph generation and finding Hamiltonian Cycles therein. A brief overview is given of some probability theory, graph generation methods, analytic results pertaining to Hamiltonian Cycles and solver strategies. The Number Partition Problem is also reviewed with previous work done with its associated phase transition and search algorithms.

Chapter 3 will discuss the algorithms used for graph generation methods, solver strategies and software used when analyzing the phase transition for the Hamiltonian Cycle Problem. Algorithms and instance creation will also be discussed for the Number Partition Problem.

Chapter 4 will give numerical evidence of the phase transition and pickup in search cost for graphs whose degree distribution is chosen from a modified Lévy-Stable distribution. Some of the pitfalls of the algorithms used will be discussed at the end.

Chapter 5 will discuss the phase transition of the Number Partition Problem and present numerical evidence for what the phase transition looks like when instances are drawn from a symmetric, absolute valued and truncated Lévy-Stable distribution. Pitfalls of current algorithms and a candidate class of algorithms to solve NPP instances in the region where solutions almost surely exist are discussed.

Chapter 6 will conclude with a discussion of results, potential pitfalls and future work.

Chapter 2

Previous Work

Previous work is presented on random graphs, random graph generation, Hamiltonian Cycles, the phase transition of the Hamiltonian Cycle Problem and algorithms used to determine Hamiltonicity. The Number Partition Problem (NPP) is also presented along with work done on analysis of its phase transition. Though not further pursued in this thesis, a brief description of 3-SAT, its phase transition and the recent algorithms to analyze large instances is provided as it provides context for many of the ideas and motivations behind this thesis.

2.1 Phase Transitions of Hamiltonian Cycles

Erdös and Rényi introduced the concepts of random graphs in [42]. Let N be the number of vertices. Choose a probability, p, and consider each potential edge between the N vertices, joining an edge between them with probability p. One can also consider fixing the number of edges, M, then choosing edges to place in the graph by a random uniform choice from the set of ordered pairs of vertices. These two formulations have minor differences but, under generally lax conditions for the Hamiltonian Cycle Problem, the $G_{N,p}$ ensemble can be replaced by the $G_{N,M}$ ensemble by setting $p = M/\binom{N}{2}$ (See [18], Theorem 2.2). The two formulations will be used interchangeably as the need arises.

Erdös and Rényi were first to notice a phase transition for the appearance of the giant component [43], [44]. This work provided the basis for much work done on other qualities of the graph, most notably for this thesis, the appearance of a Hamiltonian Cycle. The appearance of a Hamiltonian

Cycle and the appearance of the giant component are distinct graph qualities but both of them follow a rapid transition of appearing as the edge probability parameter, p, is increased.

The microscopic quantity that is varied that induces the macroscopic change in state, or phase change, is usually called the order parameter. The property that is searched for determines the exact location of the phase transition point. Finding the suitable parametrization for a given NP-Complete class is still an open problem ([77]).

Following Bollobás's reference, [18], the order parameter for the phase transition of Hamiltonian Cycles in Erdös-Rényi random graphs is:

$$M(N) = (N/2)(\ln N + \ln\ln N + c_N)$$

where $M(N)$ is the number of edges as a function of the number of vertices, N, and c_N is the critical parameter.

Cheeseman, Kanefsky and Taylor [29] gave numerical results for search cost when running algorithms on Erdös-Rényi random graphs and reported an exponential increase in computational search cost near the critical threshold. Unfortunately their choice of heuristics turned out to be ill suited and with better algorithms one can almost surely determine Hamiltonicity in Erdös-Rényi random graphs in polynomial time.

Bollobás, Fenner and Frieze [19] improved on algorithms presented by Angluin and Valiant [8] and Shamir [96] to create their HAM algorithm to find Hamiltonian Cycles in Erdös-Rényi random graphs and show:

$$\lim_{N\to\infty} P(\text{HAM finds a Ham. Cycle in } G_{N,M}\ ) = \begin{cases} 0, & \text{if } c_N \to -\infty \\ e^{-e^{-c}}, & \text{if } c_N \to c \\ 1, & \text{if } c_N \to \infty \end{cases}$$

HAM runs in $o(n^{4+\epsilon})$ time, for some $\epsilon > 0$ giving us an almost sure polynomial time algorithm to find Hamiltonian Cycles in Erdös-Rényi random graphs.

In practice one can do much better. One such optimization is proposed by Culberson and Vandegriend [105] who constructed a lowest degree first backtracking algorithm with some additional heuristics to prune the graph as a path is tried. Chapter 3 will go into more detail about Culberson

and Vandegriend's algorithm and heuristics used. A more detailed description of other algorithms used for Hamiltonian Cycle determination is given in the next subsection.

Culberson and Vandegriend found that hard instances were rarely encountered and became more infrequent as larger instances of graphs were generated. In this way, the random Hamiltonian Cycle Problem for Erdös-Rényi graphs is one that is theoretically known to have an efficient solution and has a practically efficient algorithm.

Culberson and Vandegriend pointed out that their algorithm becomes exponential in its search cost when graph instances are Interconnected-Cutset (ICCS) graphs even though ICCS graphs are known to have a polynomial time algorithm to verify Hamiltonicity. In this thesis, Culberson and Vandegriend's algorithm has been used to provide numerical evidence of an increase in search cost for graphs whose degree sequence is chosen from a Lévy-Stable distribution, but this should be taken in context of the known shortcomings of this particular algorithm.

2.1.1 Solvers for the Hamiltonian Cycle Problem

What follows is a brief description of algorithms employed in finding Hamiltonian Cycles. Vandegriend's Master's Thesis ([104]) gives an in depth survey of algorithms used in solving Hamiltonian Cycle Problems that are listed below.

Pósa [93] introduced a randomized algorithm based on a heuristic that does path rotations when extending a potential path becomes impossible. This algorithm is discussed in Chapter 3 and the reader is referred there for details. Briefly, a path is extended until no further extension is possible. When furthering a path proves impossible, a middle vertex in the path, w_k, is chosen in the current path, $(w_0, w_1, \ldots, w_{k-1}, w_k, w_{k+1}, \ldots, w_{n-2}, w_{n-1})$, that is connected to an endpoint, w_{n-1}. A rotation is then performed, creating a new path $(w_0, w_1, \ldots, w_{k-1}, w_{n-1}, w_{n-2}, \ldots, w_{k+1}, w_k)$ with, w_k as the new endpoint. The path is then extended if possible. It is possible to get into dead ends precluding this algorithm from finding a solution. Another drawback is the possibility of getting into infinite loops of rotations that preclude the algorithm from ever succeeding but whose condition is not trivially detectable. Angluin and Valiant [8] introduced the UHC algorithm, a variant on Pósa's idea, that deletes edges as a path is being created. Angluin and Valiant also introduced the DHC algorithm which extends the ideas from the UHC algorithm to directed graphs.

Bollobás, Fenner and Frieze [19] introduce the HAM heuristic algorithm which extends Pósa's in two significant ways. The first is adding a cycle extension heuristic whereby if a path cannot be

extended, the current path is checked to see if it also a cycle. If so, the endpoints for the path can be made to be any two neighboring vertices in the current path. If the current path is not a cycle, then a rotation is done as normal. The second is a backtrack component that stores all potential rotation choices and does a breadth first search to continue searching. Frieze gives a SparseHam [50] variant on the HAM algorithm by making it depth first search, only allowing extension from one endpoint and storing the edge used in the rotation to preclude that path from being tried again.

The HideHam algorithm by Broder, Frieze and Shamir [27] attempts to exploit low degree vertices in graphs by creating a set of low degree vertices and then using them to create a set of disjoint paths that each have at least one of the vertices from the set. These disjoint paths are extended via rotational or cycle extensions and then joined together.

Brunacci introduces the DB2 and DB2A algorithm [28] by converting the Hamiltonian Cycle Problem instance into a Traveling Salesman problem instance, where each of the edges is given a weight according to whether they are forced (a neighboring vertex of degree 2), normal or non-edges. Disjoint cycles are initially constructed by connecting edges to the path sorted by lowest degree first, making sure to connect forced edges first. The algorithm terminates should connecting forced edges preclude a Hamiltonian Cycle from occurring. The cycles are then connected to form a path and the algorithm proceeds via a series of rotations to move the current cycle onto edges in the graph that are labeled as normal. DB2A is merely a version of DB2 intended for use with directed graphs by converting the original instance to a directed Hamiltonian Cycle Problem instance using standard techniques.

Kocay and Li [67] introduce LongPath which is a backtracking algorithm with more complicated rotations and extension heuristics. LinearHam by Thomason [100] employs chain extensions specifically tailored to Erdös-Rényi random graphs, giving it expected $O(cn/p)$ running time when graphs are drawn from $G_{n,p}$. The MultiPath algorithm by Kocay [67] does a complete search by keeping a list of multiple paths and attempting to extend and connect where appropriate while pruning edges off of partial paths to test for a backtrack condition.

The KTC algorithm by Shufelt and Berliner [97] employs a list of 26 pruning techniques attached to a backtracking algorithm. Culberson and Vandegriend [105] borrowed some of the more successful pruning techniques from KTC and added them to a complete backtracking algorithm with random restart. The details of the pruning techniques will be covered in more detail in Chapter 3 as Culberson and Vandegriend's algorithm is employed in this thesis on graphs of small size.

Because of its generality and success in Erdös-Rényi random graphs, Culberson and Vandegriend's algorithm was chosen for graphs of small ($N < 100$) size. For larger graphs, Pósa's algorithm was employed for its speed and simplicity. Chapter 3 goes into further details of both of these algorithms and the minor alterations made.

2.2 Power Law Degree Distributed Graphs

Many real world graphs display a power law in their degree distribution, from graphs based on the Web connectivity [5] to social networks [7], [87]. These power law degree distributed graphs have a much different characteristic than the Erdös-Rényi graphs, displaying diverging second moment in their degree distribution and having a smaller average diameter ([101], [102], [103]). The diverging second moment for the degree distribution gives the power law degree distributed graphs a much lower average diameter than Erdös-Rényi random graphs. Erdös-Rényi random graphs have diameter approximately $\ln(N)$ whereas power law degree distributed random graphs have diameter that is either a constant or $\ln(\ln(N))$ depending on whether the critical exponent $\alpha \in (0, 1)$ or $\alpha \in (1, 2)$ respectively ([101], [102] and [103]).

Bianconi and Marsili [14] give analysis on loops of all lengths on power law degree distributed graphs and show that small loops are much more common in graphs whose second moment in the degree distribution diverges. Gleiss et all [56] confirm this in known real world power law degree distributed graphs of large metabolic networks to find that triangles are much more common than in their Erdös-Rényi counterparts.

Molloy and Reed [85] introduced the "Erased Configuration Model" which pairs vertices based on a degree sequence chosen proportional to the remaining free slots. Britton, Deijfen and Martin-Löf [26] show that the Erased Configuration Model asymptotically approaches the prescribed degree distribution chosen. While not guaranteeing the exact degree sequence input, the erased configuration model is chosen as the graph generation method in this thesis for its simplicity and speed.

See Berger [13] and Chung and Lu [33] for discussions on other types of graphs and their generation. Blitzstein and Diaconis [15] show a polynomial time algorithm to determine whether a degree sequence is feasibly realizable as a graph and give an importance sampling method to choose from the space of possible graphs. From an algorithmic perspective this is near ideal but their algorithm presented has a worst case run-time of $O(n^2\bar{d})$, for n vertices with average degree $\bar{d}$. Blitzstein

and Diaconis discuss the average run-time of their algorithm stating that "... we do not have a better bound on the average running time than the worst case running time ...". From personal experimentation, Blitzstein and Diaconis's algorithm is not fast enough in its average run time to be used for the quantity of instances needed in this thesis's analysis. For this reason, Blitzstein and Diaconis's algorithm is not used.

Reed and Hughes [94] gives a good discussion on why power laws are so prevalent in nature. See Nolan [88], Zolotarev [108] and Feller [46] for references and further discussions on Lévy-Stable distributions.

It could be that the spectra of a graph, either from its adjacency matrix or one of its close relatives, such as its Laplacian, is a better quantitative description of graphs rather than its degree distribution. In this thesis, the degree distribution is used, but for further reference on spectral graph theory, see Chung [32] and Chung and Lu [33]. Farkas, Derényi, Barabási and Viscek [45] discuss real world graphs and their deviation from Wigner's semi-circle law. Mihail and Papadimitriou [83] give arguments for showing that the spectrum of a graph is highly tied to its degree distribution and show, for graphs with a degree distribution that is power law, the first few eigenvalues display power law behavior.

One could imagine constructing a random instance from another NP-Complete problem, transforming it to a Hamiltonian Cycle instance via some reduction, and then analyzing the resulting complexity of the graphs produced. For example, by using a standard reduction from 3-SAT to Hamiltonian Cycle, one could generate a random instance in 3-SAT, reduce it to a Hamiltonian Cycle and note the difficulty. This method introduces a problem of adding structures that are high in degree in the resulting instance.

One could run standard algorithms on this reduced graph but most algorithms are optimized to run on random graphs that make implicit assumptions about how random graphs look locally. In the above example, one of the standard reductions from 3-SAT to Hamiltonian Cycle constructs the graph by introducing a set of small widgets ([36]). These widgets are highly structured and the resulting graph is highly structured. Shortcomings in algorithms with this assumption could be overcome and redesigned to exploit the structure that was introduced from the reduction but at a cost of a polynomial increase in run time.

In the author's opinion, the deeper issue is that focusing on graph instances reduced from some other NP-Complete class obfuscates questions about intrinsic complexity. Assuming one could perfectly

overcome the structure introduced from the reduction, one is left with dealing with the intrinsic complexity of the underlying distribution used in creating the original instance while leaving questions unanswered about what distribution or where the intrinsically complex instances are located in the target NP-Complete class. It is the author's opinion that focusing on random graph instances chosen directly from a graph ensemble, rather than a derived one whose complexity is inherited from the original NP-Complete class, provide more insight into the intrinsic difficulty of the NP-Complete class under investigation.

2.3 Phase Transition in the Number Partition Problem

Fu and Anderson [51], perhaps now infamously, described the Number Partition Problem as one in which no phase transition was to be found. Gent and Walsh [53] were the first to discover a proper parametrization of the phase transition of the Number Partition Problem, using the ratio of the list length to the number of bits needed for the maximum list element.

Later, Mertens used the saddle point method when analyzing the integral representation of the Number Partition Problem to describe a more in depth parametrization of the critical parameter [79]. Building on this work, Borgs, Chayes and Pittel have completely characterized the (uniform) random Number Partition Problem analytically [23]. Following Borgs et all, if one draws a_k uniformly at random from some range $[1 \dots 2^m]$, where $m = \kappa_n n$, where

$$\kappa_n = 1 - \frac{\log_2 n}{2n} + \frac{c_n}{n}$$

Then

$$\lim_{n\to\infty} P(\text{Perfect Partition Exists}) = \begin{cases} 0, & \text{if } c_n \to -\infty \\ 1 - (1/2)r(c_n)(r(c_n)+1) & \text{if } c_n \to c \\ 1, & \text{if } c_n \to \infty \end{cases}$$

Where

$$r(c_n) = exp(-\sqrt{\frac{3}{2\pi}}2^{-c_n})$$

Borgs, Chayes, Mertens and Nair [21] have considered the Number Partition Problem as a Random Energy Model (REM) by looking at the distribution of random partitions and associating an energy to the distance of the random sum to 0. They have proven that the Number Partition Problem behaves as a random energy model at low energies. This suggests that any algorithm using an energy as the distance of a partial or complete answer from 0 will do no better than random guessing as, even for small systems, the energy distribution near the desired solution is essentially random.

Korf introduced a Complete Karmarker Karp (CKK) algorithm to search for solutions [70]. CKK does well with small instances, $n < 30$, but search cost become prohibitive for anything larger, even well above the critical region where a solution is almost sure to exist. Considering the results [21], this is not surprising as the CKK algorithm is essentially using distance from the desired solution as a metric when searching the solution space. It is an open question whether there exists any efficient algorithm to solve the random Number Partition Problem. To the author's knowledge, even in the region where a solution is almost sure to exist no algorithm is known that can solve instances of even moderate size ($100 < n < 200$).

One technique that shows promise in solving NPP instances in the region past the phase transition, where solutions are almost sure to exist, is to use lattice reduction methods, such as LLL [74] and PSLQ [47]. LLL has been successful in breaking cryptographic protocols based on Subset Sum Problems ([6], [73], [37]). The use of lattice reduction techniques in solving the Subset Sum Problem were for 'low-density' regions where the bit size was much smaller than the list length. No work has been done in the 'high-density' region [1] where the bit size is much smaller than the list length, putting it well above the critical threshold and almost surely guaranteeing a solution.

2.4 Phase Transitions in Other NP-Complete Problems

The focus of this thesis is primarily on the Hamiltonian Cycle and the Number Partition problem as representative NP-Complete classes but there are many other NP-Complete problems that have been studied. It is instructive to briefly review a few these.

3-SAT has been observed to have a phase transition. 3-SAT asks for a Boolean assignment of variables $x_0 \ldots x_{n-1}$ for a given formula in Conjunctive Normal Form, where each clause has exactly 3 variables appearing in it. For example,

[1] To the author's knowledge

$$(x_0 \vee -x_1 \vee x_3) \wedge (-x_2 \vee x_3 \vee x_1) \wedge (x_3 \vee x_4 \vee -x_5)$$

is a small example of a 3 Conjunctive Normal Form with 6 variables and 3 clauses.

Random 3-SAT chooses a ratio of clauses to variables and assigns variables to clauses at random, usually choosing to negate a variable in a clause with probability 1/2. The phase transition has been observed for the Random 3-SAT problem when the ratio of clauses to variables is approximately 4.3 [66]. This is still under investigation and the reader is referred to [4], [3], [82], with Mertens, Mézard and Zecchina [80] using some more advanced statistical mechanical methods to determine the threshold value up to a high degree of accuracy.

Braunstein, Mézard and Zecchina have introduced their survey propagation ([25]) algorithm that has been successful at finding satisfying instances very near the critical threshold. Survey propagation has been developed to only work near this threshold but [25] have reported successfully solving satisfiable instances whose size ranges in the millions of variables. This suggests that 3-SAT instances, even very near the critical threshold, might be analogous to the Hamiltonian Cycle problem in that for this type of random distribution, hard instances might be vanishingly sparse. Unlike the Hamiltonian Cycle Problem though, finding a proof of unsatisfiability does not appear to be as easy. Resolution methods ([11], [12]) and Davis-Putnam branching procedures ([75]) both need an exponential increase in search cost to determine unsatisfiability near the critical threshold.

Phase transitions have been observed in many others NP-Complete problems, including the Traveling Salesman Problem [54], minimum vertex cover [58] and graph coloring [29]. For surveys of statistical mechanical methods in other combinatorial optimization problems the reader is referred to [59] and [77].

Chapter 3

Algorithms

In this chapter, algorithms used for investigating the phase transition of both the Number Partition Problem (NPP) and the Hamiltonian Cycle Problem are presented. Also presented are the algorithms used to generate problem instances for the NPP and random graphs for use in the Hamiltonian Cycle Problem.

Instance generation algorithms for the Number Partition Problem are presented followed by the Complete Karmarkar Karp (CKK) used as a solver. The Lenstra, Lenstra, Lovász (LLL) lattice reduction algorithm is briefly discussed along with some methods of encoding instances of Subset Sum and the Number Partition Problems in them.

Graph generation is then discussed followed by the two algorithms to search for Hamiltonian Cycles. Depending on the size of graph, different algorithms are chosen. For small graphs ($N < 100$) a complete algorithm is used. For larger graphs ($N > 100$) a probabilistic algorithm based on a heuristic by Pósa is used to search for a solution.

The internal graph representation for each of the algorithms used is briefly discussed at the end of this chapter.

3.1 Number Partition Generation and Algorithms

3.1.1 NPP Instance Generation

Problem instances are generated from a truncated Lévy-Stable distribution for a particular choice of γ, the scale parameter. Recall that the Lévy-Stable distribution in general does not admit a closed form solution for its probability density function for arbitrary $\alpha \in (0, 2]$. Instead, it is most often described by its characteristic function. For simplicity, the symmetric Lévy-Stable distribution is used. For a random variable that follows a Lévy-Stable distribution with characteristic exponent α and scale parameter γ, the probability distribution function is given by:

$$p_X(x) = \frac{1}{2\pi} \int_{-\infty}^{\infty} \exp(-ixt - |\gamma t|^{\alpha}) dt$$

Instances are generated by populating each number in the list from a draw of this random variable truncated to an integer. Algorithm 1 gives the pseudo-code used to generate an instance. `gsl_ran_levy` in algorithm 1 is a call to the `gsl_ran_levy` function in the GNU Scientific Library (GSL). `gsl_ran_levy` simulates a symmetric Lévy alpha stable random variable with exponent α and scale parameter γ.

Algorithm 1: GenerateNPPInstance

Input: int n, double $\alpha \in (0.0, 2.0]$, double $\gamma > 0.0$, double $\delta > 0$
Output: int a[ ]
foreach $i \in [0, 1, \ldots (n-1)]$ **do**
  $\lfloor$ a[i] = $\lfloor$ |gsl_ran_levy(γ, α)| + δ $\rfloor$;
return a ;

3.1.2 Complete Karmarkar Karp Algorithm

The Complete Karmarkar Karp (CKK) ([70]) algorithm is used to do a complete search for instances generated. CKK extends a heuristic by Karmarkar and Karp [63] to make a new, smaller, instance of the Number Partition Problem by identifying the two largest numbers in the list, taking them out and reinserting their difference. CKK proceeds on the new, smaller, list to try and find a solution. The idea is that this difference and reinsert operation has the potential to put a smaller number back into the list, reducing the problem complexity. By attaching a backtracking component and reinserting the sum of the two numbers after the difference has been tried makes this algorithm into

a complete one.

Pseudo-code for the CKK algorithm is shown in figure 2. If a solution is found, one can recreate the partition by retracing the steps taken in the recursion. CKK does well for small problem instances but for even moderate sized instances ($n > 30$), search cost becomes prohibitive. See Gent and Walsh's analysis of some heuristics used for NPP [55].

Algorithm 2: CKK

```
Input:  int A[]
Output: int r
if |A| = 1 then
  return |A[0]| ≤ 1 ;
a = SortDescending(A) ;
u = a[0], v = a[1] ;
a' = a[ u - v , a[2:n-1] ] ;
if CKK(a') then
  return 1 ;
a' = a[ u + v , a[2:n-1] ] ;
if CKK(a') then
  return 1 ;
return 0 ;
```

CKK's observed exponential search cost is most likely due to the distance metric behaving randomly when traversing the search space ([21], [22]). CKK is moderately better than random chance and works well for extremely small instances, so it is a good choice for $n < 30$.

3.1.3 LLL

The Number Partition Problem and the closely related Subset Sum Problem can be reformulated as a problem of finding a short vector on a lattice. The merits of this approach and its subsequent failure will be discussed later. In this section a very brief introduction to lattice reduction, the Lenstra, Lenstra and Lovász algorithm (LLL) and some encodings of NPP and Subset Sum will be provided.

A lattice, Λ, for a given set of basis vectors, $B = [b_0, b_1, \ldots, b_{n-1}]^T$, $b_k^T \in \mathbb{Q}^m$, is the set of points:

$$\Lambda(B) = \{\sum_{k=0}^{n-1} c_k b_k | c_k \in \mathbb{Z}\}$$

The shortest vector (or vectors), ε, is defined as the non-trivial lattice point (or points) whose length is of minimum 2-norm:

$$\varepsilon = \text{argmin}_v\{||v|| \ \ | \ v \in \Lambda(B)\}$$

Under the assumption $NP \nsubseteq RP$, approximating the shortest vector of a lattice to within a constant factor is NP-Complete ([65]). Instead we ask for an approximation to the shortest vector on the lattice. LLL is an algorithm to find an approximation to the shortest vector, finding vectors that are within an exponential factor of optimal. If v is the short vector returned by LLL, LLL guarantees:

$$||v|| \leq 2^{(m-1)/2}||\varepsilon||$$

Often this exponential factor is good enough and in practice vectors returned are much shorter than the exponential bound would imply. See Lenstra, Lenstra and Lovász [74] for the original introduction of the LLL algorithm and its use in solving integer polynomial factorization. Borwein [24] has a summary of LLL and its uses in integer relation detection. Yap [106] provides further details.

In what follows, Yap's notation, treatment and analysis will be used. Without loss of generality, we can assume that the basis vectors provided are linearly independent. The LLL algorithm finds a reduced basis, where a reduced basis is defined to be:

$$||b_i^*||^2 \leq 2||b_{i+1}^*||^2 \quad , i \in [0 \cdots n-2]$$

$B^* = [b_0^*, b_1^*, \cdots, b_{n-1}^*]^T$ are the orthogonal Gram-Schmidt vectors associated with the basis $B = [b_0, b_1, \cdots, b_{n-1}]^T$. The basis B can be reduced as a product of M and B^* such that $B = MB^*$. Here, B^* is the matrix of the Gram-Schmidt reduced vectors and M holds the history of row reductions made to produce B^*:

$$b_i = \sum_{k=0}^{i} \mu_{i,k} b_k^*$$

where

$$M = \begin{pmatrix} 1 & 0 & 0 & \dots & 0 & 0 \\ \mu_{1,0} & 1 & 0 & \dots & 0 & 0 \\ \mu_{2,0} & \mu_{2,1} & 1 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \mu_{n-1,0} & \mu_{n-1,1} & \mu_{n-1,2} & \dots & \mu_{n-1,n-2} & 1 \end{pmatrix}$$

Algorithm 3 gives the Gram-Schmidt (GS) algorithm for constructing B^* and its history stored in M from a given set of bases, B. Intuitively, the GS algorithm is iteratively constructing a set of orthogonal vectors by subtracting off the shared component of the vectors previously created. Figure 3.1 shows the first two steps of this process on the basis vectors b_0 and b_1. Here, e_0 and e_1 are unit vectors, orthogonal to each other, in the appropriate directions. [1].

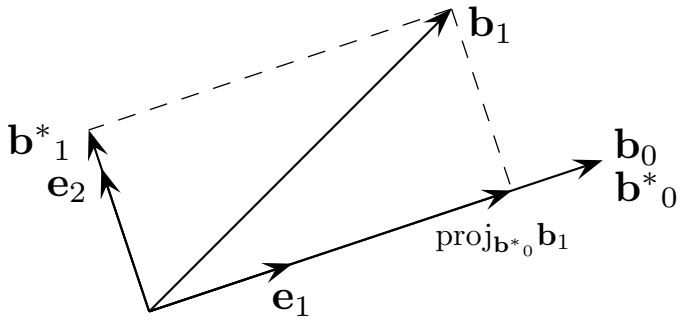


Figure 3.1: The first two steps of the Gram-Schmidt process on the basis vectors b_0 and b_1. b_1^* is produced by subtracting the shared component from $b_0^* = b_0$. Here, e_0 and e_1 are unit orthogonal vectors.

To generate a reduced basis, the LLL algorithm repeatedly applies a weak reduction step followed by a reduction step until the basis is reduced. The weak reduction step and the reduction step will be described next.

A weak reduction consists of subtracting an integral amount of one basis vector from another. The choice of coefficient is chosen so as to make each of the $|\mu_{i,j}| \leq 1/2$. Subtracting one row in M from another induces the same integer linear combination in B. A basis is said to be weakly reduced if:

[1] Code to produce this diagram uses a slightly modified version of the Gram-Schmidt_process.tex PSTricks source provided by user Gustavb from http://en.wikipedia.org/wiki/GramSchmidt_process. It's use is granted under public domain.

$$|\mu_{i,j}| \leq \frac{1}{2} \quad , 0 \leq j < i < n$$

Starting from $j = n - 2$ and going down, choosing:

$$c = \lfloor \mu_{i,j} \rceil \quad , i > j$$

Where $\lfloor \cdot \rfloor$ is the nearest integer function.

The appropriate basis vector is then transformed:

$$b'_i = b_i - cb_j$$

until the basis is weakly reduced. Starting in the lower right corner of the M matrix then working up and left until each $|\mu_{i,j}| \leq 1/2$ constitutes the weak-reduction step. Algorithm 4 provides the pseudo-code to produce a weakly reduced basis.

After a basis has been weakly reduced, a pair of basis vectors is chosen to swap that violate the reduction criteria. A strong reduction step is then to find i such that $||b^*_i||^2 > 2||b^*_{i+1}||^2$ and then to swap those two basis vectors. The choice may not be unique and any pair may be chosen that violates the reduction criteria. Algorithm 5 is provided for completeness. If no more reduction steps can occur, the basis is reduced and the algorithm terminates.

By noticing that $||\varepsilon|| \geq \min_{i=0\ldots(n-1)}(||b^*_i||)$ and that a reduced basis gives $||b_0||^2 = ||b^*_0||^2 \leq 2^{i-1}||b^*_i||^2 \leq 2^{m-1}||\varepsilon||^2$ it can be seen that the shortest lattice vector, ε, on $\Lambda(B)$ can be tied to the Gram Schmidt orthogonalization of B. The shortest lattice vector, ε, can then be compared with the lattice vector computed. Once LLL has found a reduced basis, this gives a basis vector, b_0, that is within an exponential bound of the optimal shortest vector, i.e. $||b_0|| \leq 2^{(m-1)/2}||\varepsilon||$.

The LLL algorithm is presented in algorithm 6.

If we define a measure of volume as:

$$V(B) = \prod_{i=0}^{n-1} V_i(B)$$

where

Algorithm 3: GramSchmidt

Input: $A = [a_0, a_1, \ldots, a_{n-1}]^T$
Output: $A^* = [a_0^*, a_1^*, \ldots, a_{n-1}^*]^T, M,$ s.t. $A = MA^*$
$M \in \mathbb{R}^{n,m}$;
$a_0^* = a_0$;
$[M]_{0,0} = 1$;
for $i \in [1, 2, \ldots, n-1]$ **do**
    $[M]_{i,i} = 1$;
    $a_i^* = a_i$;
    for $j \in [0, 1, \ldots, i-1]$ **do**
        $[M]_{i,j} = (a_i \cdot a_j^*)/(a_j^* \cdot a_j^*)$;
        $a_i^* = a_i^* - [M]_{i,j} \cdot a_j^*$;
return M, A^* ;

Algorithm 4: WeakReduce

Input: $B = [b_0, b_1, \ldots, b_{n-1}]^T$
Output: a B that is weakly reduced
decompose B: $MB^* = B$;
foreach $i \in [n-1, n-2, \ldots, 1]$ **do**
    foreach $j \in [i-1, i-2, \ldots, 0]$ **do**
        $q = \lfloor [M]_{i,j} \rceil$;
        $b_i = b_i - q b_j$;
        foreach $k \in [j, j-1, \ldots, 0]$ **do**
            $[M]_{i,k} = [M]_{i,k} - q[M]_{j,k}$;
return $B = [b_0, b_1, \ldots, b_{n-1}]^T$;

Algorithm 5: StrongReduce

Input: $B = [b_0, b_1, \ldots, b_{n-1}]^T$
Output: $B = [b_0, b_1, \ldots, b_{k-1}, b_{k+1}, b_k, b_{k+2}, \ldots, b_{n-1}]^T$ if $\|b_k^*\|^2 > 2\|b_{k+1}^*\|^2$, otherwise B is untouched
decompose B: $MB^* = B$;
foreach $i \in [0, 1, \ldots (n-2)]$ **do**
    if $\|b_k^*\|^2 > 2\|b_{k+1}^*\|^2$ **then**
        return $B' = [b_0, b_1, \ldots, b_{k-1}, b_{k+1}, b_k, b_{k+2}, \ldots, b_{n-1}]^T$;
return B ;

Algorithm 6: LLL

Input: $B \in Z^{n,m}$
Output: A reduced basis B
WeakReduce(B) ;
while *B is not reduced* **do**
    StrongReduce(B) ;
    WeakReduce(B) ;
return B ;

$$V_i(B) = \prod_{j=0}^{i} \|b_j^*\| = \sqrt{\det(B_i^T B_i)}$$

Where $B_i = [b_0, b_1, \ldots, b_i]^T$.

Then it can be shown that after each reduction step from a weakly reduced basis decreases this value by at least a constant factor. The integral value of $V(B)$, combined with a reduction by at most a constant value after the pair of reduction steps has been applied, guarantees termination. Proofs of polynomial running time and polynomial intermediate bit length are not provided and the reader is referred to [106] and [74] for a more in depth description.

Consider the following small example on a basis of 5 row vectors, $B \in \mathbb{Z}^{5,6}$:

$$B = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 32000 \\ 0 & 1 & 0 & 0 & 0 & 23000 \\ 0 & 0 & 1 & 0 & 0 & 13000 \\ 0 & 0 & 0 & 1 & 0 & 17000 \\ 0 & 0 & 0 & 0 & 1 & 11000 \end{pmatrix}$$

After running the LLL algorithm on the above basis of row vectors, the following reduced basis could be produced:

$$B' = \begin{pmatrix} 0 & 1 & 0 & -2 & 1 & 0 \\ 1 & 0 & -2 & -1 & 1 & 0 \\ -1 & 1 & -1 & 0 & 2 & 0 \\ 0 & 2 & -1 & 0 & -3 & 0 \\ 0 & 1 & -1 & 0 & -1 & -1000 \end{pmatrix}$$

See Appendix A for a worked example of the LLL algorithm run on a small example basis.

3.1.4 Number Partition and Subset Sum Problem Encodings in LLL

Three encodings of the Number Partition Problem and Subset Sum in terms of finding a reduced basis are provided.

Following Lenstra, Lenstra and Lovász [74], one of the simplest encodings is to create a basis whose

last element is each number in the NPP instance, multiplied by an appropriate factor. The following n x $(n+1)$ basis describes this approach:

$$B = \begin{pmatrix} 1 & 0 & \dots & 0 & Ca_0 \\ 0 & 1 & \dots & 0 & Ca_1 \\ \vdots & \vdots & \ddots & & \vdots \\ 0 & 0 & \dots & 1 & Ca_{n-1} \end{pmatrix}$$

Where $C \in \mathbb{Z}$. By choosing C appropriately big, an integer relation is guaranteed to be found if one exists within $\lceil C \cdot 2^{1/2-n} \rceil$ ([24]). Once a call to LLL is run on on the above basis, the coefficients can be read from the first n columns in the resulting b_0 vector.

Following Lagarias and Odlyzko ([73]), one can set up a Subset Sum instance in a similar manner. If $s = (1/2)\sum_{k=0}^{n-1} a_k$, then construct the $(n+1)$ x $(n+2)$ basis as follows:

$$B = \begin{pmatrix} 1 & 0 & \dots & 0 & 0 & Ca_0 \\ 0 & 1 & \dots & 0 & 0 & Ca_1 \\ \vdots & \vdots & \ddots & & \vdots & \\ 0 & 0 & \dots & 1 & 0 & Ca_{n-1} \\ 0 & 0 & \dots & 0 & 1 & -Cs \end{pmatrix}$$

Coster, Joux, LaMacchia, Odlyzko, Schnorr and Stern [37] proposed the following basis as an improvement:

$$B = \begin{pmatrix} 2 & 0 & \dots & 0 & 0 & Ca_0 \\ 0 & 2 & \dots & 0 & 0 & Ca_1 \\ \vdots & \vdots & \ddots & & \vdots & \vdots \\ 0 & 0 & \dots & 2 & 0 & Ca_{n-1} \\ -1 & -1 & \dots & -1 & 1 & -Cs \end{pmatrix}$$

Coster, Joux, LaMacchia, Odlyzko, Schnorr and Stern [37] also proposed the following basis as a further improvement:

$$B = \begin{pmatrix} n+1 & -1 & \dots & -1 & -1 & Ca_0 \\ -1 & n+1 & \dots & -1 & -1 & Ca_1 \\ \vdots & \vdots & \ddots & & \vdots & \vdots \\ -1 & -1 & \dots & n+1 & -1 & Ca_{n-1} \\ -1 & -1 & \dots & -1 & n+1 & -Cs \end{pmatrix}$$

These three encodings of the Number Partition Problem and Subset Sum problem are used to probe the region of the phase transition where solutions are almost sure to exist. Chapter 5 will briefly review the merits and subsequent failures of these encodings to provide solutions when large instances are generated.

3.2 Graph Generation and Algorithms

3.2.1 Graph Generation

The "Erased Configuration Model" ([85]) graph generation algorithm is used for graph construction. Speed of graph generation is a concern. Constructing graphs with the exact degree sequence specified is of little concern as long as the limiting distribution is the same. This is the case for the Erased Configuration Model ([26]).

The algorithm creates hooks for each vertex, where the number of hooks for each vertex is initially given by the degree sequence. Going from the vertex with the most hooks first, another hook is chosen uniformly at random from the pool still unattached. A connection is made for this pairing. For simplicity's sake, vertices are temporarily allowed to have self loops and multiple edges. Once the pool of hooks is exhausted, self loops are removed and multi-edges are collapsed into one. Algorithm 7 gives pseudo-code for this process.

To generate graphs with heavy-tails, a truncated Lévy-Stable distribution is chosen. For simplicity, the Lévy-Stable distributions chosen are symmetric and centered around the origin. Only the scale parameter and critical exponent are varied.

For each vertex, a random draw is taken from this distribution, forced positive and truncated to an integer. The value is then clamped to be within the minimum and maximum degree, if specified. If none are specified, the default minimum and maximum values are 0 and $n-1$ respectively. Algorithm

Algorithm 7: GenerateApproxDegSequenceGraph

Input: int n, int deg[ ]
Output: Graph G

```
int hook[ n ][ ] ;
int nhook=0 ;
Graph G ;
sort deg in descending order ;
foreach v in 0 ... (n − 1) do
    foreach j in 0 ... deg[v] − 1 do
        hook[ v ][ j ] = v ;
foreach v in 0 ... (n − 1) do
    while hook[ v ] do
        Choose vertex u uniformly at random from all entries in hook array ;
        Remove a hook for u and a hook for v from the hook array ;
        Add edge u ∼ v to G ;
Collapse multiple edges and remove self loops in G ;
return G ;
```

8 shows pseudo-code for this process.

Algorithm 8: GenerateLevyDegreeSequence

Input: α, γ, mindeg, maxdeg
Output: deg[ ]

```
for i → 0 to n − 1 do
    t = | gsl_ran_levy(α, γ) | ;
    if t < mindeg then
        t = mindeg ;
    if t > maxdeg then
        t = maxdeg ;
    deg[ i ] = t ;
return deg ;
```

The `gsl_ran_levy` in Algorithm 8 is a call to the `gsl_ran_levy` function in the GNU Scientific Library (GSL). The `gsl_ran_levy` function is used to simulate a symmetric Lévy alpha stable random variable with exponent α and scale parameter γ. Since graphs under consideration are less than 1000 vertices, arbitrary precision is not a concern.

3.2.2 Complete Hamiltonian Cycle Algorithm

A slightly modified version of Culberson and Vandegriend's algorithm ([105]) is used. This modified Culberson and Vandegriend's algorithm is a graph traversal algorithm with backtracking. A path is extended, if possible, and the graph is pruned. If no extension is possible, backtracking occurs.

There are three pruning techniques. The first is whenever a path is extended, all edges connected to

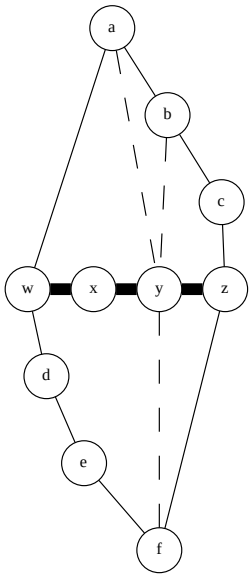


Figure 3.2: Pruning edges from a partial path through the graph. Bold lines are the partial path. Dashed lines are the edges that can be pruned.

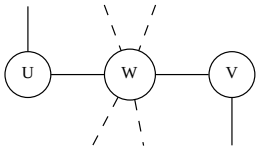


Figure 3.3: Pruning edges from a vertex, W, that is centered between two degree 2 vertices, U and V. A Hamiltonian Cycle must go through W through U and V, so all other edges can be pruned. Edges that can be pruned are dashed in this figure.

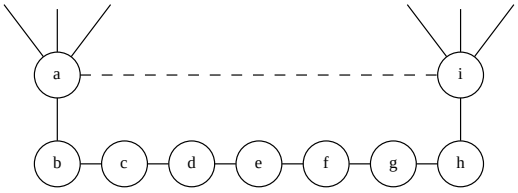


Figure 3.4: Chains of degree 2 vertices whose endpoints are connected can have their connected edge pruned. In this graph, the chain of degree 2 vertices are b through h, with the endpoints a and i connected. The connected edge that can be pruned is dashed.

non endpoints of the current path that are not part of the path itself, are removed, as no Hamiltonian Cycle will be able to use those edges. Figure 3.2 shows an example of this pruning technique, where edges that are available for pruning are dashed. The second pruning technique considers any vertex that sits in between vertices that have degree exactly 2. For all such vertices found, edges not connected to its degree 2 neighbors can be pruned from this middle vertex as a Hamiltonian Cycle must pass through this vertex using its degree 2 neighbors. Figure 3.3 shows an example of this pruning technique, where u and v are both degree 2 vertices, with all of w's edges able to be pruned save the ones connected to u and v.

Algorithm 9: PruneGraph

```
Input: Graph G
Output: Graph G'
stillPruning = true ;
while stillPruning do
    twov = all vertices of degree 2 ;
    foreach u in twov do
        (v_0, v_1) = Neighbors(u) ;
        for i → 0, 1 do
            foreach w ∈ Neighbors(v_i)/u do
                if deg(w) != 2 then
                    continue ;
                foreach x ∈ Neighbors(v_i)/{u, w} do
                    E = E/(x, v_i) ;
                    if deg(v_i) < 2 then
                        return ( ) ;
                    if deg(v_i) == 2 then
                        twov = twov ∪ v_i ;
                    if deg(x) < 2 then
                        return ( ) ;
                    if deg(x) == 2 then
                        twov = twov ∪ x ;
    (stillPruning, G) = PruneChain(G) ;
    if G == null then
        return ( ) ;
return G ;
```

The last pruning technique is to consider any chain of degree 2 vertices less than the total number of vertices in the graph. If there is an edge connecting the endpoints of this chain, it can be pruned as any path using this edge would close off a loop and preclude a Hamiltonian Cycle from occurring. Figure 3.4 shows an example of this. Algorithms 9 and 10 gives pseudo-code for these pruning techniques.

Any graph that can be partitioned into $k+1$ or more components by the removal of k vertices cannot

Algorithm 10: PruneChain

```
Input: Graph G
Output: boolean prunedEdge, Graph G'
(V, E) = G ;
marked[|V|] = () ;
twov = all vertices of degree 2 ;
foreach v ∈ twov do
    if marked[v] then
        next ;
    (v_0, v_1) = Neighbor(v) ;
    marked[v] = 1 ;
    for i ∈ 0, 1 do
        v_prev = v ;
        while deg(v_i) == 2 do
            if v_i == v then
                return (false, G) ;
            marked[v_i] = 1 ;
            t = v_i ;
            v_i = Neighbor(v_i)/v_prev ;
            v_prev = t ;
    if not v_0 ∼ v_1 then
        next ;
    E = E/(v_0, v_1) ;
    if deg(v_0) < 2 or deg(v_1) < 2 then
        return (undef, null) ;
    return (true, G) ;
return (false, G) ;
```

Algorithm 11: CutSetHeuristic

```
Input: Graph G
Output: Boolean noham
(V, E) = G ;
k = 1 ;
foreach v ∈ V, maximum degree first do
    remove v from G ;
    c = NumComponents(G) ;
    if c > k then
        return false ;
    k = k + 1 ;
return true ;
```

have a Hamiltonian Cycle in it. This can be seen by a pigeon hole argument as any Hamiltonian Cycle must visit each of the $k+1$ or more components with only k vertices to connect them. See Bondy and Murty [20] for a proof. The converse is not true as graphs exist for which there is no Hamiltonian Cycle yet no choice of k vertices to remove will partition the graph into $k+1$ or more components. Finding a set of k vertices that partition the graph into $k+1$ or more components provides a certificate of non-Hamiltonicity, whereas the inability to find such a cut-set gives no guarantee as to whether a Hamiltonian Cycle exists in the graph.

Using this idea, a cut-set heuristic is added to Culberson and Vandegriend's algorithm and is used as a test to find graphs for which no Hamiltonian Cycle can occur. Graphs are initially filtered through the cut-set heuristic, once at the start of the search, before the back tracking algorithm commences. The cut set heuristic works as follows: The cut-set heuristic algorithm proceeds in n iterations. At iteration k, the vertex with the largest degree is taken out and the graph is checked to see if there are greater than k connected components. If, during this removal process, the number of connected components, c, is ever greater than k, we know that no Hamiltonian Cycle can exist. One can choose any schedule of vertex removal for this heuristic. The maximum degree first is chosen as it has been the author's observation that choosing large degree vertices first for this removal works best, most likely due to the larger connectivity of the high degree vertices and thus a higher likelihood of separating the graph into more components. Algorithm 11 gives pseudo-code for this heuristic.

A schedule of lowest degree first is used when traversing. If the nodes searched reaches a threshold, the algorithm is restarted with another vertex, chosen at random, and the threshold doubled. In this way bad initial choices are mitigated against and the threshold will eventually be large enough to traverse the whole search space.

A bad initial pick of starting vertex could lead to an unnecessary increase in search cost were a better vertex initially picked. See Culberson and Vandegriend [105] for details about observed inflated run-times when searching for Hamiltonian Cycles in Erdös-Rényi random graphs. As an extra precaution against bad initial vertex choice, the algorithm is initially run without the exponential threshold restart, setting the threshold to $2n$ and running the algorithm, using each vertex in the graph as the starting vertex. Only after this initial check is run on all vertices is the complete algorithm run to determine Hamiltonicity. Algorithm 12 gives the pseudo-code for the recursive algorithm to find a Hamiltonian Cycle and Algorithm 13 gives the pseudo-code for the complete algorithm.

Algorithm 12: FindHamCycle

Input: Graph G, path p, int l, int MaxIter, int CurIter
Output: int Iter, Boolean found, path p
CurIter = CurIter + 1 ;
if *CurIter* $\geq$ *MaxIter* **then**
  return (CurIter, false, ( )) ;
$(V, E) = G$;
if $|p| = |V|$ **then**
  if $p_0 \sim p_{n-1}$ **then**
    return $(CurIter, true, p)$;
  else
    return $(CurIter, false, ())$;
$G' = PruneGraph(G)$;
$vchoice = DegSortAsc(Neighbors(p_l)/p_{l-1})$;
foreach $v \in vchoice$ **do**
  $p_{l+1} = v$;
  $G'' = PrunePath(G', path)$;
  $(CurIter, r, pp) = FindHamCycle(G'', p, l + 1, MaxIter, CurIter)$;
  if r **then**
    return $(CurIter, true, pp)$;
return $(CurIter, false, ())$;

Algorithm 13: FindHamCycleComplete

Input: Graph G, path p, length l
Output: int Iter, Boolean found, path p
$(V, E) = G$;
if *CutSetHeuristic(G) is false* **then**
  return $(Iter, false, ())$;
for $v \in V$ **do**
  $MaxIter = 2|V|$;
  (Iter, found, p) = FindHam(G, p, 0, $2|V|$, 0) ;
  if *found* **then**
    return (Iter, true, p) ;
found = false ;
while *not found* **do**
  (Iter, found, p) = FindHam(G, p, 0, MaxIter, 0) ;
  if *found or (not found and Iter < MaxIter)* **then**
    return (Iter, found, p) ;
  MaxIter = 2 MaxIter ;

Algorithm 14: PrunePath

Input: Graph G = (V, E), path p
Output: Graph G'
foreach $i \in [1, \ldots, |p| - 2]$ **do**
  foreach $w \in Neighbors(p_i)/\{p_{i-1}, p_{i+1}\}$ **do**
    $E = E/(p_i, w)$;
return (V, E) ;

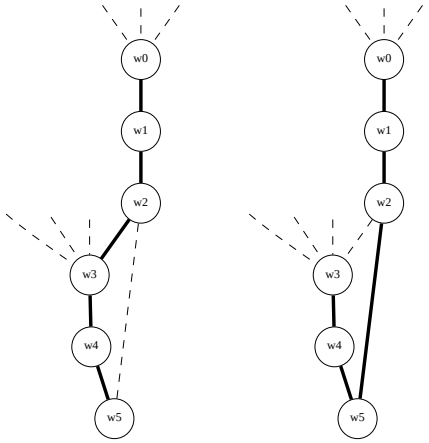


Figure 3.5: An example of the Pósa Heuristic used to rotate the current path so that extension is still possible.

With all these heuristics combined, finding Hamiltonian Cycles in Erdös-Rényi graphs becomes much easier. The author was unable to find any instances of Erdös-Rényi graphs that took more than a small constant factor of n in nodes traversed to determine Hamiltonicity. This suggests that, while the pruning techniques and checks for cut-sets are only heuristics that, in general, do not give an exponential gain in algorithmic speed, they are well suited to search for Hamiltonian Cycles in Erdös-Rényi graphs.

Culberson and Vandegriend's algorithm did not include the CutSetHeurstic or the initial small threshold check for Hamiltonian Cycles. The initial small threshold check was added after noticing that nearly all Erdös-Rényi random graphs were solved by a good initial vertex choice. The CutSetHeurstic check was added after noticing that many Lévy-Stable degree distributed random graphs in the impossible region were easily verified to have no Hamiltonian Cycle with this heuristic in place. Without the cut-set heuristic, many Lévy-Stable degree distributed random graphs are not easily found to have no Hamiltonian Cycle with just Culberson and Vandegriend's algorithm.

3.2.3 Pósa Heuristic

For larger graphs, a non-complete randomized algorithm is desirable for speed concerns. The Pósa heuristic transforms the current incomplete path in an attempt to jostle it into a state where the path can be further extended. Assume your current path of length l is $w_0 w_1 \dots w_{l-1} w_{l-1}$. Choose a random k such that w_k and w_{l-1} are connected. Remove the edge (w_k, w_{k+1}) from the current path and add edge (w_k, w_{l-1}) to the current path, making a new path $w_0 w_1 \dots w_{k-1} w_{l-1} w_{l-2} \dots w_{k+1} w_k$.

The hope is that the path from w_k can now be further extended. If no choice of k is available, the algorithm can be restarted. Figure 3.5 shows an example of applying the Pósa Heuristic for a portion of a graph.

Algorithm 15: Pósa

```
Input: Graph G, Iter
Output: path p
Iter = 0;
visited = ø ;
p_0 = random vertex ;
l = 1;
repeat
    if |Neighbors(p_{l-1})/visited| > 0 then                    /* extend path */
        p_l = a random vertex from Neighbors(p_{l-1})/visited ;
        visited = visited ∪ p_l ;
        l = l + 1 ;
        continue ;
    else if l ≤ 2 or deg(p_l) == 1 then
        return ø ;
    else                                                        /* rotate */
        A = Neighbors(p_{l-1}) ∩ (visited/p_{l-2}) ;
        w = choose a random vertex from A ;
        p = p_0 p_1 p_2 ... p_{k-1} p_{l-1} p_{l-2} ... p_{k+1} w ;
        Iter = Iter + 1 ;
until Iter == MaxIter or cycle is found ;
if Iter == MaxIter then
    return ø ;
else
    return p ;
```

This algorithm does not determine non-Hamiltonicity. For very large graphs, though, this algorithm can be run efficiently and works reasonably well in practice. The reader is referred to chapter 4 for some comparisons on how the Pósa heuristic fares against the modified version of Culberson and Vandegriend's algorithm for Erdös-Rényi and Lévy-Stable degree distributed random graphs. Algorithm 15 provides pseudo-code for the algorithm using the Pósa heuristic.

3.3 Graph Implementation Details

There are two different ways graphs are implemented, depending on whether the complete algorithm or the Pósa Heuristic algorithm is used.

For small graphs used in the complete algorithm, each vertex has a list of its neighbors in a dynamic array. A "back-pointer" matrix is used that doubles as an adjacency matrix. If the $[i][j]$ entry in the

back-pointer matrix, a, is non-negative, this denotes there is an edge between vertex i and vertex j and the value of $a[i][j]$ is the position of vertex j in i's neighbor list. In this way, edges can be inserted and deleted with only a constant number of operations. This is desirable for the complete algorithm as edges are being removed and inserted frequently during execution of the algorithm. Arrays are kept for quick lookup of vertices of degree 2 so that the graph need not be traversed again when these vertices are needed.

For larger graphs, a simple vertex list is used instead, where each vertex has a neighbor list ordered by vertex name. The ordering allows for $O(\lg(n))$ adjacency tests. This simple structure is sufficient for the Pósa Heuristic algorithm and is used for graphs in the range of 100 to 1000 vertices.

Chapter 4

The Hamiltonian Cycle Phase Transition in Power Law Degree Distributed Graphs

In this section results of numerical simulations for finding Hamiltonian Cycles in random Lévy-Stable distributed degree sequence graphs are presented. First small graph generation ($N < 100$) is done and the numerical results for the probability of finding a Hamiltonian Cycle vs. the pickup in search cost for the modified complete algorithm, **FindHamCycleComplete**, of Culberson and Vandegriend, is shown. Often, the increase in search cost is due to graphs with no Hamiltonian Cycle present, so graphs with a Hamiltonian Cycle forced are considered and the pickup in search cost for **FindHamCycleComplete** is examined. Larger graphs ($100 < N < 1000$) run with the **Pósa** algorithm as the solver are then considered and the probability of finding a Hamiltonian Cycle and the pickup in search cost is examined. This chapter ends with a brief discussion on the conclusions and pitfalls of this type of analysis.

4.1 Introduction and Motivation

A Hamiltonian Cycle is a path through the graph, $G = (V, E)$, such that every vertex is visited exactly once and the ending vertex is adjacent to the starting one.

One can generate a random graph chosen from a particular distribution and then ask what the probability of finding a Hamiltonian Cycle is in the resulting instance. One such graph ensemble is when graphs are generated by choosing each edge independently and at random with some fixed probability, p. Erdös and Rényi first introduced and studied these types of random graphs ([42]). Graphs created in this manner are often called Erdös-Rényi random graphs.

Erdös and Rényi noticed in [43] and [44] that a phase transition occurred in the appearance of the so called giant component as the probability for edge creation increased. Here, a phase transition denotes a rapid transition from 0 to 1 as the probability of edge inclusion, p, varies. Also noticed was that a phase transition occurred in the probability of finding a Hamiltonian cycle whose probability function had many of the same characteristics as the phase transition for the giant component.

For Erdös-Rényi random graphs, Komlós and Szemerédi [69] and Korshunov [71] were first to show that the trivial condition of the minimum degree being 2 is sufficient for a Hamiltonian cycle to appear. Bollobás [17] was first to show that the probability of the minimum degree being 2 was equal to the probability of a Hamiltonian Cycle occurring.

Angluin and Valiant [8], Shamir [96] first presented almost sure polynomial time algorithms for Hamiltonian Cycle determination. This was later refined by Bollobás, Fenner and Frieze [19] who gave proofs of phase transition for the Hamiltonian Cycle in Erdös-Rényi random graphs and provide an almost sure polynomial time algorithm. The reader is referred to Bollobás's reference [18] for further reference and details.

The efficiency of finding Hamiltonian Cycles was later verified numerically by Culberson and Vandegriend in [105] where they showed that a complete backtracking search algorithm, with some additional heuristics, often did much better than the algorithm proposed by Bollobás, Fenner and Frieze. The algorithm of Culberson and Vandegriend is optimized for finding Hamiltonian Cycles on Erdös-Rényi random graphs and is not always effective on graphs that are not created from this domain. To highlight this, Culberson and Vandegriend also provided a class of known polynomially solvable graphs, the Interconnected-Cutset (ICCS) graphs, for which their algorithm exhibited exponential run-time.

Despite these drawbacks, a slightly modified version of Culberson and Vandegriend's algorithm, the **FindHamCycleComplete** as discussed in Chapter 3, is used on small graphs. For larger graphs a randomized search algorithm based on Pósa's heuristic [93], the **Pósa** algorithm discussed in Chapter 3, is used. Chapter 2 goes into more detail of other solvers. The reader is also referred to

Vandegriend's masters thesis [104] for a more in depth discussion of Hamiltonian Cycle solvers.

In this section, a class of random graphs are presented as a candidate for intrinsically hard graphs whose Hamiltonicity is hard to determine. Random graphs are generated by choosing a degree distribution from the family of slightly modified Lévy-Stable distributions. The power law tails of the Lévy-Stable distributions give the degree distributions a diverging second moment. This destroys the local tree-like structure that the Erdös-Rényi graphs enjoy ([81]) and could be the relevant feature that makes their Hamiltonicity intrinsically difficult to determine. The modified Lévy-Stable distributions are chosen for convenience, as opposed to some other distribution that displays power law behavior, as they are the convergent distributions of sums of stable random variables.

The success with Erdös-Rényi random graphs make Culberson and Vandegriend's algorithm a good starting point to discuss run-times of general solvers. Pósa's algorithm is better suited for large graphs because of its speed and simplicity. Both of these algorithms have drawbacks, which will be discussed later, and are only used as initial evidence that the graph ensemble presented produces intrinsically hard graphs whose Hamiltonicity is difficult to determine.

These results are presented with a note of caution: increase in search cost is intimately tied with the algorithms used. Claims about intrinsic difficulty have been made about Hamiltonian Cycles in Erdös-Rényi random graphs when the increase in search cost was only due to bad algorithmic choices. For example, it was a bad vertex schedule choice for the complete search that gave Cheeseman, Kanefsky and Taylor's algorithm in [29] a pickup in search cost for determining Hamiltonicity. This exponential increase in search cost completely disappears when better algorithms are chosen, as mentioned above. As Culberson and Vandegriend point out in [105], the converse is also true and graphs with known polynomial time algorithms to find Hamiltonian Cycles often subvert algorithms that fare well on Erdös-Rényi random graphs.

The random graph generation method and subsequent analysis is meant to suggest only a candidate class of random graphs whose Hamiltonicity is difficult to determine. There is a possibility that the random graphs presented are easily found to be Hamiltonian by a better algorithm not considered.

Culberson and Vandegriend [105] point out that one possible reason their algorithm fares so poorly on the class of ICCS graphs and has the occasional spike in run time for small Erdös-Rényi random graphs is due to the high variability in degree. This variability in degree is one of the defining features of generating graphs whose degree sequence is Lévy-Stable and this could be a source for the large

run times of Culberson and Vandegriend's algorithm on these classes of graphs. It is unclear to the author what algorithm would be best suited to take advantage of the high degree variability, should one even exist, and is a potential future avenue of investigation.

4.2 Small Graph Instances

A slightly modified version of Culberson and Vandegriend's algorithm was used, **FindHamCycle-Complete**, to find Hamiltonian Cycles in small graphs ($N < 100$ vertices) whose degree sequence was chosen from a modified Lévy-Stable distribution. The algorithms used in this section are briefly discussed below. The reader is referred to Chapter 3 for further explanation and pseudo-code. Culberson and Vandegriend's original algorithm can be found in [105].

The Erased Configuration Model by Molloy and Reed [85] is employed for creating a graph from its degree sequence. A degree distribution is created where each vertex degree is a random value drawn from a discretized random variable whose distribution is Lévy-Stable. After the degree sequence is chosen, a best effort approach is done to connect vertices, collapsing multiple edges into one and removing loops. See Chapter 3, algorithm **GenerateApproxDegSequenceGraph** for pseudo-code for this algorithm.

Once a graph is generated, **FindHamCycleComplete** is used to try to find a Hamiltonian Cycle. First, a cut set heuristic is employed to see if a removal of a subset of the vertices would produce enough disjoint components to preclude a Hamiltonian Cycle from occurring. Assuming this heuristic is passed, an initial attempt at finding a Hamiltonian Cycle is employed starting from each vertex in the graph but only running for $2n$ iterations. Assuming no Hamiltonian Cycle is found from this initial pass, a backtracking search is then started, using a schedule of least degree first to traverse the search space. A maximum cut off of iterations is initially set and if the algorithm has not terminated before hitting this maximum, the algorithm is restarted with another initial vertex pick and the maximum iteration count doubled.

At each point in the backtracking algorithm, three pruning techniques are employed. Firstly, any vertex straddled by vertices of degree 2 has the rest of its edges removed as no Hamiltonian Cycle can use them. Secondly, any non complete path of degree 2 vertices has the edge connecting the two endpoints removed, if it exists, as no Hamiltonian Cycle will be able to use it. Finally, edges incident to any of the non-terminal vertices used in the current path that are not part of the current

path are removed.

The only modification to Culberson and Vandegriend's algorithm has been the initial heuristics of the cut set and initial loop through the vertices. The cut set heuristic was added after noticing that, even in the probability 0 region, many power law degree distributed graphs needed an exponential number of iterations to decide Hamiltonicity. The addition of this heuristic helped mitigate this increase in search cost and is the main reason why the search cost is so low in the probability 0 region for the graphs considered.

The initial loop through the vertices and test to find a Hamiltonian cycle in $2n$ iterations was employed after noticing that, for Erdös-Rényi random graphs, the occasional increase in node traversal disappeared if the graph was initially checked this way. After this heuristic was employed, the author found that the occasional polynomial search cost noticed by Culberson and Vandegriend disappeared. This heuristic was kept for graphs generated from a power law degree distribution as it was so effective with Erdös-Rényi random graphs.

4.2.1 Numerical Results for Small Graph Instances

Figures 4.1, 4.2 and 4.3 show the probability of finding a Hamiltonian Cycle and nodes searched for the **FindHamCycleComplete** algorithm for $N \in \{20, 26, 32, 38\}$ for $\alpha \in \{0.5, 1.0, 1.5\}$ as a function of the scale parameter, γ. Each point represents 200 simulation runs. Run times are plotted on a semi-log scale.

Each graph was generated by a call to **GenerateLevyDegSequence** to generate a degree sequence where each entry was drawn from a symmetric, absolute valued and truncated Lévy-Stable distribution. This degree sequence was then passed to **GenerateApproxDegSequenceGraph** with a minimum degree of 2 to generate the graph. Passing a minimum degree into **GenerateApproxDegSequenceGraph** only inflates values in the degree sequence to the minimum value provided. Since **GenerateApproxDegSequenceGraph** is a best effort algorithm, this could lead to a graph returned that has a minimum degree less than the minimum degree passed in.

For each of the simulation runs in 4.1, 4.2 and 4.3, graphs whose actual minimum degree was less than 2 were filtered out and not tested for Hamiltonicity. Enough graphs were generated so as to provide for 200 graphs that met the minimum degree 2 requirement for each α, γ and N needed.

Figures 4.4, 4.5 and 4.6 show the number of 'Hard' instances found for $\alpha \in \{0.5, 1.0, 1.5\}$ respectively.

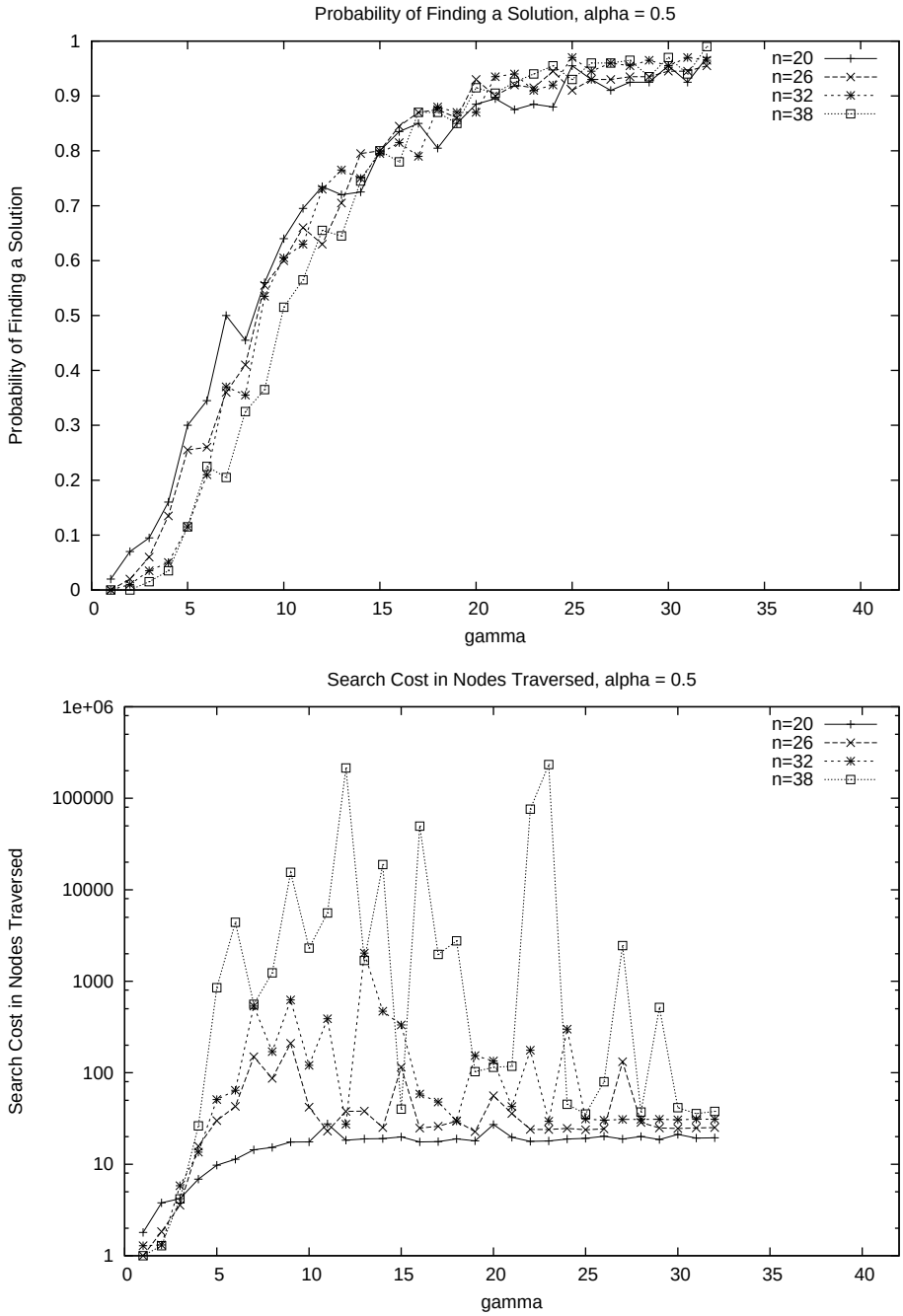


Figure 4.1: Probability and nodes searched for $\alpha = 0.5$. Each γ point represents the average of 200 graphs. A modified version of Vandegriend's algorithm (Algorithm **FindHamCycleComplete** from Chapter 3) was used to find Hamiltonian Cycles. Nodes searched are on a log-scale.

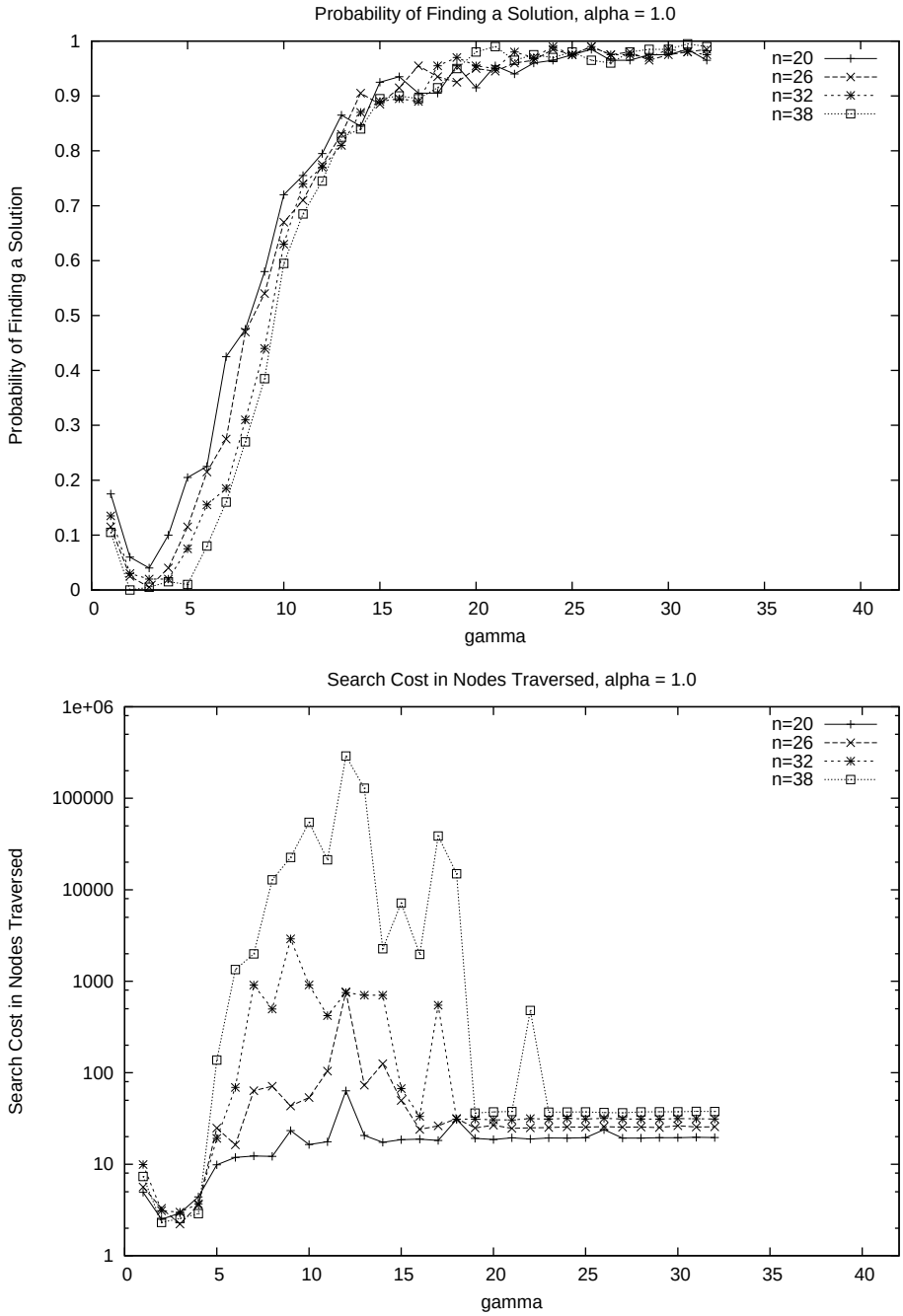


Figure 4.2: Probability and nodes searched for $\alpha = 1.0$. Each γ point represents the average of 200 graphs. A modified version of Vandegriend's algorithm (Algorithm **FindHamCycleComplete** from Chapter 3) was used to find Hamiltonian Cycles. Nodes searched are on a log-scale.

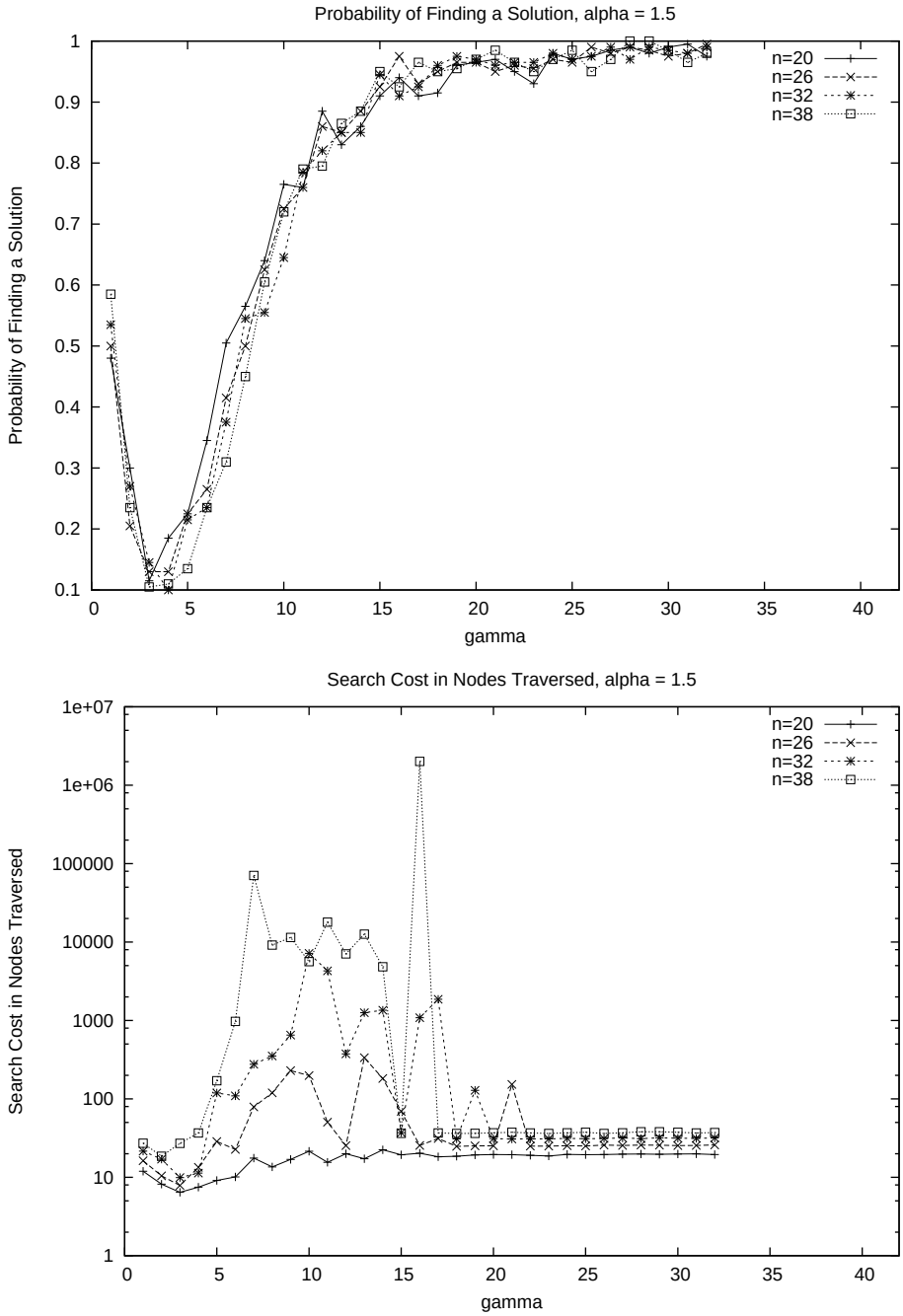


Figure 4.3: Probability and nodes searched for $\alpha = 1.5$. Each γ point represents the average of 200 graphs. A modified version of Vandegriend's algorithm (Algorithm **FindHamCycleComplete** from Chapter 3) was used to find Hamiltonian Cycles. Nodes searched are on a log-scale.

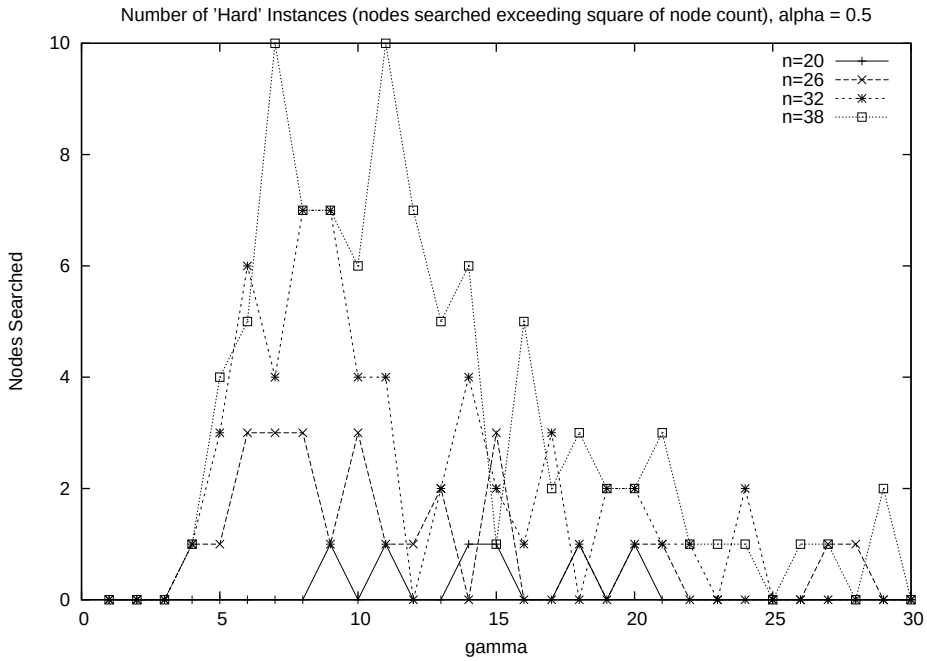


Figure 4.4: Number of 'Hard' instances found for $\alpha = 0.5$. A modified version of Vandegriend's algorithm (Algorithm **FindHamCycleComplete** from Chapter 3) was used to find Hamiltonian Cycles.

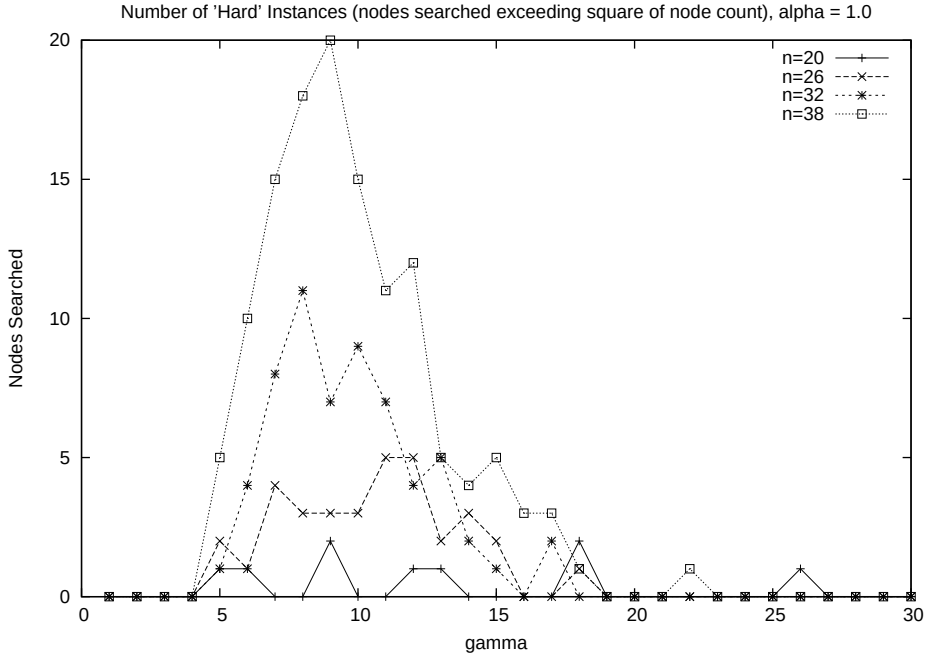


Figure 4.5: Number of 'Hard' instances found for $\alpha = 1.0$. A modified version of Vandegriend's algorithm (Algorithm **FindHamCycleComplete** from Chapter 3) was used to find Hamiltonian Cycles.

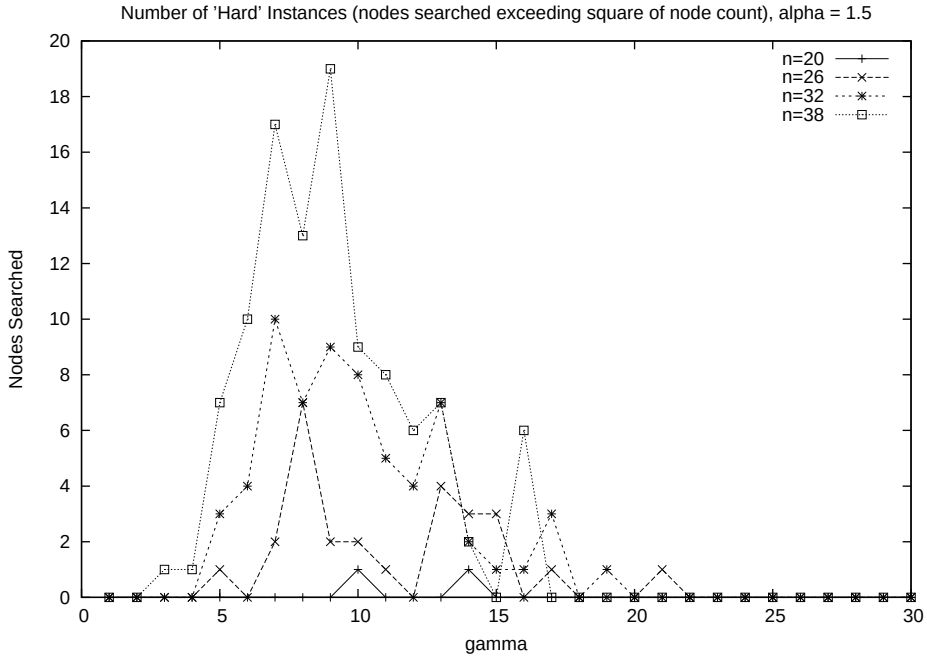


Figure 4.6: Number of 'Hard' instances found for $\alpha = 1.5$. A modified version of Vandegriend's algorithm (Algorithm **FindHamCycleComplete** from Chapter 3) was used to find Hamiltonian Cycles.

'Hard' in this case refers to graphs that take more than N^2 nodes of search in order to determine their Hamiltonicity.

The complete search was relaxed to only allow graphs that took more than N^2 nodes of search for Culberson and Vandegriend's algorithm. Figures 4.7, 4.8 and 4.9 show the percentage of graphs whose run time exceeded the N^2 node cutoff for $\alpha \in \{0.5, 1.0, 1.5\}$ respectively. Each point represents the average of 200 run times for graphs generated randomly. The N^2 node search cut off allows us to increase the maximum search size to $N = 220$.

Selecting graphs whose minimum degree was 2 has the unfortunate side effect of giving an initial inflation of the probability of finding a Hamiltonian Cycle. This will be discussed later in this chapter, but briefly, high α and low γ gives graphs that are too homogeneous in their degree sequence that, with the additional properties of minimum degree 2 and connectivity, temporary inflate the probability of finding a Hamiltonian Cycle.

An increase in γ has the effect of increasing the likelihood of picking higher degrees. As higher degree vertices get chosen in the degree sequence, we would expect this to eventually guarantee a Hamiltonian Cycle, which is numerically verified. Search cost quickly becomes too prohibitive, even for such small instances, and this is the reason for choosing graph sizes that are so small. It should

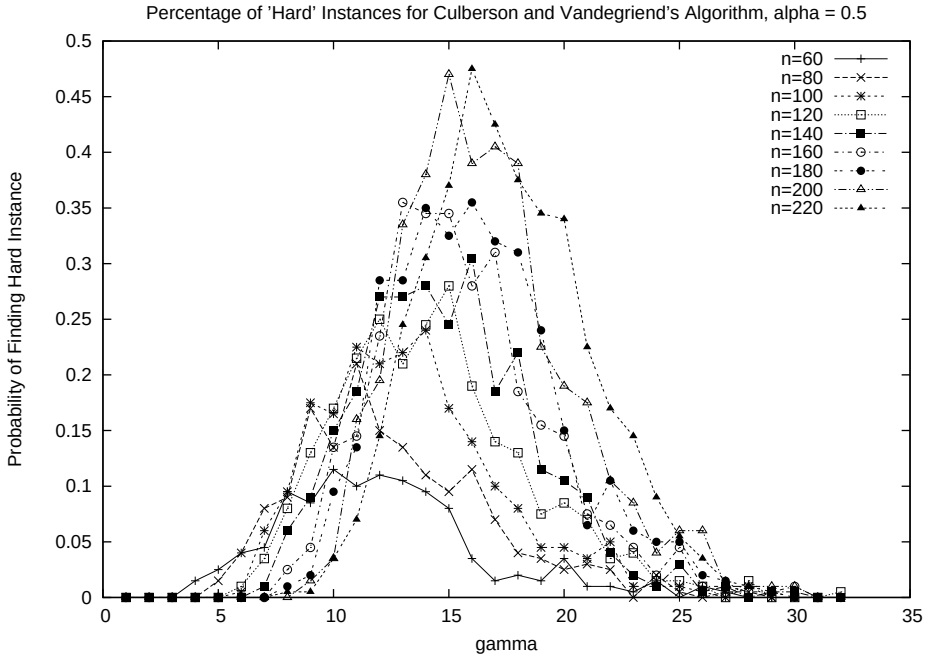


Figure 4.7: Number of 'Hard' graph instances generated for $\alpha = 0.5$ with the modified version of Vandegriend's algorithm (Algorithm **FindHamCycleComplete** from Chapter 3) used as the search algorithm. Graphs are only searched with a maximum of N^2 nodes of search before terminating the search algorithm. Each node represents the average of 200 graphs whose search exceeded the N^2 threshold.

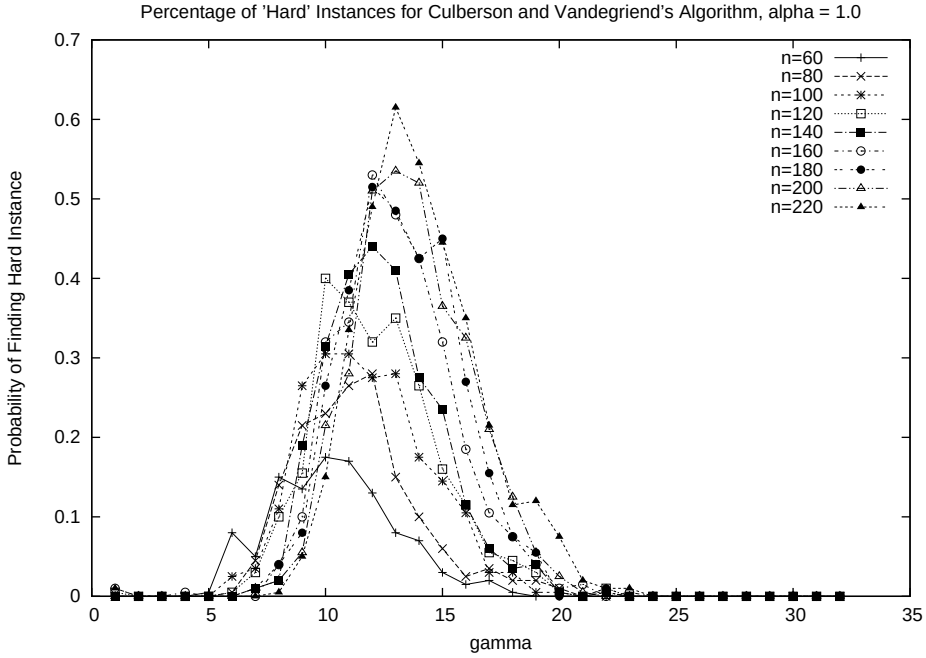


Figure 4.8: Number of 'Hard' graph instances generated for $\alpha = 1.0$ with the modified version of Vandegriend's algorithm (Algorithm **FindHamCycleComplete** from Chapter 3) used as the search algorithm. Graphs are only searched with a maximum of N^2 nodes of search before terminating the search algorithm. Each node represents the average of 200 graphs whose search exceeded the N^2 threshold.

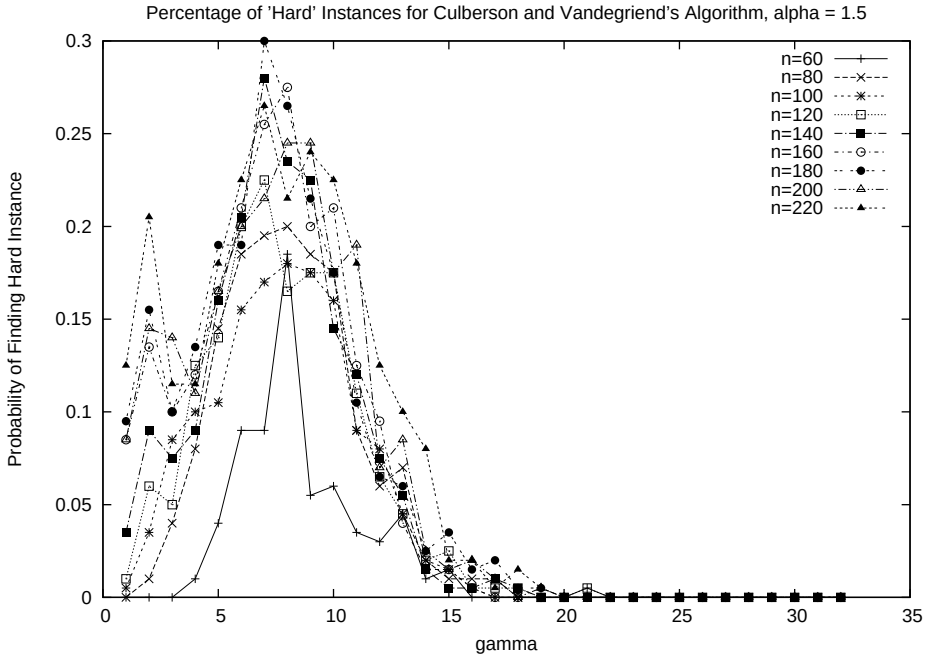


Figure 4.9: Number of 'Hard' graph instances generated for $\alpha = 1.5$, with the modified version of Vandegriend's algorithm (Algorithm **FindHamCycleComplete** from Chapter 3) used as the search algorithm. Graphs are only searched with a maximum of N^2 nodes of search before terminating the search algorithm. Each node represents the average of 200 graphs whose search exceeded the N^2 threshold.

be noted that nearly all of the contribution to the nodes searched comes from graphs which do not have a Hamiltonian Cycle. For the graphs whose search for Hamiltonicity was complete, only 4 graphs that had a Hamiltonian Cycle present took more than $2N$ iterations to find.

4.2.2 Small Graph Instances with Hamiltonian Cycles

Here, generated graphs have a Hamiltonian Cycle explicitly put in. A degree schedule is generated and a Hamiltonian Circuit is put in, decrementing the degrees of the vertices that lie on the Hamiltonian cycle chosen. The remaining vertices are paired as usual from the **GenerateApproxDegSequenceGraph** and the **FindHamCycleComplete** algorithm is run. A "round robin" heuristic has been employed for the solver: instead of immediately increasing the threshold when the number of nodes has exceeded the current bound, **FindHamCycleComplete** is run again with a different initial vertex until the whole list is exhausted before doubling the threshold and trying again. The number of nodes searched is then reported as if the algorithm had started from this vertex, should a Hamiltonian Cycle be found. This ensures that run times are not subject to bad initial vertex choices.

Figure 4.10, 4.11 and 4.12 show the average and standard deviations of nodes searched for $\alpha = \{0.5, 1.0, 1.5\}$ respectively. Often the average run time is inflated by just one or two hard instances. This can be seen in Figures 4.13, 4.14 and 4.15 show the number of instances found where the number of nodes searched exceeded N^2 (labeled as "Hard" graphs).

The increasing standard deviation and the jumps in run times between successive data points is an indication of the sporadic nature of generating graphs whose Hamiltonicity is not trivially found. While graphs that are difficult for this algorithm are relatively uncommon, they show up with increasing frequency. Even for these small graphs, the **FindHamCycleComplete** search cost become prohibitive and search time becomes unreasonable for graphs beyond $N \geq 200$.

4.3 Larger Graph Instances

In this section larger graphs ($N \in (64, 76, 91, \ldots, 796)$) are generated and the **Pósa** algorithm is used to try to find a Hamiltonian Cycle. See Chapter 3 for pseudo code of the **Pósa** algorithm. The **Pósa** algorithm is only used to randomly sample the space of large graphs and is not meant as a complete search algorithm. Briefly, the **Pósa** algorithm extends a path until a dead end is reached or a Hamiltonian Cycle is found. When further progress is not possible, a rotation is attempted whereby the current path is changed from

$$(w_0, w_1, \ldots, w_{k-1}, w_k, w_{k+1}, \ldots, w_{l-2}, w_{l-1})$$

to

$$(w_0, w_1, \ldots, w_{k-1}, w_{l-1}, w_{l-2}, \ldots, w_{k+1}, w_k)$$

where w_k and w_{l-1} share an edge. If no such rotation is possible, the algorithm is restarted. If a rotation is done, the path is extended if possible and the algorithm keeps going. A maximum iteration threshold is set and if the number of rotations attempted exceeds the maximum iteration count, the algorithm is restarted. If the maximum restart count is exhausted, the algorithm terminates. Cycles of vertex rotation choice are possible, precluding the algorithm from making any more progress and this is the motivation for including a restart component into the **Pósa** algorithm.

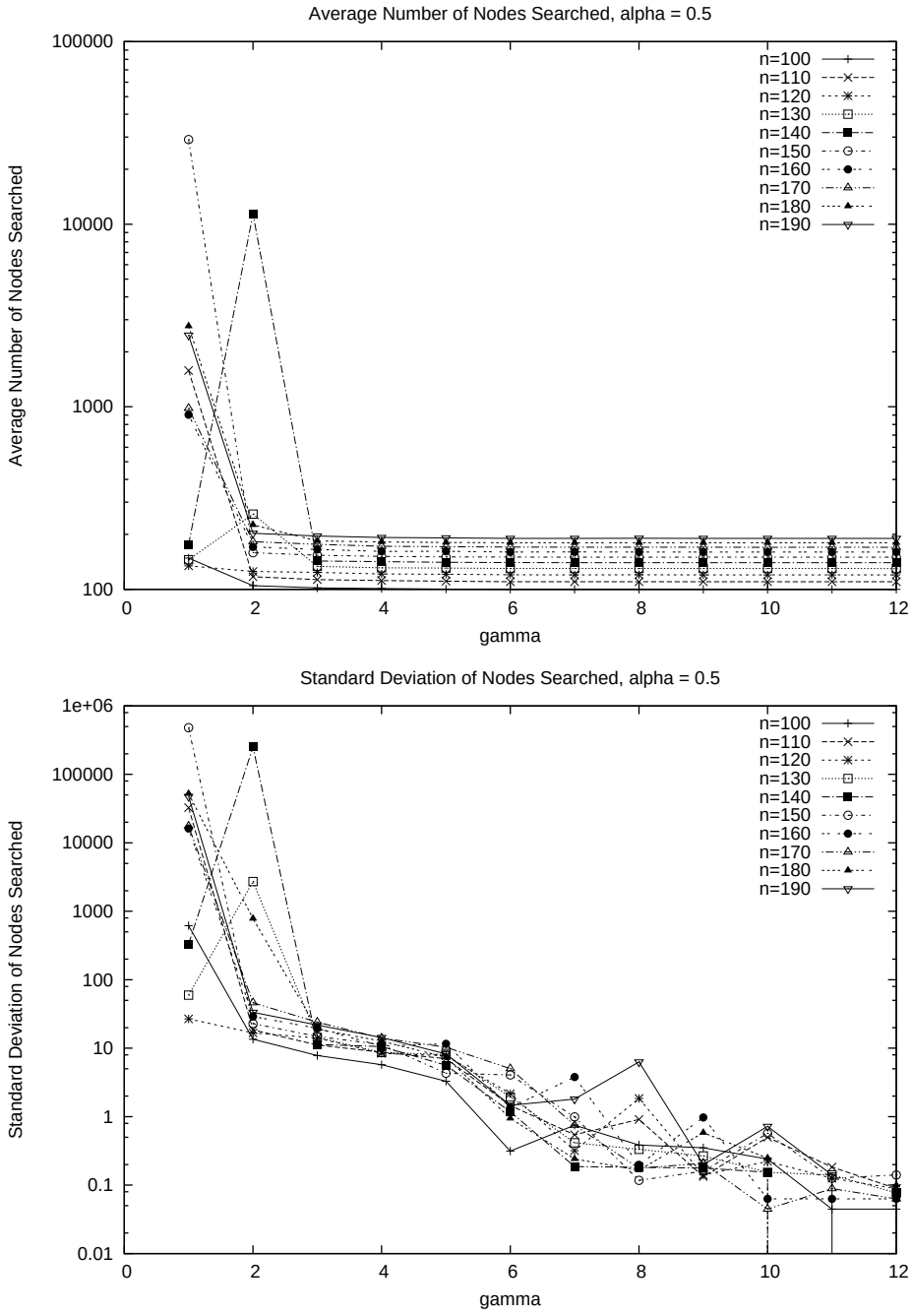


Figure 4.10: Average number of nodes searched for $\alpha = 0.5$ and $N \in \{100, \ldots, 190\}$ when a Hamiltonian cycle is explicitly put in. The average number of nodes searched and the standard deviation are plotted on a semi-log plot as a function of the scale parameter γ. The sporadic jumps are indicative of the rarity of finding "hard" instances. Each point represents the average of 500 simulation runs.

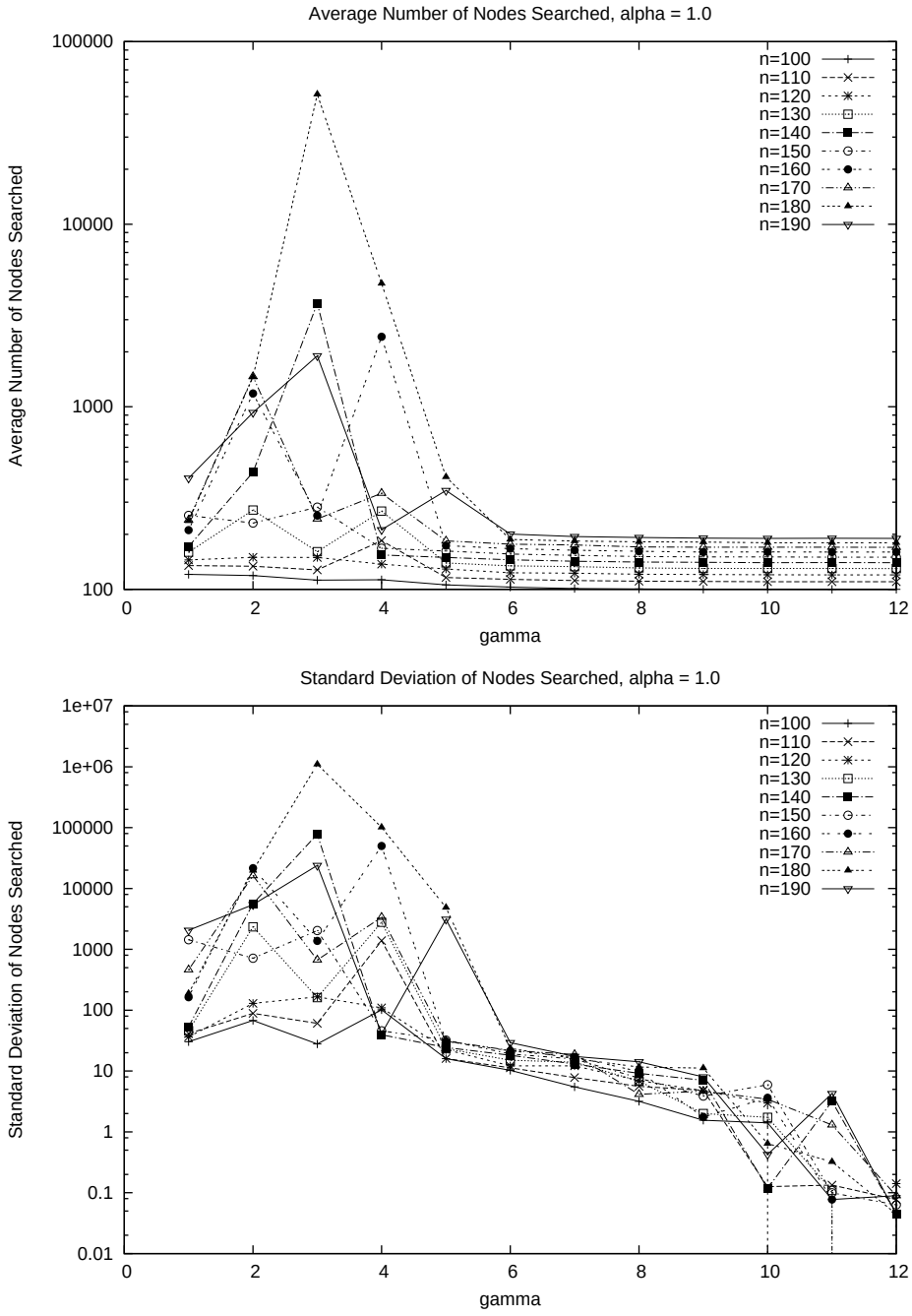


Figure 4.11: Average number of nodes searched for $\alpha = 1.0$ and $N \in \{100, \ldots, 190\}$ when a Hamiltonian cycle is explicitly put in. The average number of nodes searched and the standard deviation are plotted on a semi-log plot as a function of the scale parameter γ. The sporadic jumps are indicative of the rarity of finding "hard" instances. Each point represents the average of 500 simulation runs.

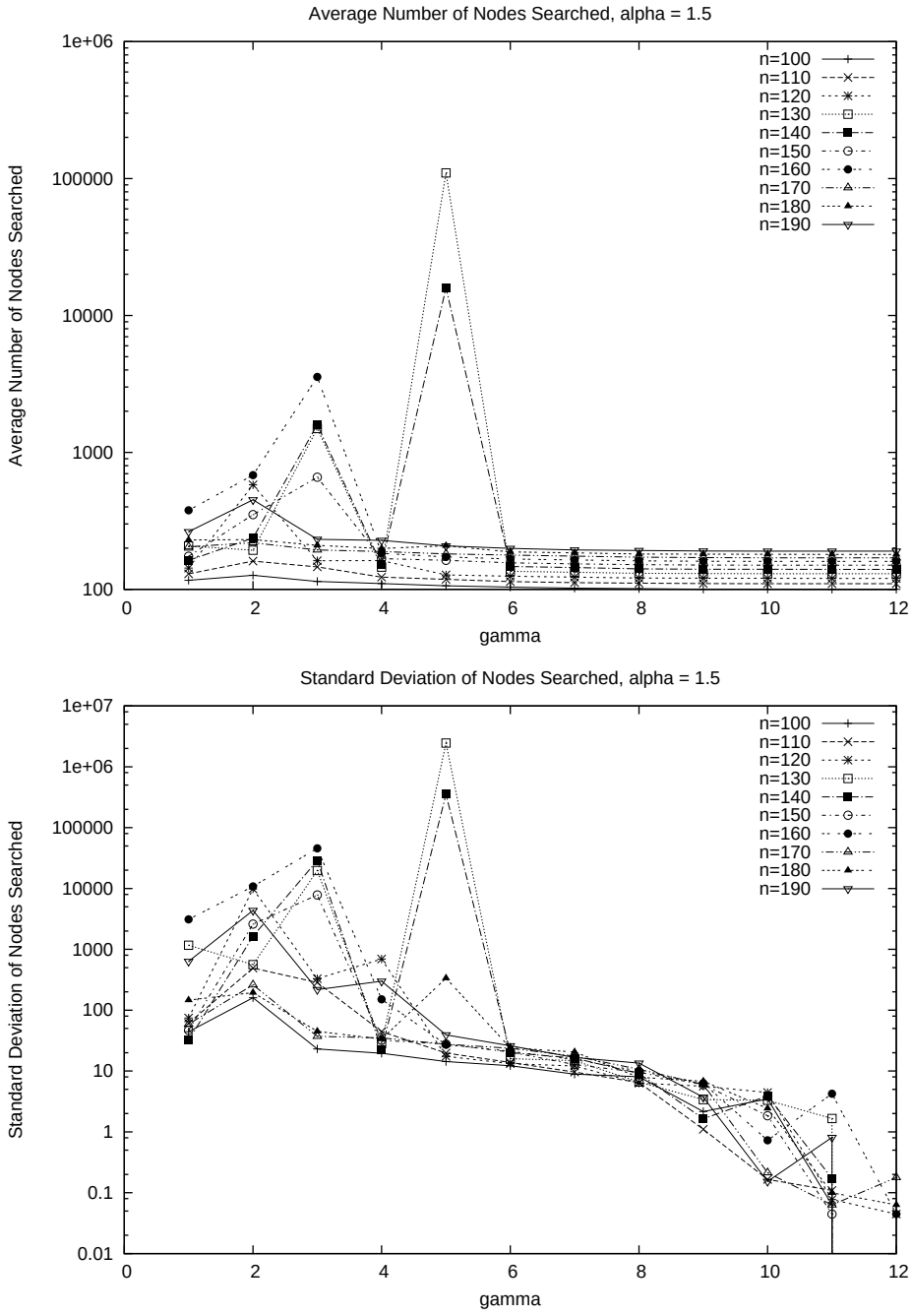


Figure 4.12: Average number of nodes searched for $\alpha = 1.5$ and $N \in \{100, \ldots, 190\}$ when a Hamiltonian cycle is explicitly put in. The average number of nodes searched and the standard deviation are plotted on a semi-log plot as a function of the scale parameter γ. The sporadic jumps are indicative of the rarity of finding "hard" instances. Each point represents the average of 500 simulation runs.

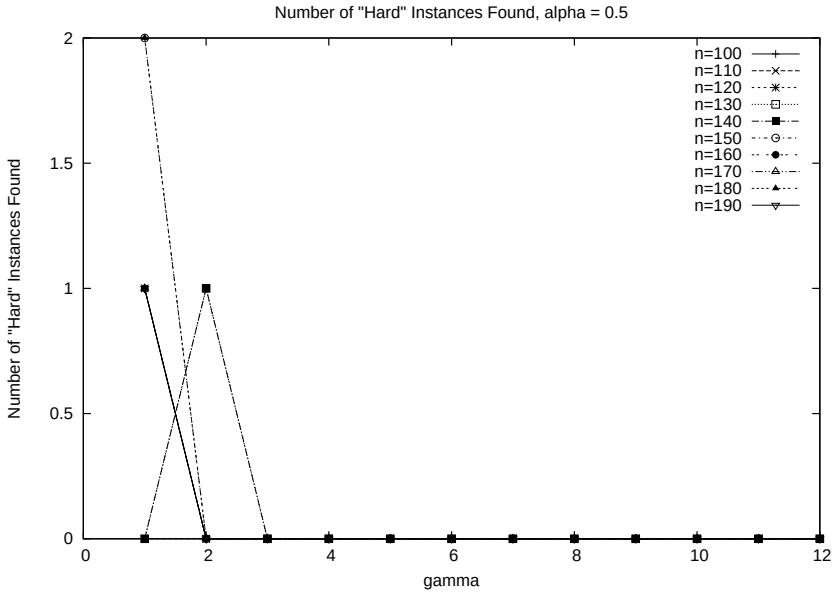


Figure 4.13: The number of instances found where nodes searched in the **FindHamCycleComplete** algorithm exceeded N^2 for $\alpha = 0.5$ as a function of γ for graphs with a Hamiltonian cycle explicitly put in. Often just one of these hard instances inflate the average and standard deviation of the nodes searched.

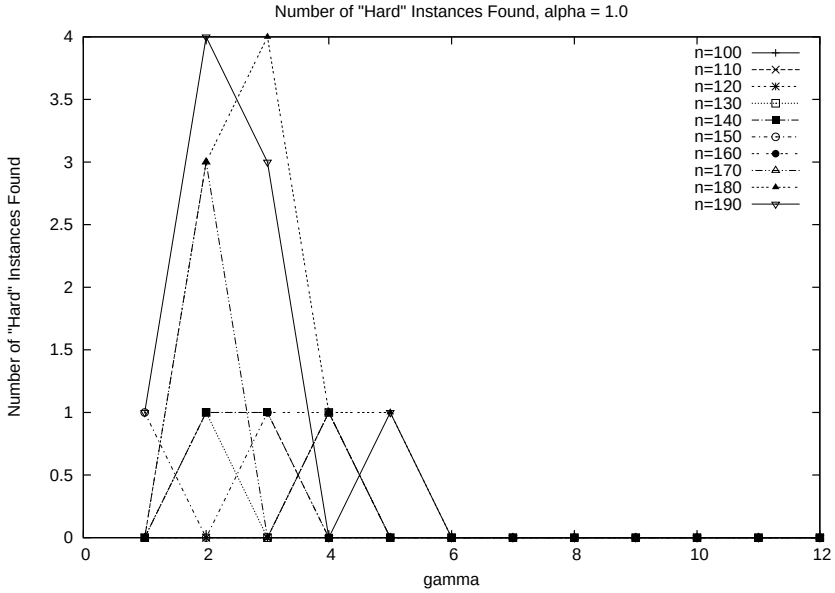


Figure 4.14: The number of instances found where nodes searched in the **FindHamCycleComplete** algorithm exceeded N^2 for $\alpha = 1.0$ as a function of γ for graphs with a Hamiltonian cycle explicitly put in. Often just one of these hard instances inflate the average and standard deviation of the nodes searched.

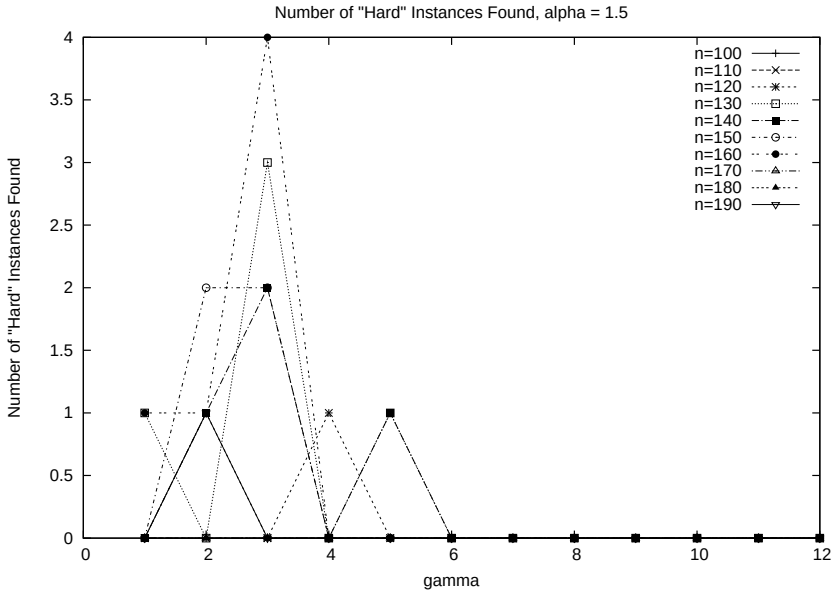


Figure 4.15: The number of instances found where nodes searched in the **FindHamCycleComplete** algorithm exceeded N^2 for $\alpha = 1.5$ as a function of γ for graphs with a Hamiltonian cycle explicitly put in. Often just one of these hard instances inflate the average and standard deviation of the nodes searched.

A maximum of 10 restarts is used with a maximum number of iterations (per restart) of N^2. Each point represents 1000 simulation points. The graphs were generated from a symmetric, capped, floored and absolute-valued Lévy-Stable distribution for $\alpha \in (0.5, 1.0, 1.5)$. Plots are shown for the probability of the **Pósa** algorithm finding a Hamiltonian Cycle versus the scale parameter, $\gamma \in (1, 2, 3, \ldots, 32)$. Note that there is a selection bias, as graphs with a Hamiltonian Cycle that are not found by the **Pósa** algorithm do not count towards the probability, deflating the probability curve shown. Figures 4.16, 4.17 and 4.18 show the results of these simulations.

Run time for the **Pósa** heuristic is also shown for graphs that have a Hamiltonian Cycle found. As this probabilistic algorithm only checks for Hamiltonian Cycles and has no facility to look for features which would preclude the graph from being Hamiltonian, the number of iterations for non-Hamiltonian graphs is necessarily the maximum iteration count and thus are excluded in these plots. It should be re-emphasize that the run-times are capped at N^2 by construction for the **Pósa** algorithm used.

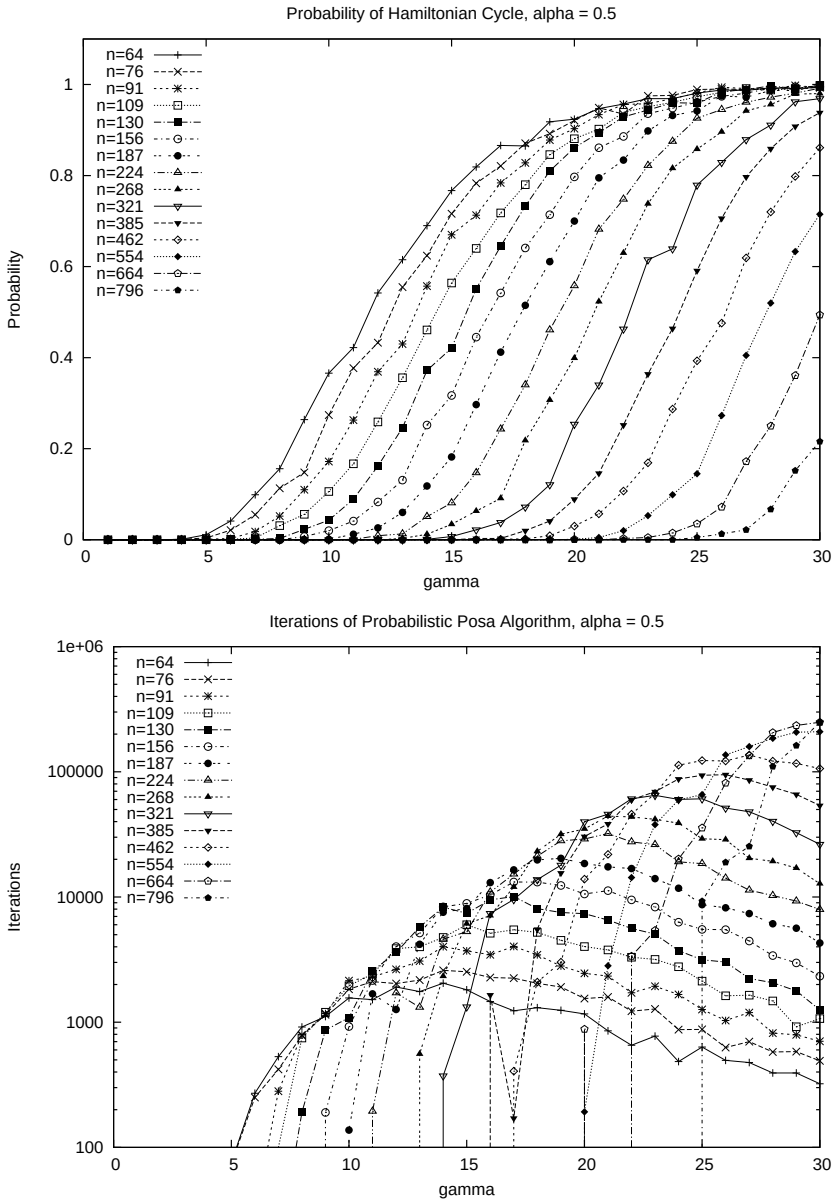


Figure 4.16: Probability and average nodes searched for $\alpha = 0.5$ for the Probabilistic **Pósa** Algorithm. Average nodes searched are only representative of graphs in which a Hamiltonian Cycle was found. Each γ point represents 1000 graphs produced. Iterations are shown on a semi-log plot.

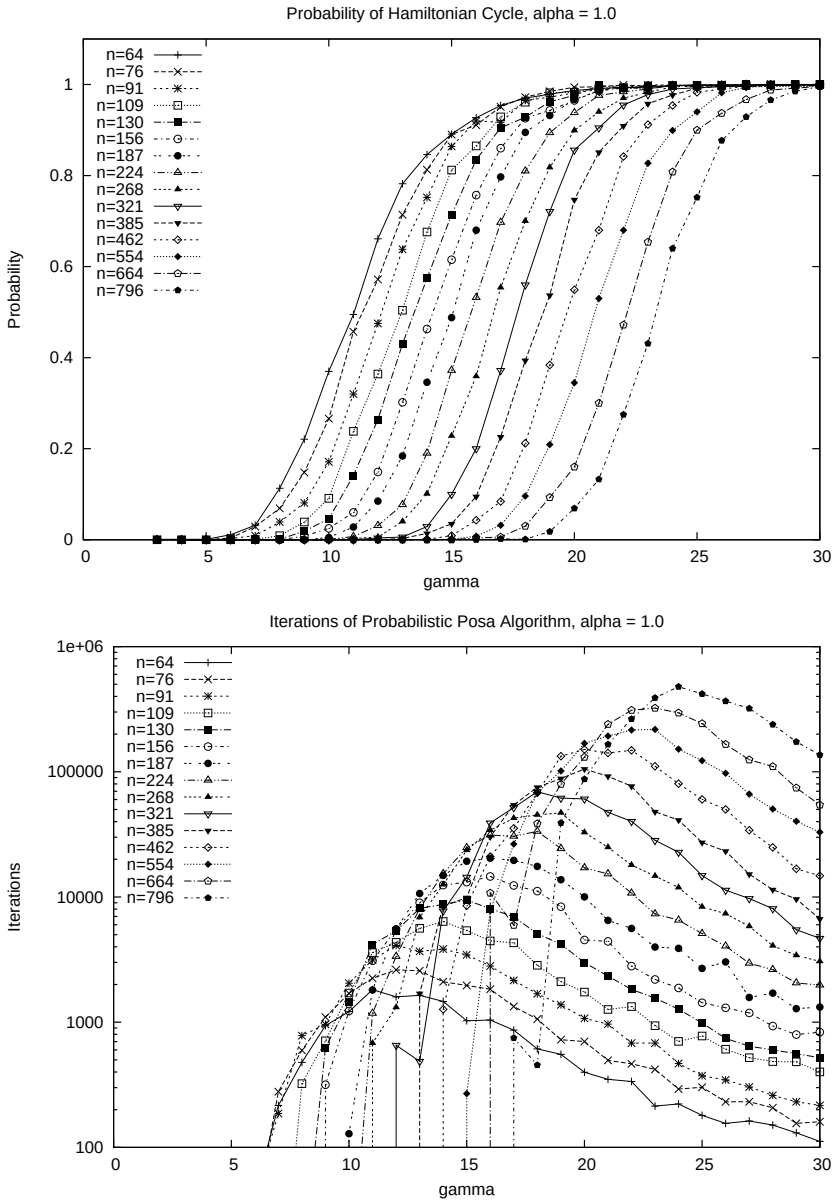


Figure 4.17: Probability and average nodes searched for $\alpha = 1.0$ for the Probabilistic **Pósa** Algorithm. Average nodes searched are only representative of graphs in which a Hamiltonian Cycle was found. Each γ point represents 1000 graphs produced. Iterations are shown on a semi-log plot.

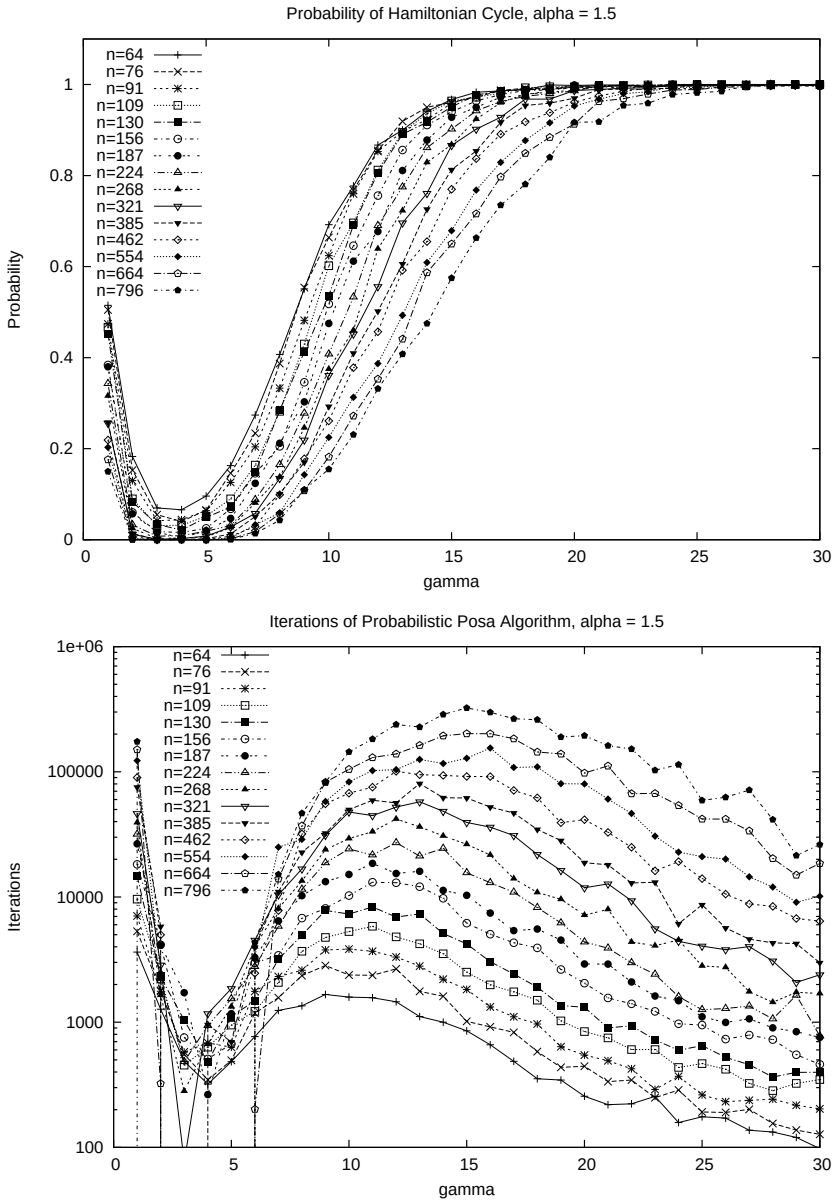


Figure 4.18: Probability and average nodes searched for $\alpha = 1.5$ for the Probabilistic **Pósa** Algorithm. Average nodes searched are only representative of graphs in which a Hamiltonian Cycle was found. Each γ point represents 1000 graphs produced. Iterations are shown on a semi-log plot.

4.4 The Initial Probability Peak

This section will attempt to briefly describe why there is an initial peak in the probability of finding a Hamiltonian Cycle for the numerical results presented in the previous sections.

The increase in probability for the $\gamma < 5$ region for Figures 4.18, 4.2 and 4.3 is most likely due to the method of graph construction. More detail will be provided below, but briefly, because only graphs with a minimum of degree 2 with a connected component of size N are chosen and because α is so high, this generates graphs without many large degree vertices compared with smaller degree vertices, giving graphs in the low γ region a higher chance of having a Hamiltonian Cycle. The reason for this effect will need a better theory behind it to describe in full, but in what follows, an attempt is made to provide a brief description of part of why this might be happening.

Graphs whose degree sequence are generated with low α tend to have higher degree vertices chosen. Lower α corresponds to a probability distribution function that has fatter tails and so larger values tend to be picked more often. Graphs that have a few large degree vertices mixed with degree 2 vertices tend to form "windmills", where degree 2 vertices connect to vertices of larger degree, or "hubs", in the graph. If more than two such degree 2 vertices connect to the same hubs, this precludes any Hamiltonian Cycle from occurring. This is most pronounced in graphs of low α and low γ, where the probability of finding a Hamiltonian Cycle is near 0. For graphs with low γ but high α, the frequency of large degree vertices that soak up the endpoints of degree 2 vertices is not as pronounced as in the low α case.

For the higher α case and when $\gamma < 5$, there are few high degree vertices. By graph construction, a minimum degree hint has been provided and graphs of minimum degree 2 with a full connected component are only considered. All these effects combined select for graphs that are more likely to be Hamiltonian in the high α and low γ case. For larger α and as γ increases, the probability of finding few higher degree vertex that will "grab" degree 2 vertices endpoints becomes more likely until γ becomes so large so as to nearly ensure a Hamiltonian Cycle.

Consider the probability of finding a "windmill" in Figure 4.19: A large degree vertex surrounded by more than two degree 2 vertices. There is a peak at 3, corresponding to the low point in the probability of finding a Hamiltonian Cycle in Figure 4.18. As further evidence, consider Figure 4.20, 4.21, 4.22 and 4.23 that show the average degree, average minimum degree, average maximum degree and average standard deviation for the degrees of graphs generated by this method for $\alpha = 1.5$. For this region, a low mean degree with a low standard deviation makes it look more like an Erdös-Rényi

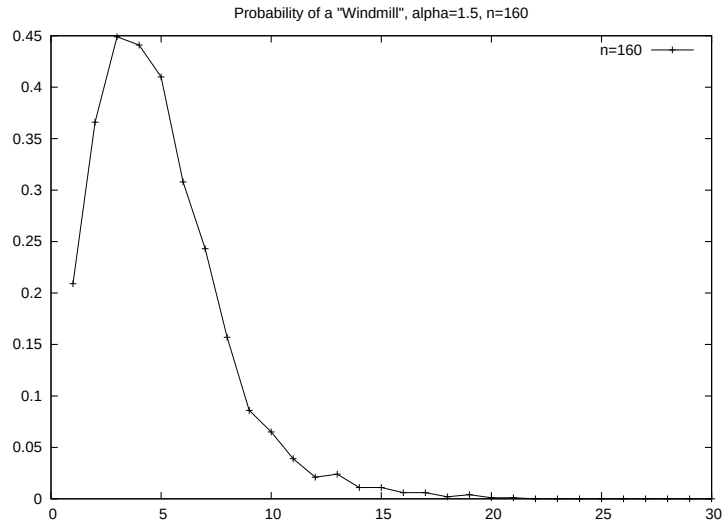


Figure 4.19: Probability of a more than two degree 2 vertices joined to the same set of vertices creating a "windmill" and precluding a Hamiltonian Cycle from occurring. The minimum degree hint used in the **GenerateApproxDegSequenceGraph** and the high α reduce the probability of high degree vertices initially appearing for low γ, giving the probability a transient "bump". Each point represents the result of averaging 1000 simulations.

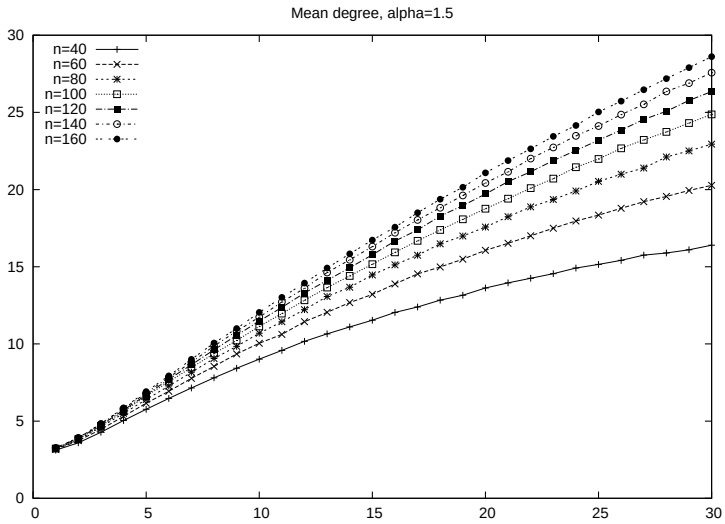


Figure 4.20: The average degree of graphs generated by the **GenerateApproxDegSequence-Graph** where the degree distribution was chosen from a symmetric, absolute valued truncated Lévy-Stable distribution with a minimum degree hint of 3, $N \in \{40, 60, 80, 100, 120, 140, 160\}$, $\alpha = 1.5$ as a function of γ, the scale parameter. Each point represents the result of averaging 1000 simulations.

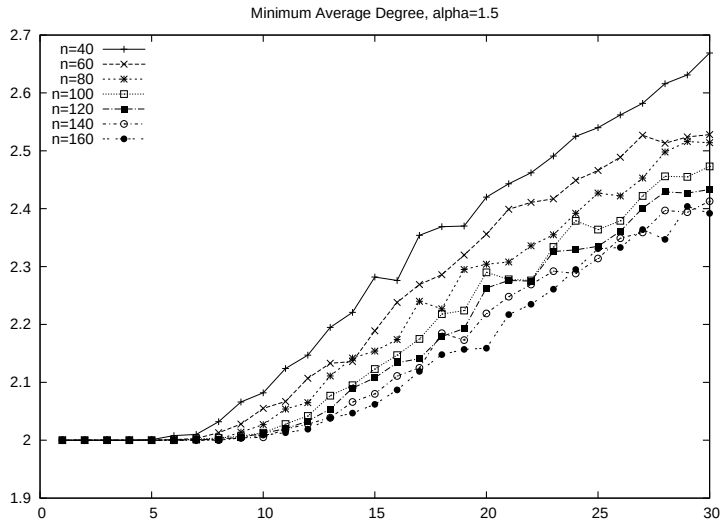


Figure 4.21: The average minimum degree of graphs generated by the **GenerateApproxDegSequenceGraph** where the degree distribution was chosen from a symmetric, absolute valued truncated Lévy-Stable distribution with a minimum degree hint of 3, $N \in \{40, 60, 80, 100, 120, 140, 160\}$, $\alpha = 1.5$ as a function of γ, the scale parameter. Each point represents the result of averaging 1000 simulations.

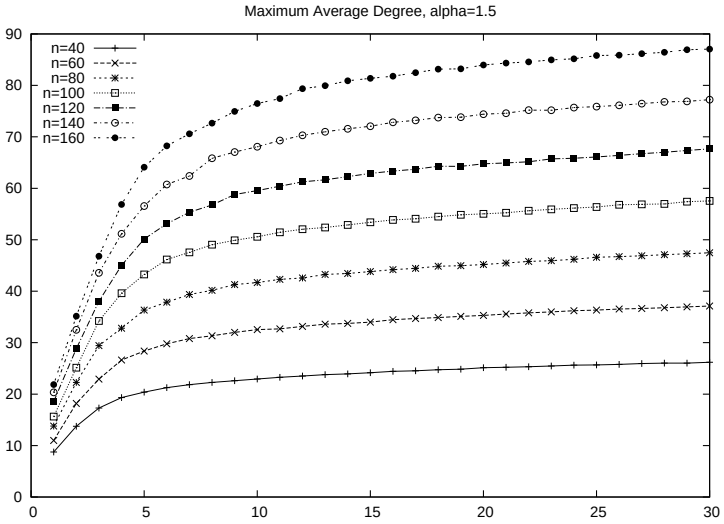


Figure 4.22: The average maximum degree of graphs generated by the **GenerateApproxDegSequenceGraph** where the degree distribution was chosen from a symmetric, absolute valued truncated Lévy-Stable distribution with a minimum degree hint of 3, $N \in \{40, 60, 80, 100, 120, 140, 160\}$ and $\alpha = 1.5$ as a function of γ, the scale parameter. Each point represents the result of averaging 1000 simulations.

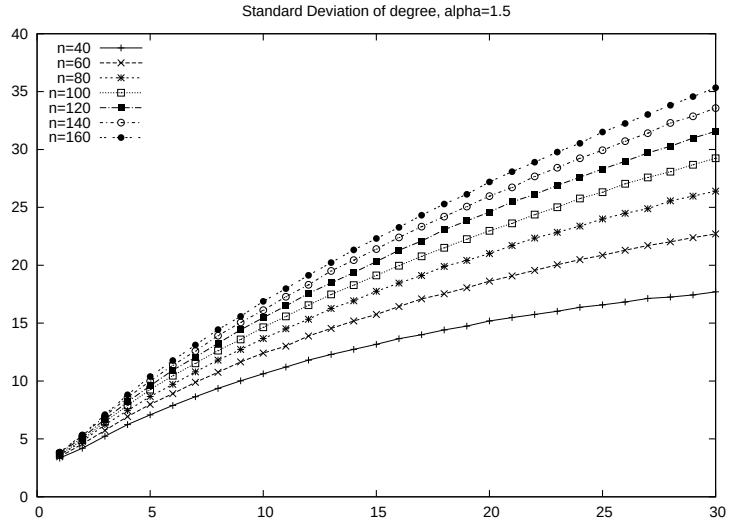


Figure 4.23: The average standard deviation of degree for graphs generated by the **GenerateApproxDegSequenceGraph** where the degree distribution was chosen from a symmetric, absolute valued truncated Lévy-Stable distribution with a minimum degree hint of 3, $N \in \{40, 60, 80, 100, 120, 140, 160\}$ and $\alpha = 1.5$ as a function of γ, the scale parameter. Each point represents the result of averaging 1000 simulations.

graph with high edge probability, giving a transient increase in probability of finding a Hamiltonian Cycle. As γ is increased, the standard deviation of the degrees chosen increases allowing for larger degree vertices to be more likely to be chosen. There are surely more complicated effects at play and this explanation is presented as an indication that the initial dip in probability is most likely only an unfortunate side effect of graph construction and nothing fundamental to the phase transition under discussion.

4.5 Pitfalls

It needs to be stressed that the above analysis depends critically on the algorithms used. The class of random graphs presented could well turn out to be easily solvable with a better algorithm that exploits knowledge of the degree distribution. While graphs with a finite variance for their degree distribution are easy, this does not imply that graphs generated with diverging variance, either by the method presented or some other, are difficult. The above class of random graphs is presented as only a potential candidate for generating intrinsically hard instances of the Hamiltonian Cycle Problem.

The **Pósa** algorithm has a potential problem of getting into cycles of rotation choices that preclude

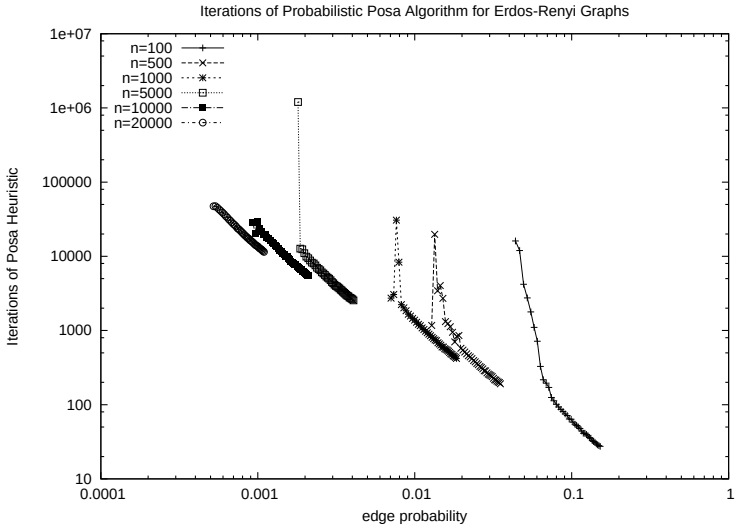


Figure 4.24: Run times of the Algorithm that uses the **Pósa** algorithm when run on Erdös-Rényi random graphs for vertex count $N \in \{100, 500, 1000, 5000, 10000, 20000\}$ where a Hamiltonian Cycle was found by the algorithm. Iterations are plotted on a log-scale for convenience of visualization.

it from ever finding a solution. This has the effect of increasing run times of the algorithm that uses the Pósa algorithm as the only way to find a Hamiltonian Cycle after this algorithm has wandered into a cycle is to restart with a different initial vertex. For comparison, Figure 4.24 shows run times of the **Pósa** algorithm run on Erdös-Rényi random graphs for varying edge probability of vertex count $N \in \{100, 500, 1000, 5000, 10000, 20000\}$. Edge probability has been chosen as $(N/2)(\ln(N) + \ln(\ln(N)) + c)$, where c is the varying parameter. Iteration count has been plotted on a log-scale for convenience of visualization.

From figure 4.24, we can see that the **Pósa** algorithm is perhaps not ideal for Erdös-Rényi graphs as there is a pickup in search cost near the critical threshold. This pickup in search cost disappears when larger graphs are generated and so is perhaps transient, appearing only on these relatively small graph instances. Also from figure 4.24, we can see that the pickup in search cost is only very near the critical probability and the region of difficult instances found for the **Pósa** algorithm decreases as graph vertex count is increased.

For comparison, the Pósa algorithm has been run on Hamiltonian graphs discovered via a complete search in section 4.2. Figures 4.25, 4.26 and 4.27 show run times in terms of number of rotations for $\alpha = \{0.5, 1.0, 1.5\}$ respectively. Figures 4.28, 4.29 and 4.30 show the number and percentages of graphs that were dropped by the Pósa algorithm. Run times do not include dropped graphs.

Figures 4.31, 4.32 and 4.33 show the run times of the Pósa run on graphs whose search was incomplete

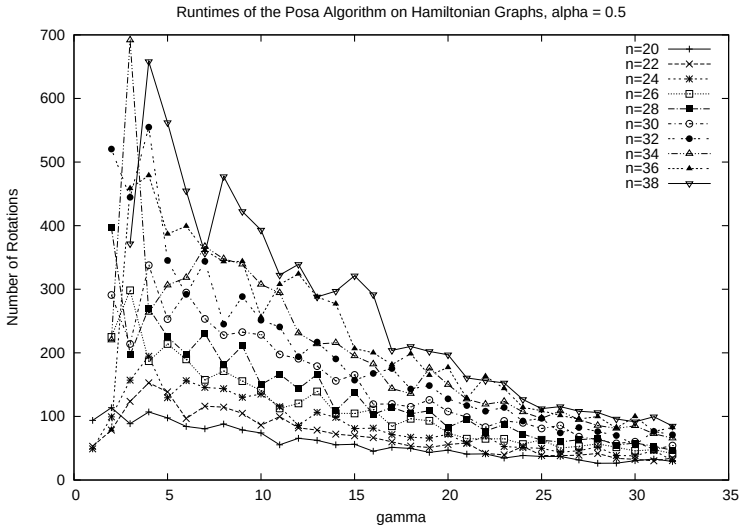


Figure 4.25: Run times of the **Pósa** algorithm when run on graphs found with a Hamiltonian Cycle in section 4.2 for $\alpha = 0.5$.

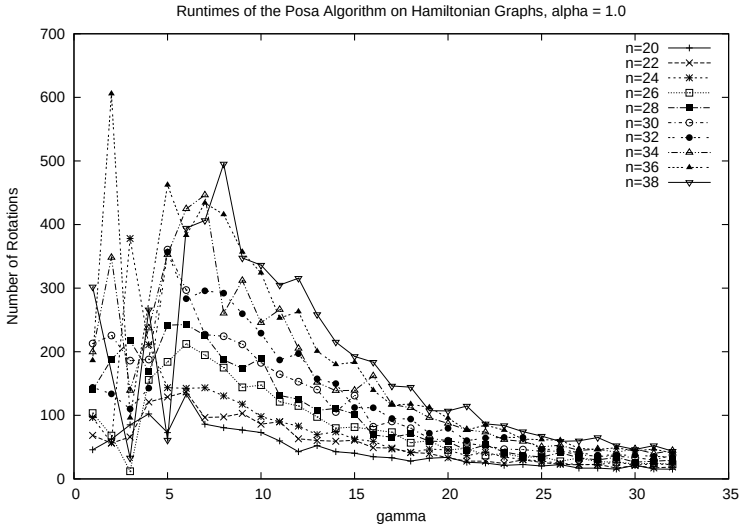


Figure 4.26: Run times of the **Pósa** algorithm when run on found with a Hamiltonian Cycle in section 4.2 for $\alpha = 1.0$.

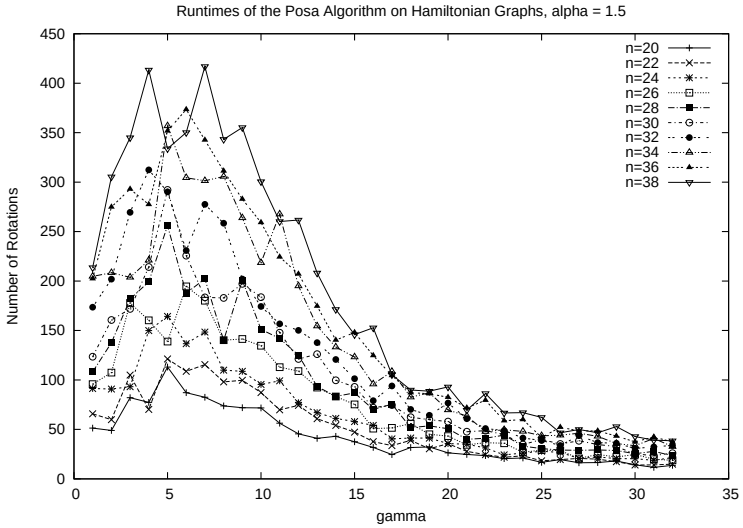


Figure 4.27: Run times of the **Pósa** algorithm when run on found with a Hamiltonian Cycle in section 4.2 for $\alpha = 1.5$.

in section 4.2 for $\alpha \in \{0.5, 1.0, 1.5\}$ respectively. Figures 4.34, 4.34 and 4.34 show the number and percentage of graphs that were dropped by the Pósa algorithm for $\alpha = \{0.5, 1.0, 1.5\}$ respectively. Graphs that were dropped by Culberson and Vandegriend's algorithm for a threshold of N^2 but were found by the Pósa algorithm are shown in figure 4.37, 4.38 and 4.39 for $\alpha \in \{0.5, 1.0, 1.5\}$ respectively.

From figures 4.25 to 4.39 one can see that the **Pósa** algorithm is a reasonable comparable choice for to Culberson and Vandegriend's algorithm as most Hamiltonian cycles are found near the critical threshold. As the number of graphs with a Hamiltonian cycle become diminish, the data for the **Pósa** algorithm dropped percentages and run times becomes more chaotic as the sample size becomes so small. As can be seen, there are cases where the **Pósa** algorithm finds solutions whereas Culberson and Vandegriend's algorithm do not. There are also cases where Culberson and Vandegriend's algorithm find solutions where the **Pósa** algorithm do not. One might expect this behavior from these two algorithms as they exploit different features of the graph in their search.

These problems with the search algorithm using the **Pósa** algorithm do not make it ideal. The search algorithm using the **Pósa** algorithm was used because of its ability to probe large graphs for Hamiltonicity, its simplicity in implementation and speed for the large graphs considered. Statements about the intrinsic difficulty of finding a Hamiltonian Cycle in graphs using the **Pósa** algorithm should be used with a note of caution. With these drawbacks in mind, the **Pósa** algorithm provides a view into the difficulty of finding Hamiltonian Cycles is graphs chosen from the modified Lévy-

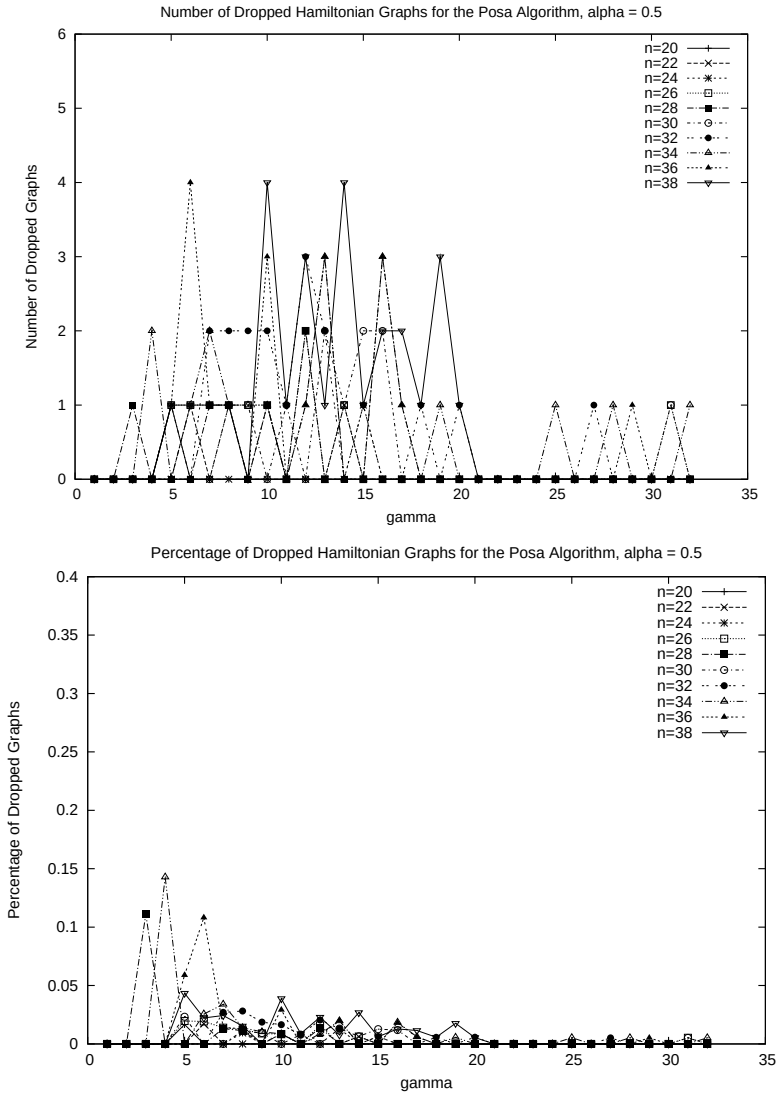


Figure 4.28: Number and percentage of graphs dropped when running the Pósa algorithm for graphs found with a Hamiltonian Cycle in section 4.2 for $\alpha = 0.5$.

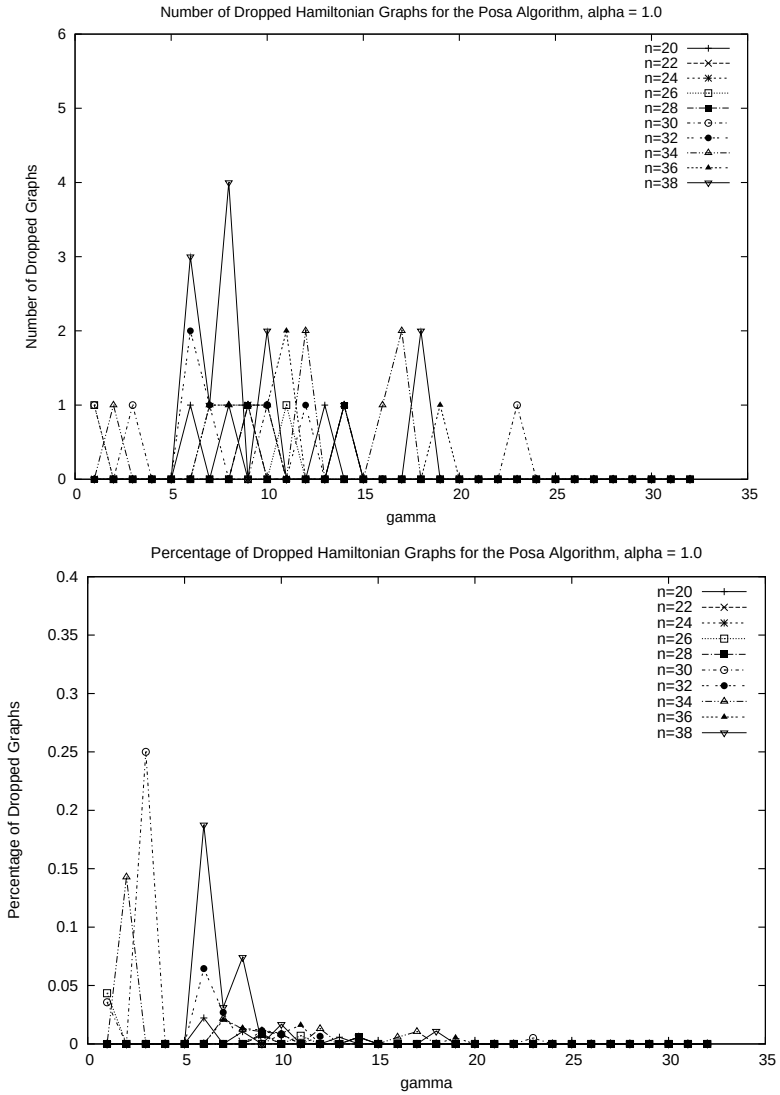


Figure 4.29: Number and percentage of graphs dropped when running the Pósa algorithm for graphs found with a Hamiltonian Cycle in section 4.2 for $\alpha = 1.0$.

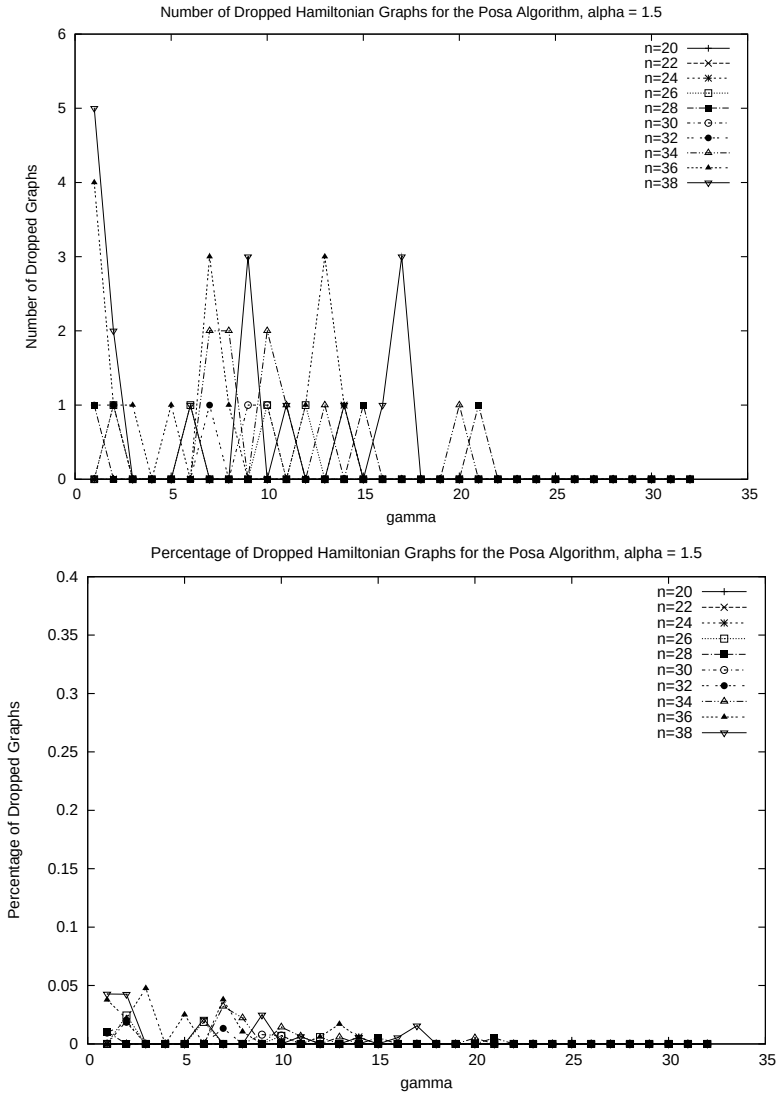


Figure 4.30: Number and percentage of graphs dropped when running the Pósa algorithm for graphs found with a Hamiltonian Cycle in section 4.2 for $\alpha = 1.5$.

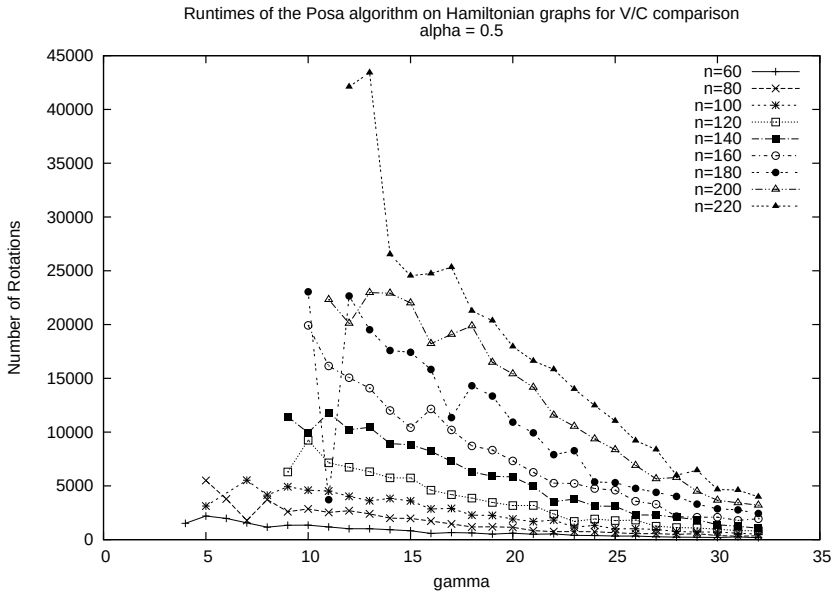


Figure 4.31: Run times of the **Pósa** algorithm when run on graphs that were dropped or found to be Hamiltonian in section 4.2 for $\alpha = 0.5$.

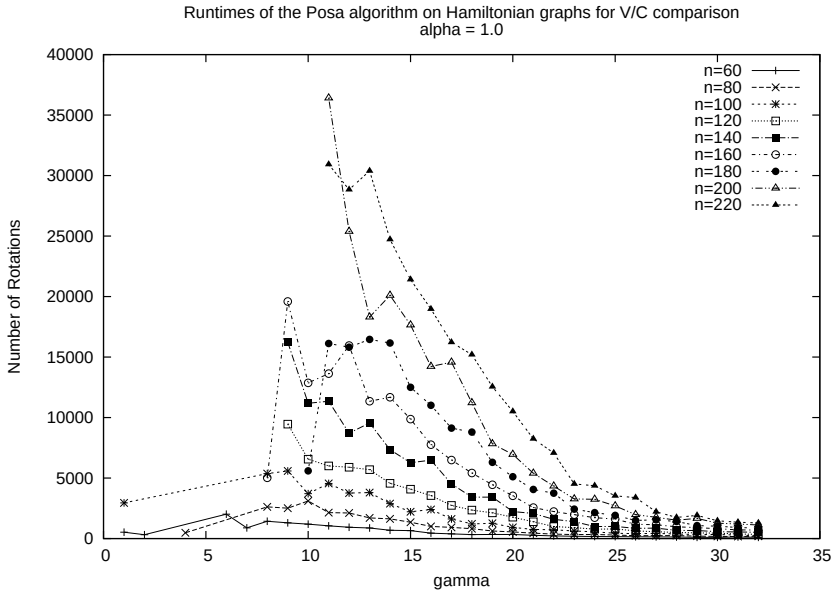


Figure 4.32: Run times of the **Pósa** algorithm when run on graphs that were dropped or found to be Hamiltonian in section 4.2 for $\alpha = 1.0$.

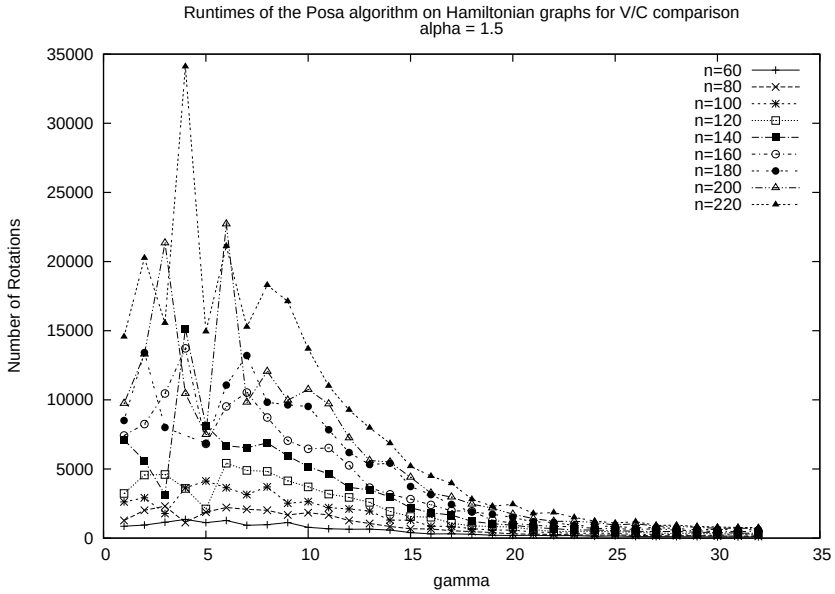


Figure 4.33: Run times of the **Pósa** algorithm when run on graphs that were dropped or found to be Hamiltonian in section 4.2 for $\alpha = 1.5$.

Stable distribution in question and provides at least an initial starting point for further investigation.

It should also be mentioned that no attempt was made at discovering what the order parameter is for the phase transition of Hamiltonian Cycles drawn from the class of graphs whose degree sequence is chosen from a modified Lévy-Stable distribution. It is unclear to the author exactly what form the order parameter should take and feels it would not be fruitful to guess until a better theory is developed. The form of phase transition appears to be Gumbel, as it is with Erdös-Rényi random graphs, and there is evidence to suggest that any monotone graph feature has this type of phase transition behavior ([49]) on the graph ensemble considered. It is left for future work to determine what the dependence on α, γ and N the order parameter for this graph ensemble has.

4.6 Conclusion

Erdös-Rényi graphs are locally 'tree-like' ([81]), in that there is a low probability of finding a path that loops back of length less than $\ln(N)$. This property might be the reason why efficient algorithms exist, allowing them to make local progress with early back-track detection. The diverging second moment of graphs whose degree sequences are drawn from a modified Lévy-Stable distribution effectively destroy this local 'tree-like' structure, enjoying, rather, an extremely short average diameter on the order of $\ln\ln(N)$ or smaller ([101], [102], [103]).

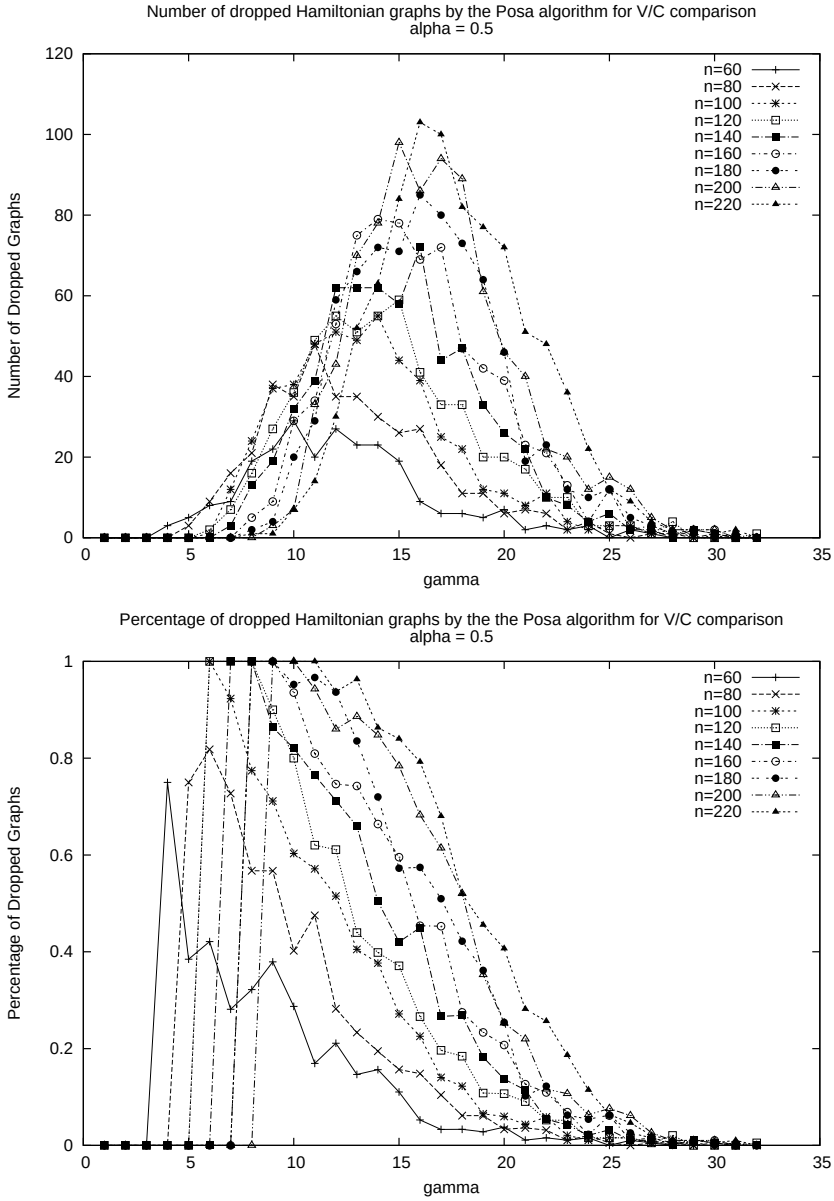


Figure 4.34: Number and percentage of graphs dropped when running the Pósa algorithm for graphs found to be Hamiltonian from section 4.2 with Culberson and Vandegriend's algorithm where the threshold was set to N^2. The above graphs were generated with $\alpha = 0.5$.

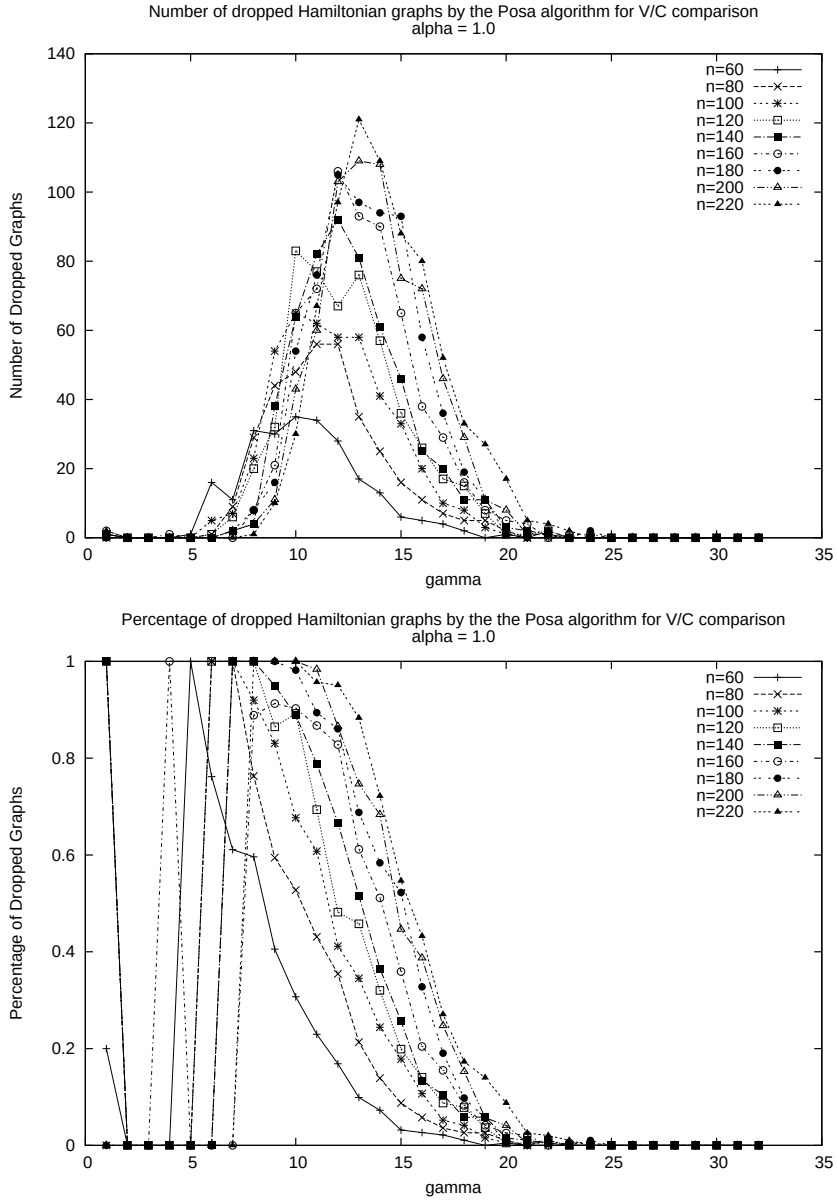


Figure 4.35: Number and percentage of graphs dropped when running the Pósa algorithm for graphs found to be Hamiltonian from section 4.2 with Culberson and Vandegriend's algorithm where the threshold was set to N^2. The above graphs were generated with $\alpha = 1.0$.

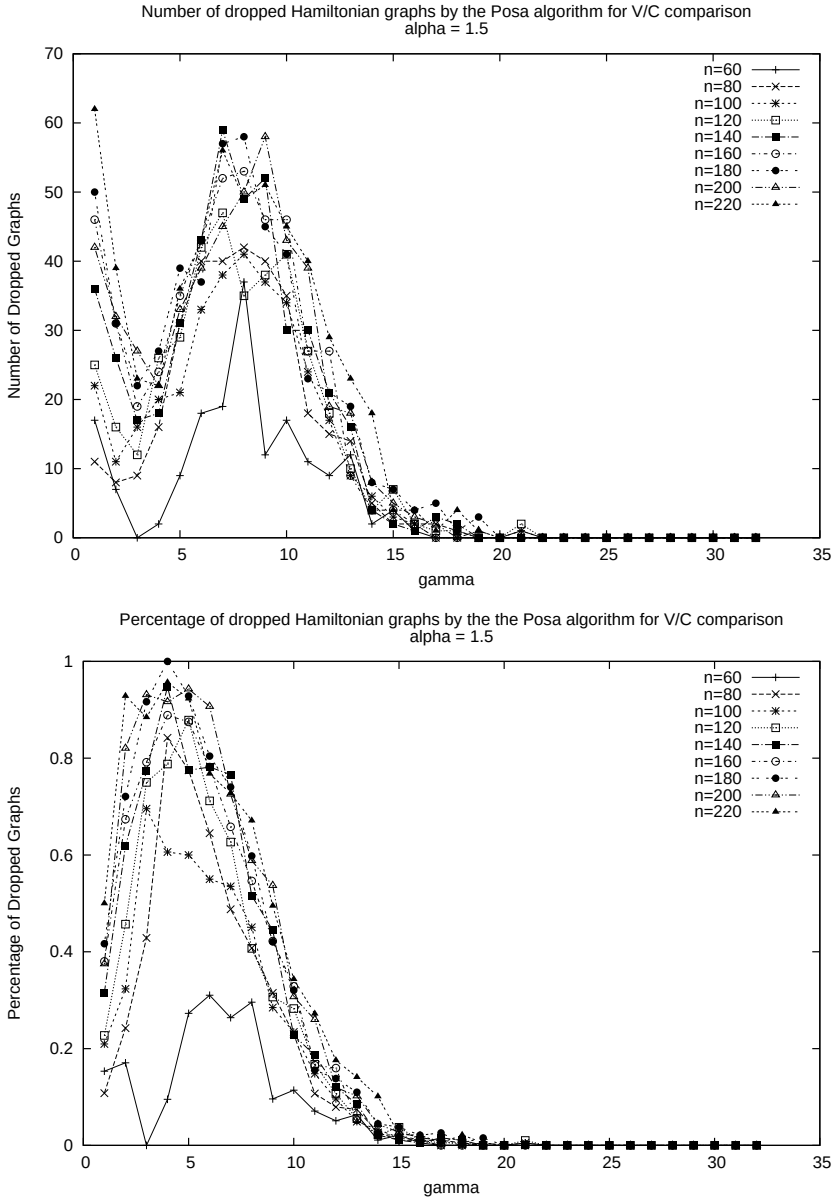


Figure 4.36: Number and percentage of graphs dropped when running the Pósa algorithm for graphs found to be Hamiltonian from section 4.2 with Culberson and Vandegriend's algorithm where the threshold was set to N^2. The above graphs were generated with $\alpha = 1.5$.

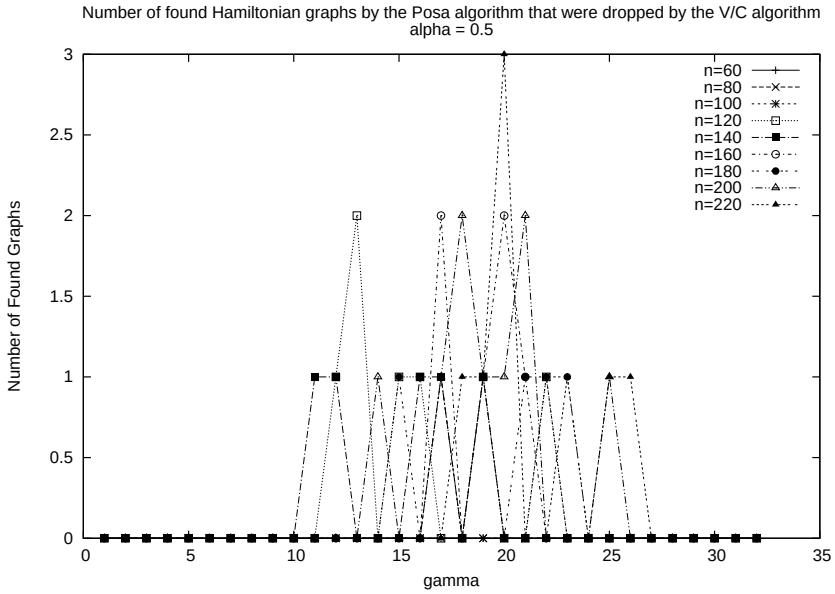


Figure 4.37: Number of graphs whose Hamiltonian Cycle was found by the Pósa algorithm but that were not found by Culberson and Vandegriend's algorithm with a threshold set to N^2 in section 4.2. $\alpha = 0.5$ and the threshold was set to N^2 for the Culberson and Vandegriend's algorithm. 200 graphs total were generated for each N, α and γ combination when originally run against Culberson and Vandegriend's algorithm.

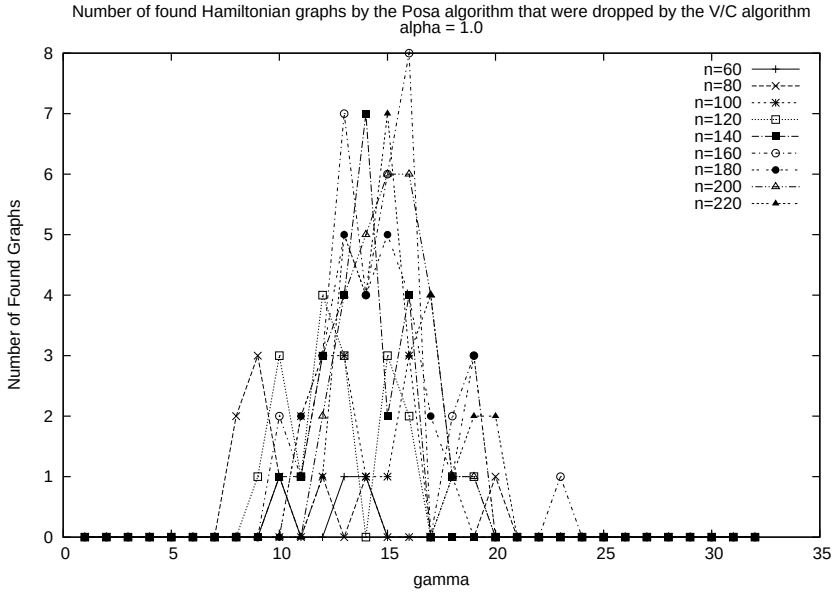


Figure 4.38: Number of graphs whose Hamiltonian Cycle was found by the Pósa algorithm but that were not found by Culberson and Vandegriend's algorithm with a threshold set to N^2 in section 4.2. $\alpha = 1.0$ and the threshold was set to N^2 for the Culberson and Vandegriend's algorithm. 200 graphs total were generated for each N, α and γ combination when originally run against Culberson and Vandegriend's algorithm.

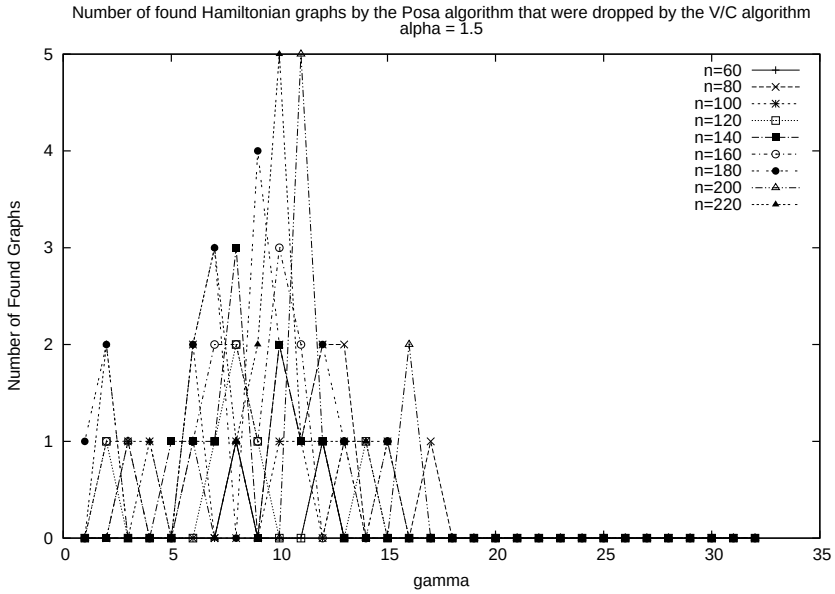


Figure 4.39: Number of graphs whose Hamiltonian Cycle was found by the Pósa algorithm but that were not found by Culberson and Vandegriend's algorithm with a threshold set to N^2 in section 4.2. $\alpha = 1.5$ and the threshold was set to N^2 for the Culberson and Vandegriend's algorithm. 200 graphs total were generated for each N, α and γ combination when originally run against Culberson and Vandegriend's algorithm.

Numerical evidence has been provided that points to a good candidate for random graph generation that produces intrinsically hard instances of the Hamiltonian Cycle Problem. This numerical evidence, at least initially, gives a good candidate for producing graphs whose Hamiltonicity is not trivially found.

The numerical evidence supports the claim that there is a phase transition in the class of graphs chosen. The form of the transition also appears to be Gumbel, as it is for the Erdös-Rényi graph ensemble, but the form of the parametrization is unknown to the author. Without a better theory, it is difficult to infer what the proper parametrization should be from the above data. Instead, the data is provided as numerical evidence to suggest a potential class of graphs that are intrinsically hard. Future work could work towards determining the form of the parametrization and the dependence it has on the parameters of the Lévy-Stable distributions in question.

Chapter 5

Phase Transition for NPP

In this section numerical results will be presented for the probability of a perfect partition existing for small instances of the Number Partition Problem where instances are drawn from a truncated Lévy-Stable distribution. To the author's knowledge, no work has been done in analyzing the phase transition of the Number Partition Problem when instance elements are generated from a Lévy-Stable distribution.

Two heuristic arguments are presented, one analytic and one numerical, that motivate the choice of order parameter used. This chapter ends with a brief discussion on failed attempts at using lattice reduction techniques to solve instances of the Number Partition Problem in the region where solutions are almost sure to exist.

5.1 Introduction and Motivation

A commonly used formulation of the Number Partition Problem is to find a perfect partition of a list of integers. i.e. Given:

$$a_k \in \mathbb{N}, \quad k \in [0 \dots (n-1)]$$

Find $\sigma_k \in \{-1, 1\}$ such that:

$$|\sum_{k=0}^{n-1} \sigma_k a_k| \leq 1, \quad \sigma_k \in \{-1, 1\}$$

The Random Number Partition Problem is one in which each a_k is chosen from some distribution. Gent and Walsh [53] first noticed a phase transition in the probability of finding a solution when each entry is chosen from a uniform distribution in some range. In this context, a phase transition indicates a rapid change from there being almost surely no solution to a region where a solution is almost sure to exist.

Consider an instance of the Number Partition Problem where each entry is chosen from the range $[1 \ldots 2^m]$ uniformly at random. Gent and Walsh gave a heuristic argument by considering the random sum $\sum_{k=0}^{n-1} \sigma_k a_k$, when σ_k are chosen to be 1 or -1 with equal probability, and noticed that this would be a perfect partition with a probability approximately 2^{-m}. There are a total of 2^n configurations giving an expected number of solutions as $2^n/2^m$. When this fraction representing the total number of solutions is less than 1, we would expect there to be no solutions, whereas if this number is very much larger than 1, we would expect there to be an abundance of solutions.

Mertens [79] later used a heuristic based on the saddle point method to approximate the integral representation of the Number Partition Problem to refine the order parameter. Later Borgs, Chayes and Pittel [23] made this method more rigorous and gave exact analytic results for the (Uniform) Random Number Partition Problem. It is the author's understanding that this is the only NP-Complete problem whose finite sized scaling effects have been completely characterized analytically ([23]).

Numerical results are given extending this to the case where random Number Partition Problem instances are drawn from a Lévy-Stable distribution and show that this also obeys a phase transition with a slightly different form in its critical parameter. Heuristic arguments will then be given, one an analytic approximation to the phase transition order parameter and the other in the same spirit as Gent and Walsh's argument, both of which will motivate the choice of critical parameter.

The number partition problem represents an NP-Complete problem that is easy to state, has a large body of literature associated with it but still is lacking efficient algorithms to solve instances even in the so-called "easy" region, where the probability of a solution existing is almost surely 1.

Borgs, Chayes, Mertens and Nair's [21] [22] work on the local random energy model gives a good reason for the failure of a family of approximation algorithms. The random energy model hypothesis

states that the energy behaves randomly in the region in question and so any local progress is quickly destroyed when moving to neighboring configurations. Because of this, algorithms that use this type of heuristic as a judge of progress will, in general, fare not much better than random guessing.

The current state of the art for solving instances of the Number Partition Problem is the Complete Karmarkar Karp algorithm (CKK) ([70]). CKK employs a heuristic that works by removing the largest two integers in the list, re-inserting their difference then recurring on the smaller list. CKK is made into a complete search by recurring again after re-inserting the sum of the two largest integers removed should no solution be found after the initial recursion. The heuristic employed by CKK gives it an energy function that is the distance from optimal of a partial solution. Because of this, CKK behaves as a random energy model.

One might be tempted to conclude that the Number Partition Problem is intrinsically hard in all regions of the phase transition. For example, the Subset Sum Equality (or the "pigeonhole" version), where given n integers, a_k, such that $\sum_{k=0}^{n-1} a_k < 2^n - 1$, still appears to be difficult to discover the partition for even though one is guaranteed to exist. Papadimitriou gives the original formulation in [89] and as of this writing, it is still unknown whether Subset Sum Equality is polynomial time solvable, though the existence of such an algorithm would imply CoNP = NP ([10]).

Though persuasive, the local REM result only makes statements about a particular energy minimization and were a different heuristic to be used, one not based on a configuration's distance from optimal, perhaps this would fare better for finding solutions efficiently.

The author does not know of any algorithms that come close to finding solutions to instances of the Number Partition Problem, even well within the region where solutions are almost sure to exist. Lattice reduction techniques look promising as a potential candidate for finding solutions in the region where solutions are almost sure to exist. Lattice reduction techniques, specifically the Lenstra, Lenstra and Lovász (LLL) algorithm have enjoyed moderate success in finding solutions to a particular class of Subset Sum problems originally designed for use in cryptography. The author's attempts to use lattice reduction techniques to solve random instances of the Number Partition Problem have met with failure as problem sizes increase. These results are provided in the hopes that they might be helpful to someone trying to design an algorithm to work with the random Number Partition Problem in this region.

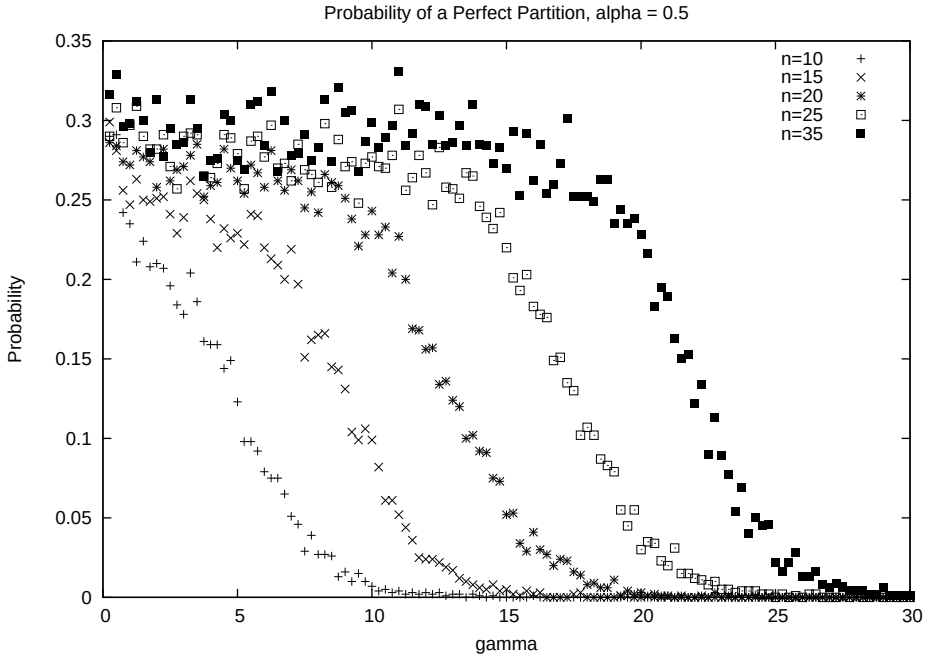


Figure 5.1: The probability of a perfect partition for $n = 10, 15, 20, 25, 30$ as a function of $m = \lg(\gamma)$, where γ is the scale parameter. Each point represents 1000 instances.

5.2 Truncated Lévy-Stable NPP Phase Transition

Instances for $n = \{10, 15, 20, 25, 30\}$ are created by choosing each number in the list from a truncated symmetric Lévy-Stable distribution. The following formula is used:

$$\Pr\{X_k = x\} \approx \lfloor | \frac{1}{2\pi} \int_{-\infty}^{\infty} exp(-itx - |\gamma t|^{\alpha}) dt | \rfloor$$

Where γ is the scale parameter, α is the critical exponent. For simplicity, integers are generated by truncating the absolute value of the sample from the family of continuous symmetric Lévy-Stable distributions. The GNU Scientific Library (GSL) was used for creating the instances.

Figures 5.1, 5.2 and 5.3 shows the probability of finding a solution for a given list length, n, as a function of the scale factor, $m = \lg(\gamma)$. Here $\lg(x) = \log_2(x)$. Each point represents 1000 simulations using the CKK algorithm to search for perfect partitions. A phase transition is observed, just as in the uniform case, for instances drawn from this family of truncated Lévy-Stable distributions.

Consider the following critical parameter, c:

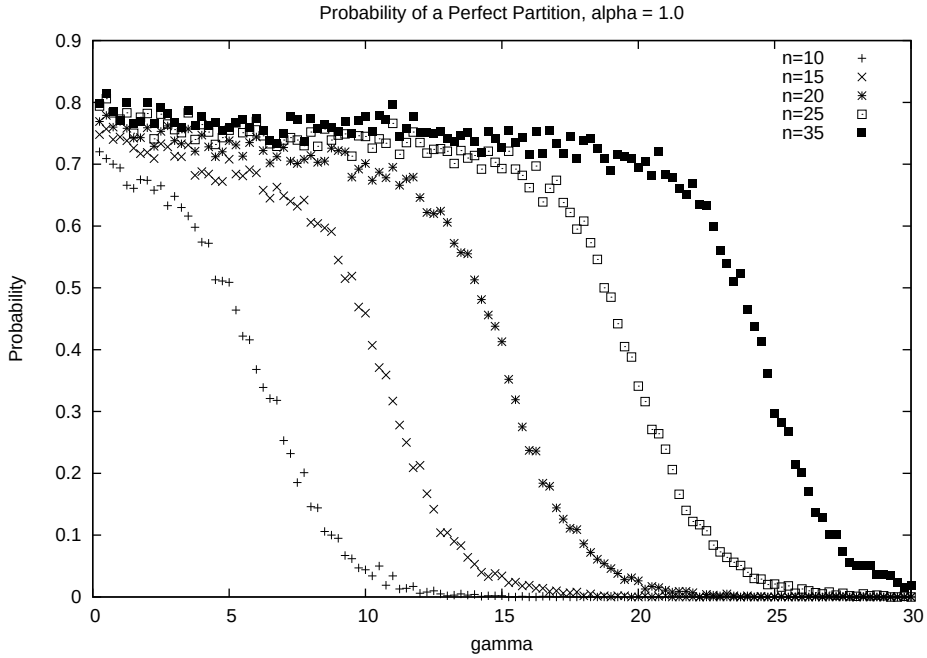


Figure 5.2: The probability of a perfect partition for $n = 10, 15, 20, 25, 30$ as a function of $m = \lg(\gamma)$, where γ is the scale parameter. Each point represents 1000 instances.

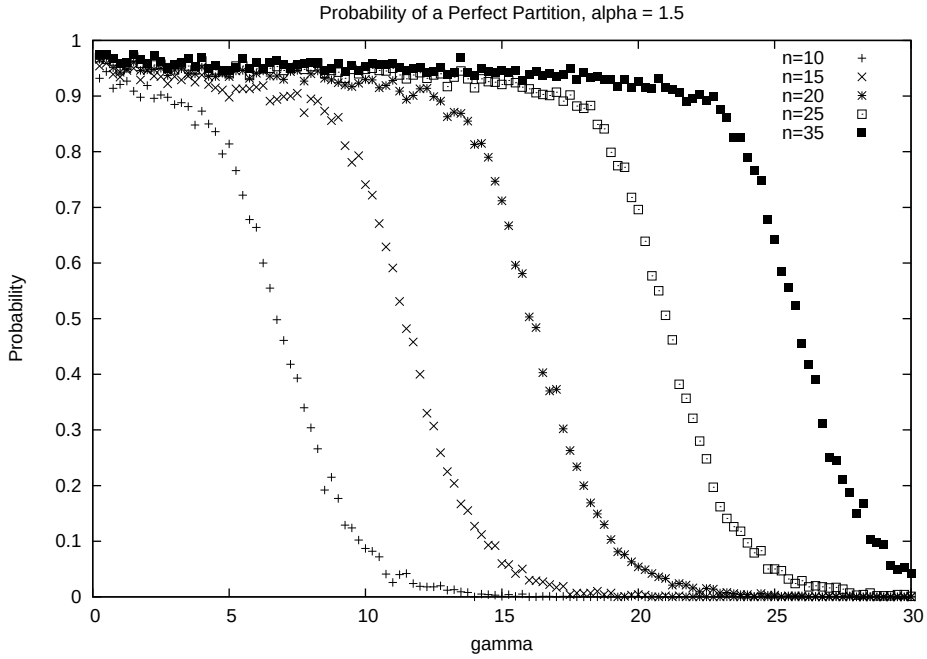


Figure 5.3: The probability of a perfect partition for $n = 10, 15, 20, 25, 30$ as a function of $m = lg(\gamma)$, where γ is the scale parameter. Each point represents 1000 instances.

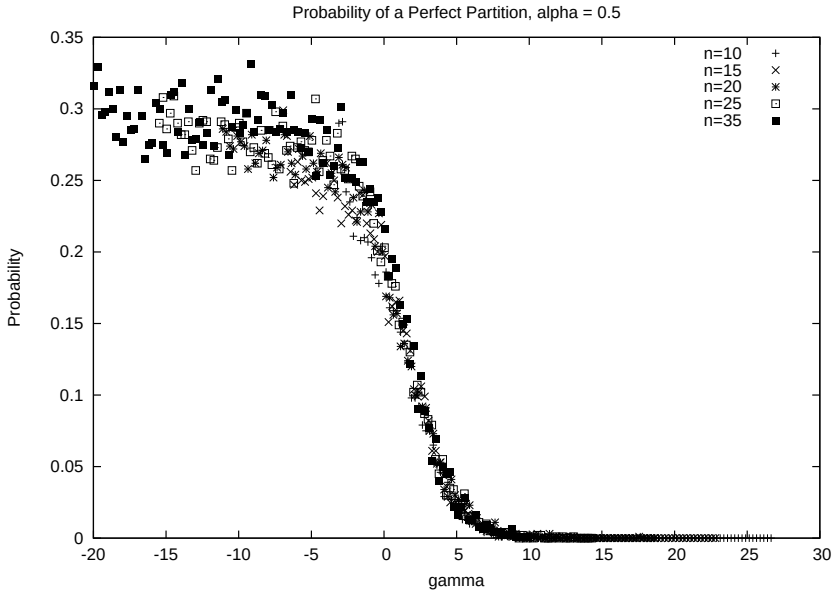


Figure 5.4: The probability of a perfect partition for $n = 10, 15, 20, 25, 30$ for $\alpha = 0.5$ as a function of c, where c is the rescaled and shifted parameter $c = m + \lg(n)/(2\alpha) - n$ for $m = lg(\gamma)$. Each point represents 1000 instances.

$$\gamma = 2^m$$

$$m = \kappa_n n$$

$$\kappa_n = 1 - \frac{lg(n)}{\alpha n} + \frac{c}{n}$$

Where c is the critical parameter. Notice that this parametrization is identical to the one given by Borgs, Chayes and Pittel [23] when $\alpha = 2$. The Gaussian distribution is the limiting distribution when the Lévy-Stable parameter $\alpha = 2$. This is most likely why the order parameters match in this case.

Figures 5.4, 5.5 and 5.6 shows the rescaled probability with the rescaled critical parameter.

The CKK algorithm was used in finding solutions. Figure 5.7, 5.8 and 5.9 show the probability of a perfect partition existing superimposed with the run time of the CKK algorithm as a function of nodes traversed for $\alpha \in \{0.5, 1.0, 1.5\}$.

Run times for the CKK algorithm have only been provided for completeness. Little weight should be given to the exponential increase in search cost exhibited by CKK as it almost surely gives no indication of the intrinsic difficulty of the Number Partition Problem instances generated. In

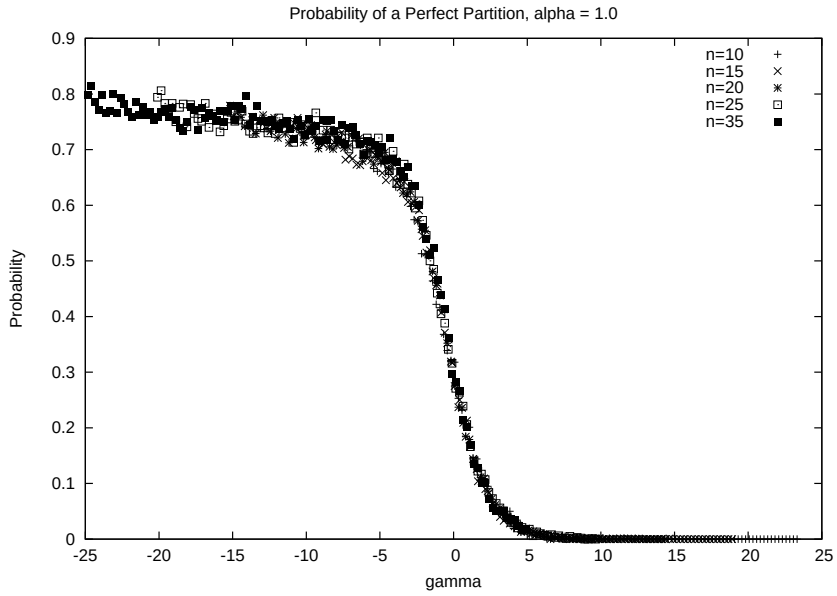


Figure 5.5: The probability of a perfect partition for $n = 10, 15, 20, 25, 30$ for $\alpha = 1.0$ as a function of c, where c is the rescaled and shifted parameter $c = m + \lg(n)/(2\alpha) - n$ for $m = lg(\gamma)$. Each point represents 1000 instances.

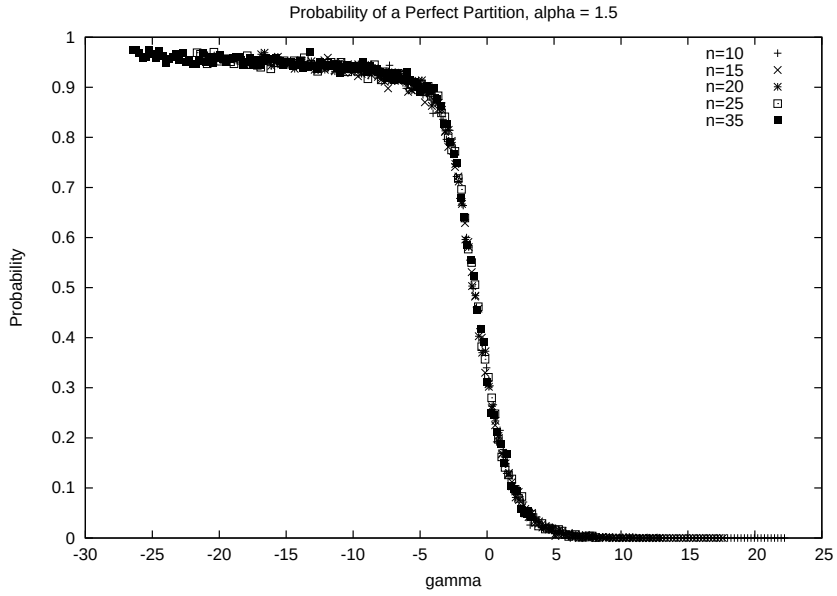


Figure 5.6: The probability of a perfect partition for $n = 10, 15, 20, 25, 30$ for $\alpha = 1.5$ as a function of c, where c is the rescaled and shifted parameter $c = m + \lg(n)/(2\alpha) - n$ for $m = lg(\gamma)$. Each point represents 1000 instances.

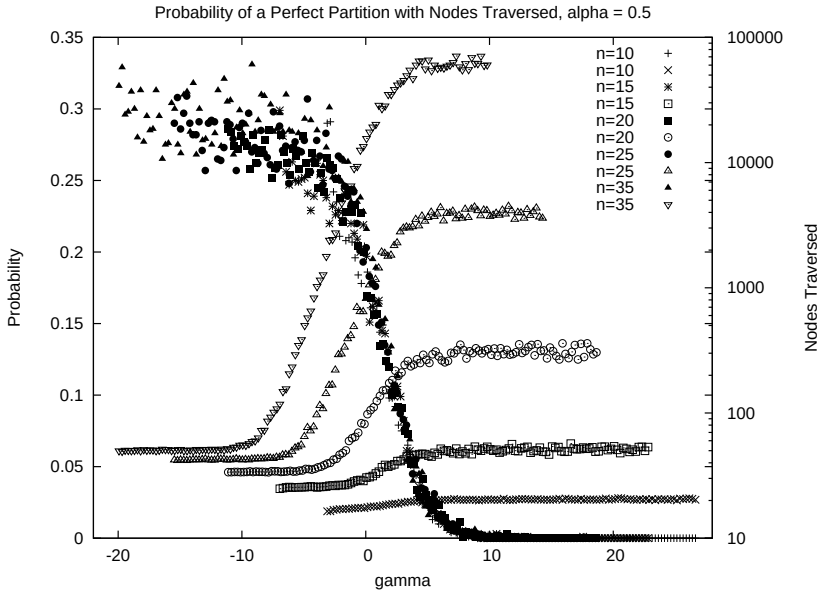


Figure 5.7: The run times for $\alpha = 0.5$ in terms of nodes traversed superimposed with the probability of a perfect partition existing. Experiments were done for list lengths $n = 10, 15, 20, 25, 30$ as a function of c, where c is the rescaled and shifted parameter $c = m + \lg(n)/(2\alpha) - n$ for $m = lg(\gamma)$. Each point represents 1000 instances.

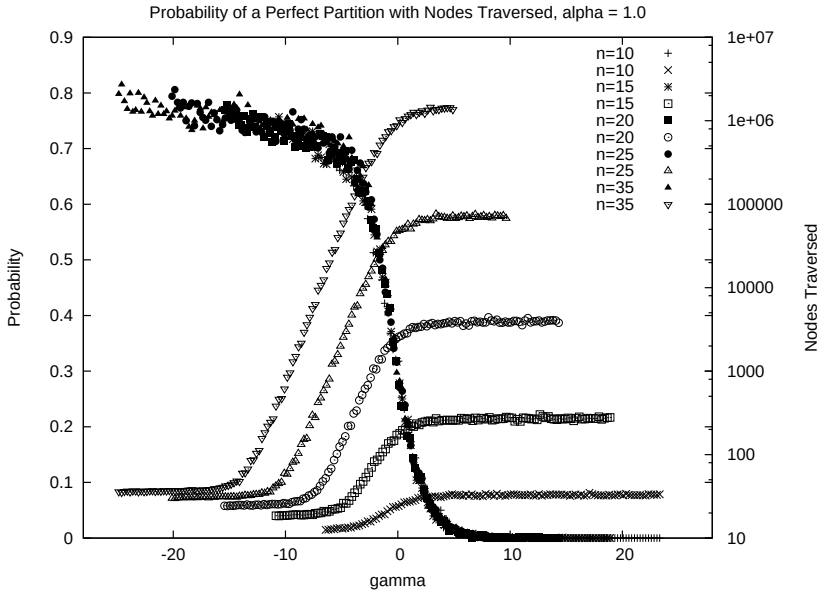


Figure 5.8: The run times for $\alpha = 1.0$ in terms of nodes traversed superimposed with the probability of a perfect partition existing. Experiments were done for list lengths $n = 10, 15, 20, 25, 30$ as a function of c, where c is the rescaled and shifted parameter $c = m + \lg(n)/(2\alpha) - n$ for $m = lg(\gamma)$. Each point represents 1000 instances.

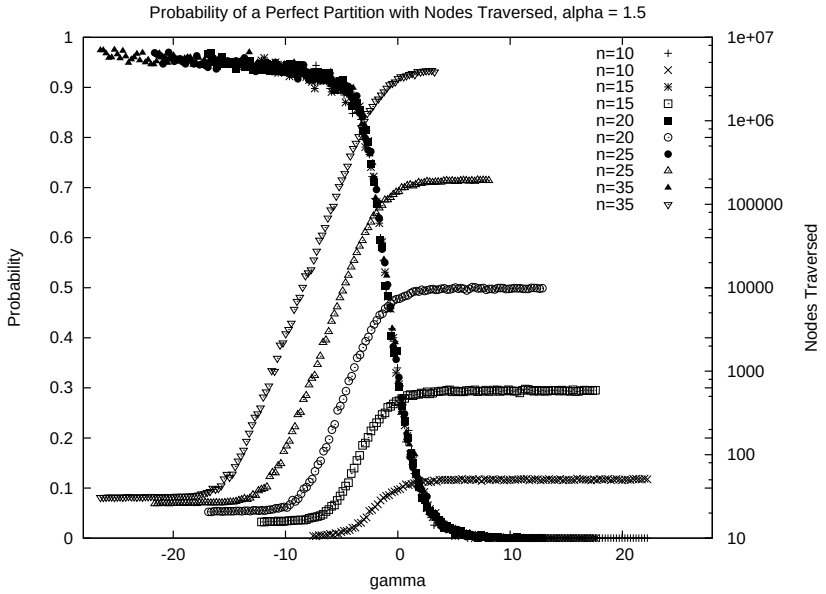


Figure 5.9: The run times for $\alpha = 1.5$ in terms of nodes traversed superimposed with the probability of a perfect partition existing. Experiments were done for list lengths $n = 10, 15, 20, 25, 30$ as a function of c, where c is the rescaled and shifted parameter $c = m + \lg(n)/(2\alpha) - n$ for $m = lg(\gamma)$. Each point represents 1000 instances.

the context of this thesis, the CKK algorithm is only used as a random search algorithm to find probabilities of perfect partitions of small instances. The exponential growth in run-time for the CKK algorithm is most likely because of its behavior as a random energy model at low temperatures and is thus subject to the results of the local-REM result proved by Borgs, Chayes, Mertens and Nair [21] [22].

It should be pointed out that the probability of finding a perfect partition for the instances generated is not unity for many of the plots. The reasons for this are non-trivial and to provide a complete reason for this is outside of the scope of this thesis. I will only provide a partial analysis with the understanding that a more complete answer would require further investigation.

A major contributing factor to this depression in probability in the easy region is the relative size of the maximum element generated exceeding the sum of the rest. If an NPP instance were generated such that the largest element exceeded the sum of the rest, this would preclude any perfect partition from occurring. For the probability of the maximum exceeding the sum of the remaining elements in the list, the following result is known:

$$M = \text{Max}(X_0, X_1, \ldots, X_{n-1})$$

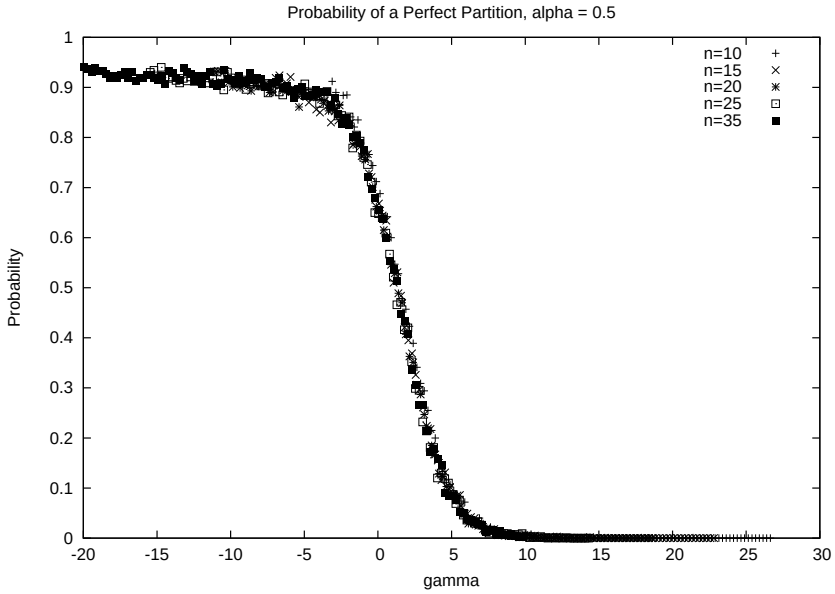


Figure 5.10: The probability of a perfect partition for instances whose maximum element does not exceed the sum of the rest. Plotted are values of $n = 10, 15, 20, 25, 30$ for $\alpha = 0.5$ as a function of c, where c is the rescaled and shifted parameter $c = m + \lg(n)/(2\alpha) - n$ for $m = lg(\gamma)$. Each point represents 1000 instances.

$$\Pr(M > \sum_{k=0}^{n-1} X_k - M) = \begin{cases} \sin(\pi\alpha/2)/(\pi\alpha), & 0 < \alpha < 1 \\ O(1/(\beta \lg(n))), & \alpha = 1 \\ O(1/n^{1-\alpha}), & 1 < \alpha < 2 \end{cases}$$

The reader is referred to Eliazar and Klafter [41], Yor and Pitman [92] and Perman [91] for details.

For $\alpha \geq 1$, this effect disappears for larger n and is only apparent in the the graphs generated because of the small list size. For $0 < \alpha < 1$, the maximum has a finite probability of being larger than the sum of the rest and so does not disappear as instance size increases.

Future work could take into account these large elements and reject the instance based on some simple heuristics, in this case making sure the largest element is smaller than the sum of the remaining elements. For comparison, Figures 5.10, 5.11 and 5.12 are provided showing the rescaled probability of a perfect partition for $n = 10, 15, 20, 25, 30$ for $\alpha \in \{0.5, 1.0, 1.5\}$ when only instances whose maximum element does not exceed the sum of the remaining elements in the list. For $\alpha = 0.5$, Figure 5.10 also displays a depression in probability, though not as severe as Figure 5.4. A significant portion of this effect is most likely due to the difference of the two maximum elements. Understanding which portion of the ensemble should be rejected as trivial Number Partition Problem instances requires

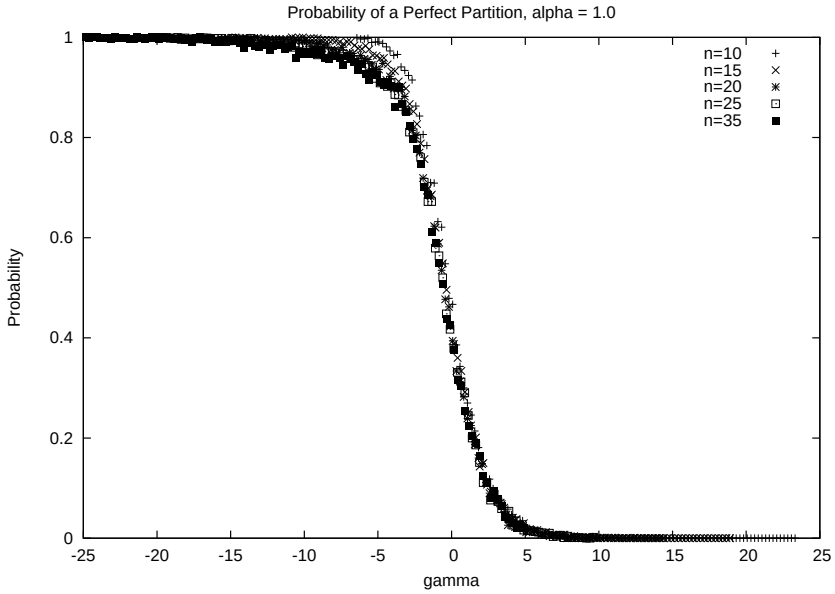


Figure 5.11: The probability of a perfect partition for instances whose maximum element does not exceed the sum of the rest. Plotted are values of $n = 10, 15, 20, 25, 30$ for $\alpha = 1.0$ as a function of c, where c is the rescaled and shifted parameter $c = m + \lg(n)/(2\alpha) - n$ for $m = lg(\gamma)$. Each point represents 1000 instances.

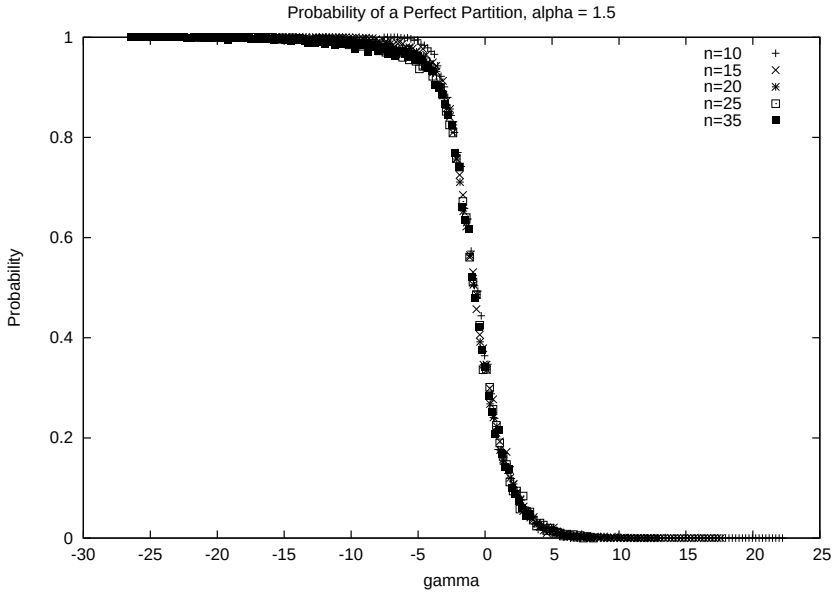


Figure 5.12: The probability of a perfect partition for instances whose maximum element does not exceed the sum of the rest. Plotted are values of $n = 10, 15, 20, 25, 30$ for $\alpha = 1.5$ as a function of c, where c is the rescaled and shifted parameter $c = m + \lg(n)/(2\alpha) - n$ for $m = lg(\gamma)$. Each point represents 1000 instances.

further investigation and is left as a potential avenue of exploration for future work.

5.3 Heuristic Derivation of Critical Parameter

In this section two heuristic derivations are provided for the critical order parameter of the Number Partition Problem phase transition. These are meant to be non-rigorous arguments that give motivation for the choice of $\kappa_n = 1 - lg(n)/(\alpha n) - c/n$ in the previous section. While the derivations that follow are non-rigorous and elementary, to the author's knowledge, the order parameter has not been discussed for the case when NPP instances are chosen from the family of Lévy-Stable distributions. Both of the following methods are heuristic derivations of the order parameter but agree well with the small numerical simulations provided in the previous section. When $\alpha = 2$, the derivations that follow also agree well with the calculated order parameter predicted by Borgs, Chayes and Pittel [23].

5.3.1 Analytic Evidence of the Number Partition Problem Order Parameter

In this section an analytic heuristic derivation of the critical parameter for the Number Partition Problem (NPP) phase transition is presented. Following Mertens [79], and Borgs, Chayes and Pittel [23], one can write an integral representation of the number of solutions for the Number Partition Problem. Consider the indicator function for an instance of a Number Partition Problem. Below, $\sigma_k \in \{-1, 1\}$, $k \in \mathbb{Z}_{\geq 0}$ and X_k are independent and identically distributed (i.i.d.) random variables (r.v.'s) from which we generate an NPP instance:

$$\mathbb{I}(\sum_{k=0}^{n-1} \sigma_k X_k = 0) = \frac{1}{2\pi} \int_{-\pi}^{\pi} \exp(i(\sum_{k=0}^{n-1} \sigma_k X_k)\theta) d\theta$$

By noticing that the exponent in the exp term only takes on integral values and the integral, $\mathbb{I}(\cdot)$, will only be non-zero in the case of a zero exponent, gives us an indicator function as desired.

Define Z_n as follows:

$$Z_n = \sum_{\sigma} \mathbb{I}(\sum_{k=0}^{n-1} \sigma_k X_k = 0)$$

Where the outer sum is over all different configurations of $\sigma_k \in \{-1, 1\}$. This gives us a random integral representation for the number of solutions:

$$Z_n = \frac{1}{2\pi} \int_{-\pi}^{\pi} \prod_{k=0}^{n-1} (e^{iX_k\theta} + e^{-iX_k\theta}) d\theta$$

If we then take the expectation, this yields:

$$\begin{aligned} \mathbb{E}[Z_n] &= \frac{1}{2\pi} \int_{-\pi}^{\pi} \prod_{k=0}^{n-1} (\mathbb{E}[e^{iX_k\theta} + e^{-iX_k\theta}]) d\theta \\ &= \frac{1}{2\pi} \int_{-\pi}^{\pi} \mathbb{E}[e^{iX'\theta} + e^{-iX'\theta}]^n d\theta \end{aligned}$$

Where we can replace the product of i.i.d. r.v.'s by a single function in a new r.v., X', drawn from the same distribution. See Borgs, Chayes and Pittel [23] for how they use this representation to find the order parameter and other results for the phase transition of the Number Partition Problem when instance elements are drawn from a uniform distribution.

Instead of choosing X' from a uniform distribution, X' is chosen to be a Lévy-Stable distribution. This is only approximate, as the choice of instance generation in the previous section was not drawn from Lévy-Stable distribution, but a truncated, absolute valued Lévy-Stable distribution. This gives an upper bound for $\mathbb{E}[Z_n]$ and is provided only to motivate the choice of order parameter in the previous section.

Since the expectation of the sum is the sum of the expectations, we can continue on:

$$\mathbb{E}[Z_n] = \frac{1}{2\pi} \int_{-\pi}^{\pi} (\mathbb{E}[e^{iX'\theta}] + \mathbb{E}[e^{-iX'\theta}])^n d\theta \tag{5.1}$$

If we take X' to be a continuous symmetric Lévy-Stable r.v. with critical exponent α and scale parameter γ we can reduce further. First, since X' is symmetric, we have $\mathbb{E}[e^{iX'\theta}] = \mathbb{E}[e^{-iX'\theta}]$. Second, $\mathbb{E}[e^{iX'\theta}]$ is the characteristic function of the random variable X', which is known to be $\phi(t) = \exp(-\gamma^\alpha |t|^\alpha)$ for a symmetric Lévy-Stable random variable. This produces:

$$< \frac{1}{2\pi} \int_{-\pi}^{\pi} [2e^{-\gamma^\alpha |\theta|^\alpha}]^n d\theta = \frac{2^n}{2\pi} \int_{-\pi}^{\pi} e^{-n\gamma^\alpha |\theta|^\alpha} d\theta$$

Substituting $t = n^{1/\alpha}\gamma\theta$ gives:

$$= \frac{2^n}{2\pi n^{1/\alpha}\gamma} \int_{-n^{1/\alpha}\gamma\pi}^{n^{1/\alpha}\gamma\pi} e^{-|t|^\alpha} dt < \frac{2^n}{2\pi n^{1/\alpha}\gamma} \int_{-\infty}^{\infty} e^{-|t|^\alpha} dt$$

$$= \frac{2^n}{\pi n^{1/\alpha}\gamma} \int_0^{\infty} e^{-t^\alpha} dt = \frac{2^n}{\pi n^{1/\alpha}\gamma} \Gamma(\frac{1}{\alpha} + 1)$$

To finally give:

$$\mathbb{E}[Z_n] < \exp[\ln(2)(n - \lg(\pi) - \frac{\lg(n)}{\alpha} - \kappa_n n + \lg\Gamma(\frac{1}{\alpha} + 1))]$$

Which implies a parametrization for κ_n:

$$\kappa_n = 1 - \frac{\lg(n)}{\alpha n} - \frac{c}{n} - (\frac{\lg(\pi) + \lg\Gamma(\frac{1}{\alpha} + 1)}{n})$$

The $(\lg(\pi) + \lg\Gamma(\frac{1}{\alpha} + 1))$ term is only dependent on α and effectively provides a constant shift to the order parameter, c. If the $(\lg(\pi) + \lg\Gamma(\frac{1}{\alpha} + 1))/n$ term is ignored, or is allowed to be subsumed into the c term, this in agreement with Borgs, Chayes and Pittel's [23] result when $\alpha = 2$.

5.3.2 Numerical Evidence of the Number Partition Problem Order Parameter

This section provides an alternate derivation of the order parameter based more on numerical evidence than analytic heuristics. Consider, again, the scale parameter $\gamma = 2^m$ and critical exponent α. We wish to find the log of the scale parameter, $\lg(\gamma) = m$ in terms of the critical parameter, κ_n and the length of the list n: $m = \kappa_n n$. What follows is a heuristic that derives this parameter.

In a very broad sense, the scale parameter, γ, sets the region in which the distribution behaves uniformly and the critical exponent, α, tells us how quickly the tails drop off after this region. If we were to now restrict our attention to a truncated absolute valued version of this random variable and ask what the probability of a bit being set is, the lower order bits would behave, for the most part, as if they were drawn at random with probability 1/2 whereas the higher order bits would be set with an exponentially decreasing probability. Figure 5.13, 5.14 and 5.15 for $\alpha \in \{0.5, 1.0, 1.5\}$,

$\gamma \in \{1, 2, 4, \ldots, 16384\}$ shows the result of a Monte-Carlo simulation to estimate the probability of a bit being set. Each data point in Figure 5.13, 5.14 and 5.15 represents the average of 1,000,000 iterations.

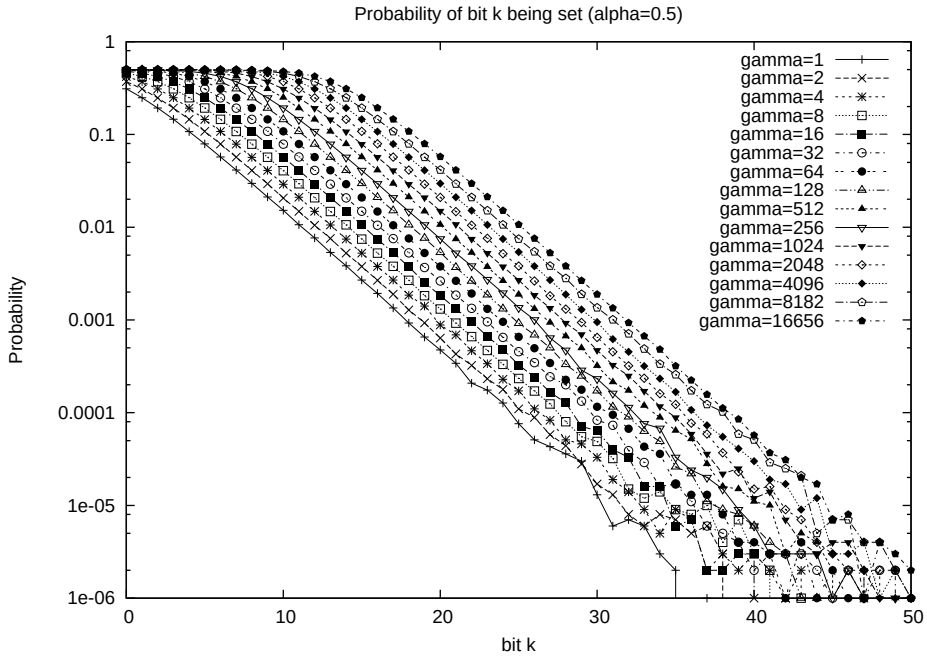


Figure 5.13: Probability of a bit being set for a random draw of a Lévy-Stable distributed random variable with $\alpha = 0.5$, $\gamma \in \{1, 2, 4, 8, \ldots, 16384\}$. Each point represents the average of 1,000,000 iterations.

Call s the point at which the probability of a bit being set transitions from 1/2 to something lower. From Figure 5.13, 5.14 and 5.15 one observes that s is related to the scale parameter by $s \approx \lg(\gamma)$. Call ρ the exponential parameter where the probability of a bit being set drops off i.e. $\Pr\{\text{bit k set}, k > s\} \propto e^{-\rho k}$. From observation, the exponential parameter, ρ, only depends on the critical exponent of the Lévy-Stable distribution, α, and not on the scale parameter, γ. Figure 5.16 is provided showing ρ as a function of α.

The sum of n independent, identically distributed (i.i.d.) Lévy-Stable random variables (r.v.) is again a Lévy-Stable r.v. with a new scale parameter of $n^{1/\alpha}\gamma$. We can now ask what the probability of all bits being 0 is for a random configuration, or sum, of the n original absolute valued truncated Lévy-Stable random variables.

$$\Pr\{\exists \text{perfect partition}\} = 1 - \Pr\{\text{no partition exists}\}$$

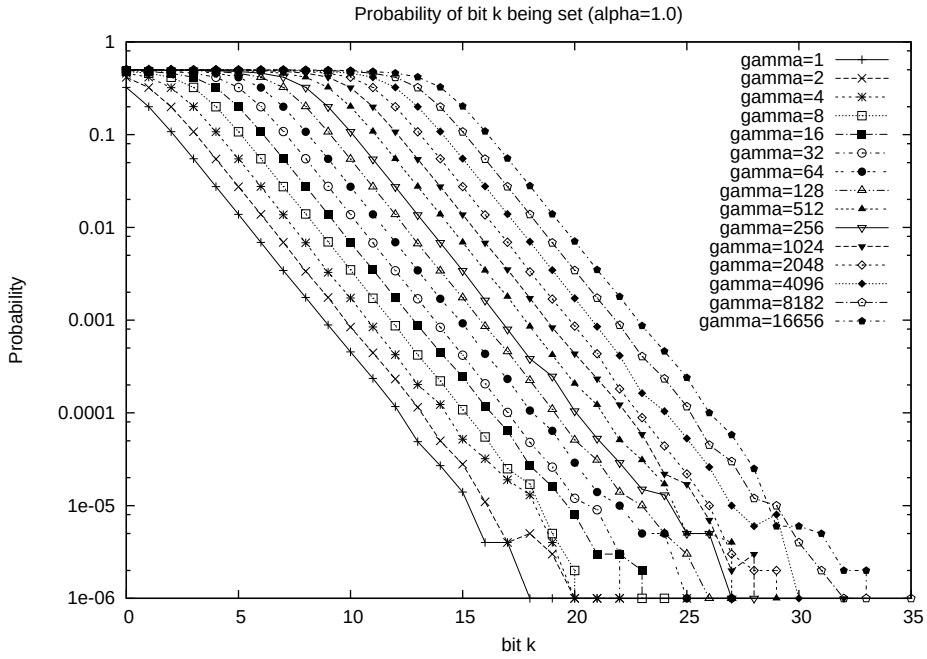


Figure 5.14: Probability of a bit being set for a random draw of a Lévy-Stable distributed random variable with $\alpha = 1.0$, $\gamma \in \{1, 2, 4, 8, \ldots, 16384\}$. Each point represents the average of 1,000,000 iterations.

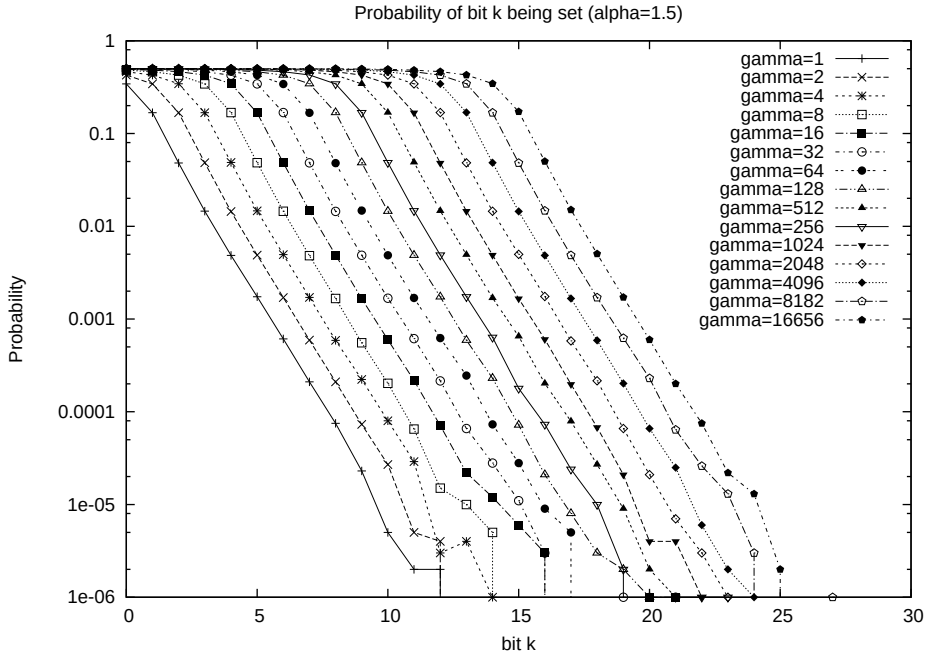


Figure 5.15: Probability of a bit being set for a random draw of a Lévy-Stable distributed random variable with $\alpha = 1.5$, $\gamma \in \{1, 2, 4, 8, \ldots, 16384\}$. Each point represents the average of 1,000,000 iterations.

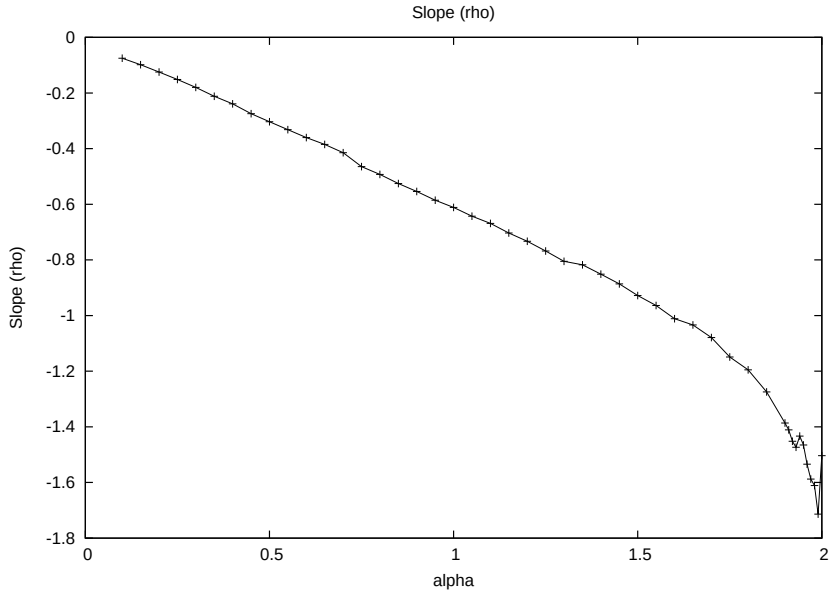


Figure 5.16: The exponent parameter of the region where the probability of finding a bit set goes from constant 1/2 to an exponentially decreasing function of the bit position. Each data point represents the average of 1,000,000 iterations.

If we make the approximation that the probability that each partition is independent from all the others, we can further reduce:

$$\approx 1 - [1 - 2^{-s} \prod_{k=0}^{\infty} e^{-2^{-(\rho k+1)}}]^{2^n} = 1 - [1 - 2^{-s+\lg(2)(1-2^{-\rho})^{-1}/2}]^{2^n}$$

Using $(1-x) \leq e^{-x}$ for $0 \leq x \leq 1$ we find:

$$\leq 1 - \exp(-2^{-s+n+\lg(2)(1-2^{-\rho})^{-1}})$$

Call $\gamma' = n^{1/\alpha}\gamma$ and we know that $s \approx \lg(\gamma') = \lg(n)/\alpha + \lg(\gamma)$, we have:

$$\approx 1 - \exp(-2^{-(\lg(n)/\alpha+m)+n+\lg(2)(1-2^{-\rho})^{-1}})$$

$$= 1 - \exp(-2^{-\kappa_n n-\lg(n)/\alpha+n+\lg(2)(1-2^{-\rho})^{-1}})$$

Besides the term involving ρ, if we set $\alpha = 2$, this is the form that the critical parameter κ_n takes when choosing instances from a uniform distribution. In general:

$$\kappa_n = 1 - \lg(n)/(\alpha n) + c/n + \lg(2)(1 - 2^{-\rho})^{-1}/n$$

For any particular α, ρ can be ignored when considering the critical parameter as it only contributes a constant shift to c. Ignoring the right most term gives us:

$$\kappa_n = 1 - \lg(n)/(\alpha n) + c/n$$

As desired.

5.4 Lattice Reduction Techniques

In this section, some numerical work in using lattice reduction techniques to solve the Random Number Partition Problem where instances are drawn from a uniform distribution is briefly discussed. The LLL algorithm is chosen for its simplicity and accessibility in the literature. The reader is referred to [74], [106], [24] and chapter 3 for details of the LLL algorithm. Chapter 3 also discusses the different encodings of the Number Partition Problems and related Subset Sum problems as questions about finding short vectors on lattices.

Three models were used in the analysis. The first model follows Lagarias and Odlyzko ([73]), by converting a Number Partition Problem instance into a Subset Sum instance. This is done by using an $(n+1)$ x $(n+2)$ basis as follows:

$$s = (1/2)\sum_{k=0}^{n-1} a_k$$

$$B = \begin{pmatrix} 1 & 0 & \dots & 0 & Ca_0 \\ 0 & 1 & \dots & 0 & Ca_1 \\ \vdots & \vdots & \ddots & & \vdots \\ 0 & 0 & \dots & 0 & Ca_{n-1} \\ 0 & 0 & \dots & 1 & -Cs \end{pmatrix}$$

The second is a model proposed by Coster, Joux, Lamacchia, Odlyzko, Schnorr and Stern ([37]) constructs the $(n+1)$ x $(n+2)$ basis as follows:

$$B = \begin{pmatrix} 2 & 0 & \dots & 0 & 0 & Ca_0 \\ 0 & 2 & \dots & 0 & 0 & Ca_1 \\ \vdots & \vdots & \ddots & & \vdots & \vdots \\ 0 & 0 & \dots & 2 & 0 & Ca_{n-1} \\ -1 & -1 & \dots & -1 & 1 & -Cs \end{pmatrix}$$

The third model used is one also proposed by Coster, Joux, Lamacchia, Odlyzko, Schnorr and Stern ([37]) and constructs the $(n+1)$ x $(n+2)$ basis as follows:

$$B = \begin{pmatrix} n+1 & -1 & \dots & -1 & -1 & Ca_0 \\ -1 & n+1 & \dots & -1 & -1 & Ca_1 \\ \vdots & \vdots & \ddots & & \vdots & \vdots \\ -1 & -1 & \dots & n+1 & -1 & Ca_{n-1} \\ -1 & -1 & \dots & -1 & n+1 & -Cs \end{pmatrix}$$

The range is chosen such that $m << n$ so that problem instances should fall well within the region where solutions are almost sure to exist. Partial solutions are ones such that $k \leq n, i_0 < i_1 < \cdots < i_{k-1}$ such that $\sum_{j=0}^{k-1} \sigma_j a_{i_j} = 0, \sigma_j \in \{-1, 1\}$. Figures 5.17, 5.18 and 5.19 shows the average length of the partial solution found for Number Partition Problem with $n = \{20, 30, 40, 50, 60, 70, 80, 90, 100\}$ as a function of the bit size of the range. Each point represents the average of 100 random instances created.

Figure 5.20, 5.21 and 5.22 shows average length of basis vectors where an integer not necessarily from the restricted set are found. From figures 5.20, 5.21 and 5.22 one can see that the length of the short vectors found are extremely small but that finding vector with a relation that is from the restricted set becomes much more improbable. This is most likely due to the fact that there is an abundance of short vectors in the neighborhood of the desired short vector. Looking at the set of figures 5.17, 5.18 and 5.19 of the average length of the partial solution compared with the set of figures 5.20, 5.21 and 5.22 of the average length of the short vector found, one can make a hypothesis that there is an abundance of short vectors that have small but invalid NPP/Subset Sum solutions in the neighborhood of the desired relations.

Each of these methods finds partial solutions for a limited range of bit size. LLL is good at finding small integer relations, often much smaller than the theoretical exponential limit. For instances

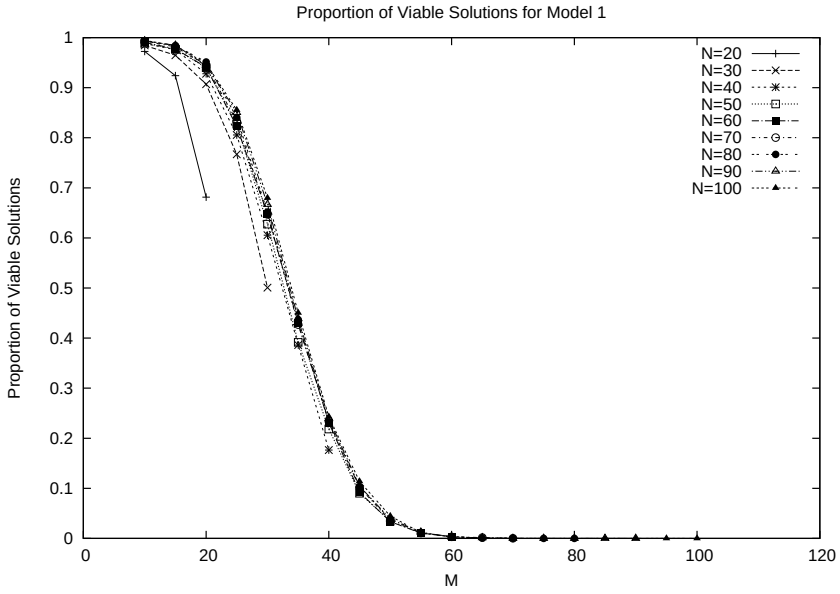


Figure 5.17: The proportion of lattice vectors that correspond to an integer relation that are viable in the sense of being partial solutions to the Number Partition Problem (coefficients that contain only -1, 0 or +1) for the first model. Each point represents 100 instances.

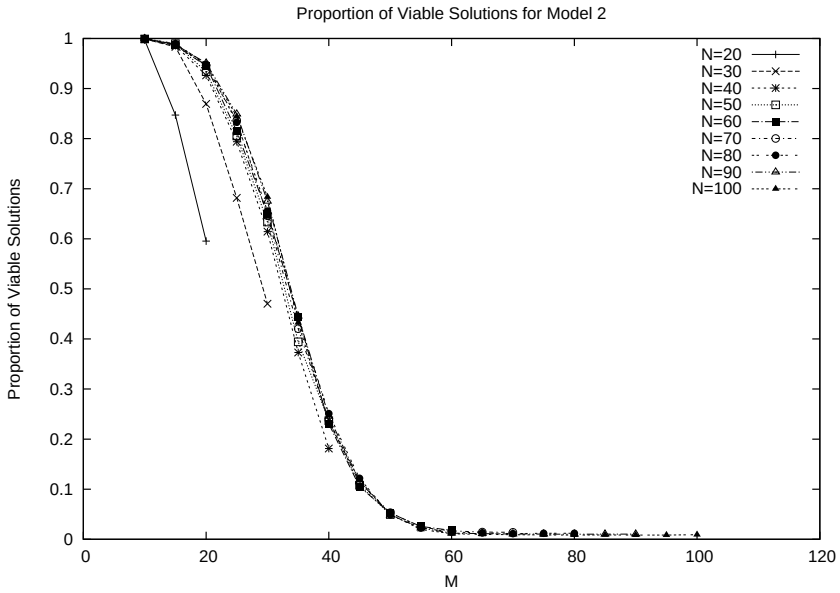


Figure 5.18: The proportion of lattice vectors that correspond to an integer relation that are viable in the sense of being partial solutions to the Number Partition Problem (coefficients that contain only -1, 0 or +1) for the second model. Each point represents 100 instances.

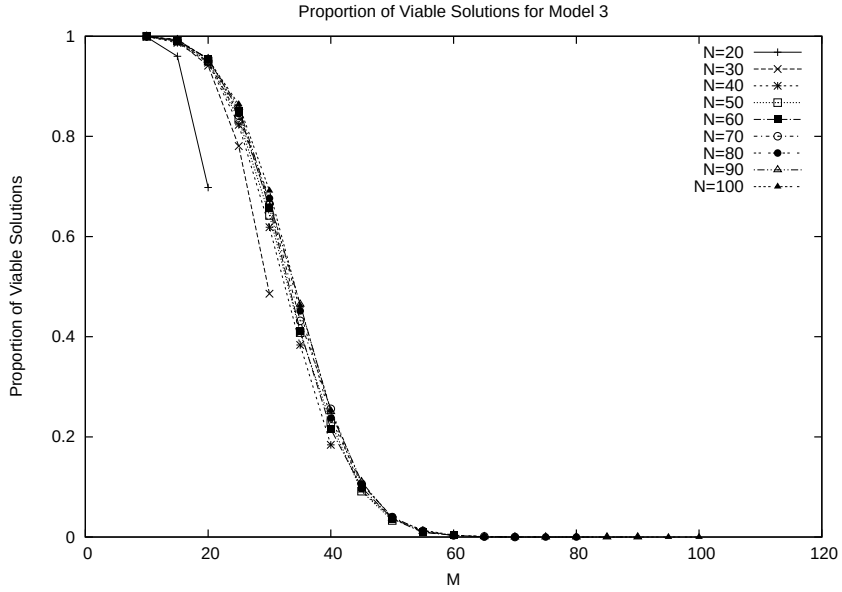


Figure 5.19: The proportion of lattice vectors that correspond to an integer relation that are viable in the sense of being partial solutions to the Number Partition Problem (coefficients that contain only -1, 0 or +1) for the third model. Each point represents 100 instances.

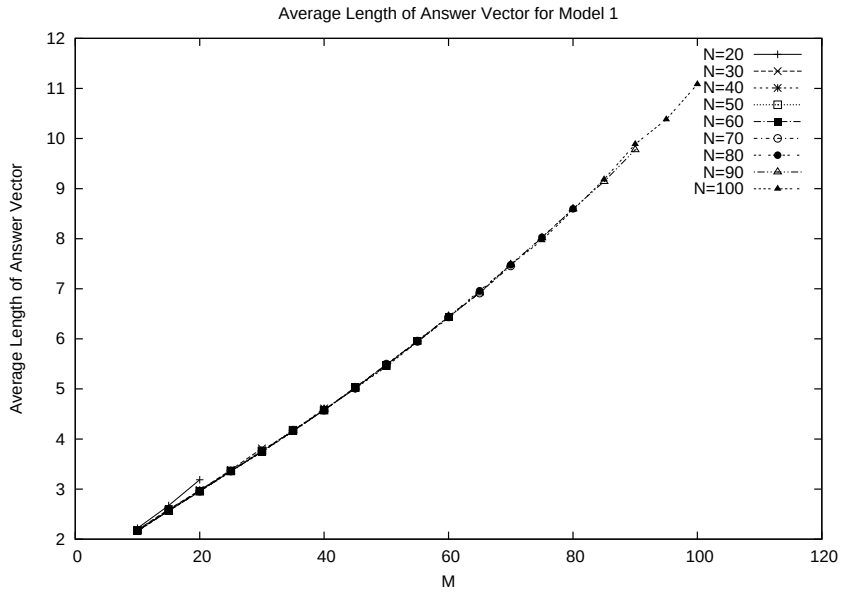


Figure 5.20: The average length of lattice vectors found by LLL that correspond to an integer relation for the first model.

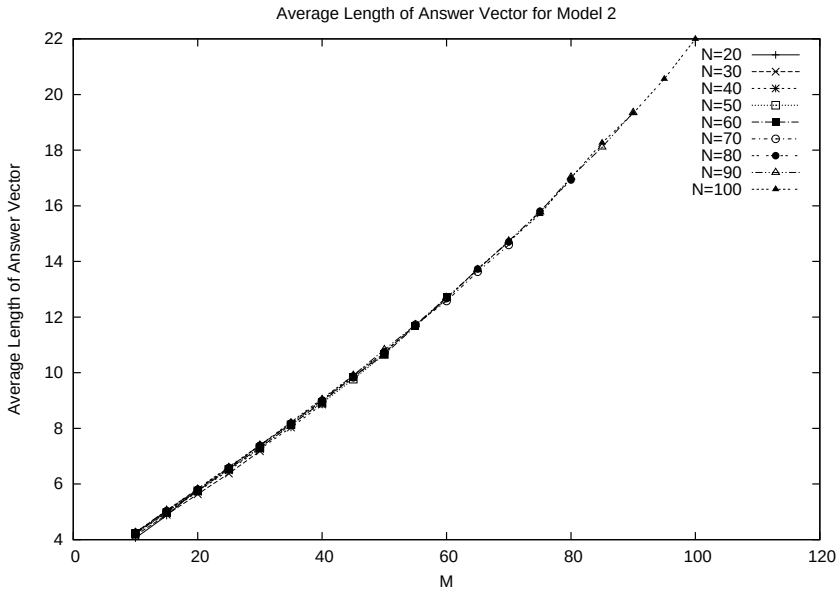


Figure 5.21: The average length of lattice vectors found by LLL that correspond to an integer relation for the second model.

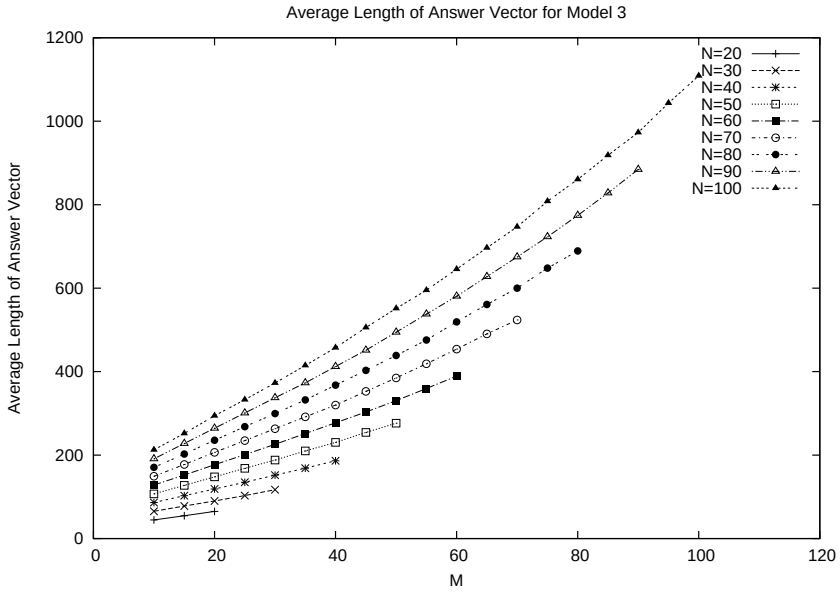


Figure 5.22: The average length of lattice vectors found by LLL that correspond to an integer relation for the third model.

generated in this way, the coefficients of the solutions found are often smaller than $\lg(n)$ and are thus much smaller than the exponential limit would otherwise suggest. Unfortunately, encoding NPP or Subset Sum instances in this way does not preclude other coefficients than the ones desired and even a small integer relation must be discarded when the coefficients do not fall within the restricted set of $\{-1, 1\}$ or $\{0, 1\}$.

In some sense the basis vectors created are giving 'hints' to the LLL algorithm in order to make it more probable that it find short vectors with the coefficients from the desired restricted set. As system size increases, the length of the shortest vector increases in the 2 norm and this allows for other short vectors that have coefficients that are outside of the restricted set regardless of the basis used. Perhaps there is another basis that can be used in order to make coefficients from the restricted set more probable, but at the time of this writing the author knows of no such encoding to make this possible.

It should be noted that algorithms searching for small integer relations using distance from 0 of an attempted solution as a search heuristic suffer from the same failure that CKK does. These algorithms have wild oscillations in their energy function that destroy any local progress that is made. LLL is a candidate for an algorithm that might not display random energies near optimal solutions. Anecdotal evidence is present with Lagarias and Odlyzko [73] and Schnorr and Euchner [95] who use have used LLL to solve low density ($m << n$) subset sum instances. From figures 5.17 through 5.22, it can be seen that integer relations are found with coefficient sizes extremely small but, since they do not fit the criteria from the restricted set of $\{-1, 1\}$ or $\{0, 1\}$, must be rejected as NPP or Subset Sum solutions.

It is the author's opinion that the technique employed by lattice reduction algorithms for making local progress on an appropriately defined energy landscape make them a good candidate for investigation into algorithms attempting to approximate the NPP or Subset Sum. Perhaps a better encoding or a different lattice reduction method, such as PSLQ, could be used to make viable solutions more probable. The reader is referred to Ferguson, Bailey and Arno [47] for details on the PSLQ algorithm. Another attractive feature of the PSLQ algorithm is in finding a lower bound on any integer relation that can occur and perhaps this could be exploited to give an indication of whether a given instance has a perfect partition or not.

5.5 Conclusion

Numerical evidence has been provided showing that a phase transition occurs even for Number Partition Problems whose list entries are drawn from a truncated Lévy-Stable distribution. Because of a lack of approximation algorithms, it is very difficult to gauge how intrinsically hard these problems actually are. The state of the art algorithms, such as the Complete Karmarkar Karp (CKK), have exponential run-times for all instances save for those whose entries are chosen for ranges that are exponentially smaller than the list length. There is a large region in between where the CKK algorithm becomes exponential in its run-time to where the critical threshold is hit where the solution of a random NPP instance almost surely exists. It is difficult for the author to believe that this almost sure probability 1 region has intrinsically hard NPP instances in it, though it is not outside the realm of possibility as the Subset Sum Equality (or the "pigeonhole" version) still appears to be difficult.

Because of the abundance of solutions and the close success of lattice reduction techniques, it is the author's opinion that instances to the Number Partition Problem have almost sure polynomial algorithms in the probability 1 region. It is the author's suspicion that Number Partition Problem instances drawn from a truncated Lévy-Stable distribution near the transition will be much more difficult than the uniform random Number Partition Problem.

Lattice reduction techniques show promise as a technique that can be used for better approximation algorithms. The failure of LLL in finding solutions to even moderately sized instances was presented in the hopes that the same dead end will not be traversed by others and that it may be expanded upon by someone with more insight than the author possesses into the problem.

Regardless of the failure of lattice reduction techniques to solve instances of the Number Partition Problem in the probability 1 region, the order parameter was numerically verified for instances whose elements were drawn from a modified Lévy-Stable distribution. Hopefully this will provide an initial stepping stone for further research into the NPP phase transition and provide insight that can be used in the future to aid in the development of approximation algorithms.

Chapter 6

Conclusion and Discussion

An important tool in the analysis of algorithms for NP-Complete problems is to consider random ensembles of instances drawn from NP-Complete classes. Understanding where hard instances are located in the ensemble gives insight into when general algorithms should fail. Conversely, locating where easy instances are in the ensemble could point us to new algorithms designed to exploit the structure of the problem that make it easy.

6.1 Contributions

Numerical evidence has been presented of an ensemble that could result in intrinsically hard graphs when attempting to determine Hamiltonicity. To the author's knowledge, no general random graph ensemble has been presented as a candidate for generating graphs that are intrinsically hard.

Care needs to be taken when making claims on intrinsic complexity for instances generated. The class of graphs has been presented only as a candidate for hard instance generation for determining Hamiltonicity. There could be algorithms specifically suited to take advantage of the power law degree structure of the graphs generated that overcome the pitfalls of the algorithms used in this thesis's analysis.

Analysis of the Number Partition Problem has been given showing that there is also a phase transition when instances are drawn from a truncated Lévy-Stable distribution. The failure of algorithms has been discussed and lattice reduction techniques have been proposed as a potentially fruitful class of algorithms to try and address these failures. The lattice reduction techniques have not met

with success. These failures were presented anyway in the hopes of providing greater insight into the nature of the failure and give more information should someone consider lattice reduction techniques to solve easy instances of the NPP in the future.

6.2 Discussion

6.2.1 Hamiltonian Cycle

The Hamiltonian Cycle Problem has been well studied both numerically and analytically. This makes it an ideal candidate as a representative in the analysis of random NP-Complete problems. The hope is that understanding where the phase transition is for the Hamiltonian Cycle problem and where the associated intrinsically hard graphs are provides a good starting point in understanding where hard instances lie for other NP-Complete problems.

It needs to be stressed that the analysis of where hard graph instances are depends critically on the algorithms used. The class of random graphs presented could be easily solvable with a better algorithm that exploits knowledge of the degree distribution. While Erdös-Rényi random graphs are easy, this does not imply that graphs generated with diverging second moment, either by the method presented or some other, are difficult. The class of random graphs presented is only a potential candidate for generating intrinsically hard instances of graphs for Hamiltonicity determination.

It should also be pointed out that the degree distribution of a graph is essentially an arbitrary characterization that need not give any indication of the intrinsic difficulty of finding a Hamiltonian Cycle. For example, via a simple reduction, one can reduce an arbitrary graph, power law degree distributed or otherwise, to a 3-regular graph by an appropriate replacement of vertices with 3-regular 'widgets', preserving Hamiltonicity. This would destroy the degree distribution while still retaining all of the potential difficulty in discovering a Hamiltonian Cycle. The question then arises of what is a better characterization of graphs that give rise to difficult instances of Hamiltonian Cycles. One candidate is characterizing graphs via their spectra, either by the eigenvalues of the adjacency matrix or one of its siblings (the Laplacian, etc). Unlike the Erdös-Rényi graphs, the eigenvalues of power law degree distributed graphs do not obey the Wigner Semi-Circular law ([45], [83]). Perhaps random graphs, even ones that have been constructed to have a k-regular degree sequence, but "secretly" have a power law degree graph embedded in them, would be better characterized by their spectra rather than their degree sequence. Understanding which graph invariant correctly parametrizes

the space of random graphs and their associated difficulty in finding Hamiltonian Cycles would be helpful. The degree distribution is only presented as a first attempt in this characterization.

6.2.2 Number Partition Problem

The Number Partition Problem has been well studied analytically but still proves intractable for numerical analysis for any moderate size, even well within the region of the phase transition where a solution almost surely exists. The failure of a large class of algorithms is well understood when considering the Random Energy Model (REM) hypothesis but does not exclude every algorithm ([21], [22]). Algorithms that use a better cost metric, ones that do not suffer from the conditions that would make them behave as random energy models, could be better at finding solutions, should they exist.

The lattice reduction techniques potentially do not suffer from the conditions that would make it behave as a random energy model. LLL had enjoyed moderate success solving 'low-density' Subset Sum problems ([37], [73], [95]) but does not extend easily into Number Partition or Subset Sum Problems in the 'high-density' region, where solutions are almost sure. Perhaps with a better encoding, a different reduction technique or a different algorithm that still preserves the essence of what makes the lattice reduction work well, would fare better.

Until we have better approximation algorithms to analyze the region close to the phase transition, or until we have a better theory that explains where difficult instances reside, one cannot make any clear statement as to the intrinsic difficulty of instances generated. As time has progressed, many random NP-Complete problems thought to once be difficult, even near the transition point, have turned out to be easy. The Hamiltonian Cycle for Erdös-Rényi random graphs is provably easy ([19], [8], [27]) and random 3-SAT near the critical threshold looks to be much simpler than initially thought ([25]).

With these other NP-Complete problems in mind, it is reasonable to suspect that the random (uniform) Number Partition Problem might be easy for a proper algorithm, even very near the critical threshold, to solve. Just as the power law degree distributed graphs potentially create instances whose Hamiltonicity is difficult to determine, so too could hard instances be potentially found for the Number Partition Problem where list elements are drawn from some Lévy-Stable distribution.

6.3 Future Work

Numerical evidence has been presented that show there is a natural ensemble of graphs in which finding Hamiltonian Cycles is hard. Care has to be taken about claims of intrinsic difficulty of random ensembles as often problems previously thought to be intractable turn out not to be, even near the critical threshold ([29], [19], [105], [25]). The results of this thesis point to graphs whose degree sequence is generated from the family of Lévy-Stable distributions as being intrinsically difficult but there could be an almost sure polynomial time algorithm to solve graphs drawn from this ensemble. With knowledge that graphs generated in this manner might be intrinsically easy, the following conjecture is proposed:

Conjecture 1. *Graphs chosen from a modified Lévy-Stable distribution with critical exponent* $0 < \alpha < 2$ *have a phase transition whose scale parameter is a function of* γ, α *and* N *in determining Hamiltonicity. Furthermore, in the critical region for* $0 < \alpha < 2$, *there is a non-negligible probability of choosing intrinsically hard instances from this distribution such that determining Hamiltonicity becomes intractable.*

If graphs generated in this manner are intrinsically hard, it would be interesting to find the graph invariant that captures the essence of what makes these graphs difficult. As has been suggested earlier, the degree sequence has drawbacks that do not make it an ideal graph invariant. Perhaps a better graph invariant would be its spectrum. It would be interesting to analyze the phase transition probability of finding Hamiltonian Cycles in graphs chosen from some distribution based on the distribution of eigenvalues rather than degree.

Without better algorithms to probe the space of solutions for the Number Partition Problem, it is difficult to generate numerical evidence. With this in mind, the following conjecture is also proposed:

Conjecture 2. *Instances of the random Number Partition Problem whose elements are drawn from a uniform distribution are intrinsically easy and there exists an almost sure polynomial time algorithm to determine whether there is a perfect partition or not.*

Which leads to the following conjecture:

Conjecture 3. *Number Partition Problem instances whose list elements are chosen from a Lévy-Stable (or power law) distribution with critical exponent* $0 < \alpha < 2$ *have a phase transition whose scale parameter is a function of* γ, α *and* N. *Furthermore, instances created in this way have a non-negligible probability of choosing intrinsically hard instances such that determining if a perfect partition exists becomes intractable.*

Lattice reduction techniques provide a promising avenue of investigation when trying to build approximation algorithms. Perhaps a judicious encoding of the Number Partition Problem in a better lattice reduction algorithm, such as PSLQ, would fare better than what has been presented in this thesis.

Future work could focus on other NP-Complete problem, such as 3-SAT, to determine what parametrization makes instance generation intrinsically hard should one exist in the first place. More importantly, the evidence presented is anecdotal and a better theory needs to be developed to explain why random instance generation is easy for some distributions but hard for others.

6.4 Conclusion

To have an NP-Complete problem that is thought to be exponentially hard to solve in general but not be able to find any random instance that are difficult for solvers in practice would be a conundrum. Numerical evidence has been presented for a class of graphs whose Hamiltonicity appears to be difficult to determine. The analysis depends critically on the solvers used and should be taken only as a candidate of a class of random graphs whose Hamiltonicity is hard to determine. Determining if the class of graphs presented, or another, are intrinsically hard would lead to a better understanding of what hard NP-Complete problem instances look like.

The beginnings of a more thorough analysis in the Number Partition Problem has been presented as well. Without better solvers, the Number Partition Problem will be limited to numerical evidence from extremely small instances without the ability to probe the space in any reasonable way.

It is the hope that this will be a starting point for the search for intrinsically hard instance generation of the Hamiltonian Cycle Problem and others. If it should turn out that other NP-Complete problem also share the property that easy instances are generated when the underlying distribution has a finite second moment, then study of the Hamiltonian Cycle problem

could serve as a guidepost of how to generate harder instances in these other areas.

Bibliography

[1] S. Aaronson. Reasons to believe. http://scottaaronson.com/blog/?p=122, posted Sep. 4th 2006, retrieved Dec. 14th 2010.

[2] D. Achlioptas, P. Beame, and M. Molloy. A sharp threshold in proof complexity. In *Proceedings of the Thirty-Third Annual ACM Symposium on Theory of Computing*, pages 337–346, 2001.

[3] D. Achlioptas and A. Coja-Oghlan. Algorithmic barriers from phase transitions. *FOCS*, pages 794–802, 2008.

[4] D. Achlioptas, A. Naor, and Yuval Peres. Rigorous location of phase transitions in hard optimization problems. *Nature*, 453:759–764, 2005.

[5] L. A. Adamic and B. A. Huberman. Power-law distribution of the world wide web. *Science*, 287:2115, 2000.

[6] L. Adelman. On breaking the iterated merkle-hellman public key cryptosystem. *Proc. 15th ACM Symp. on Theory of Computing*, pages 402–412, 1983.

[7] R. Albert and A. Barabási. Statistical mechanics of complex networks. *Reviews of Modern Physics*, 74:47–97, 2002.

[8] D. Angluin and L. G. Valiant. Fast probabilistic algorithms for Hamilton circuits and matchings. *J. Computer, Syst. Sci.*, 18:155–193, 1979.

[9] P. Bak. *How Nature Works.* Copernicus, an imprint of Springer-Verlag New York, Inc., 1996.

[10] C. Bazgan, M. Santha, and Z. Tuza. Efficient approximation algorithms for the SUBSET-SUMS EQUALITY problem. In *ICALP*, pages 387–396, 1998.

[11] P. Beame, R. Karp, T. Pitassi, and M. Saks. On the complexity of unsatisfiability proofs for random k-CNF formulas. In *Proceedings on the 30th annual ACM symposium on Theory of computing*, pages 561–571. ACM Press, 1998.

[12] E. Ben-Sasson and A. Widgerson. Short proofs are narrow – resolution made simple. *J. ACM*, 48:149–169, 2001.

[13] N. Berger, C. Borgs, J. T. Chayes, R. M. D'Souza, and R. D. Kleinberg. Competition-induced preferential attachment. *Lecture Notes in Computer Science*, 3142:208–221, 2004.

[14] G. Bianconi and M. Marsili. Loops of any size and Hamilton cycles in random scale-free networks. *J. of Statistical Mechanics: Theory and Experiment*, P06005, 2005.

[15] J. Blitzstein and P. Diaconis. A sequential importance sampling algorithm for generating random graphs with prescribed degree. Technical report, Stanford University, 2006.

[16] S. Boettcher and S. Mertens. Analysis of the Karmarkar-Karp differencing algorithm. *The European Physical Journal B - Condensed Matter and Complex Systems*, 65:131–140, 2008.

[17] B. Bollobás. The evolution of sparse graphs. *in Graph Theory and Combinatorics, Proc. Cambridge Combinatorial Conf. in honor of Paul Erdös*, pages 33–57, 1984.

[18] B. Bollobás. *Random Graphs, Second Edition.* Cambridge University Press, New York, 2001.

[19] B. Bollobás, T. I. Fenner, and A. M. Frieze. An algorithm for finding Hamilton paths and cycles in random graphs. *Combinatorica*, 7:327–341, 1987.

[20] J. A. Bondy and U. S. R. Murty. *Graph Theory with Applications.* Elsevier, Amsterdam, 1976.

[21] C. Borgs, J. Chayes, S. Mertens, and C. Nair. Proof of the local REM conjecture for number partitioning I: Constant energy scales. *Random Structures and Algorithms*, 34(2):217–240, 2009.

[22] C. Borgs, J. Chayes, S. Mertens, and C. Nair. Proof of the local REM conjecture for number partitioning II: Growing energy scales. *Random Structures and Algorithms*, 34(2):241–284, 2009.

[23] C. Borgs, J. Chayes, and B. Pittel. Phase transition and finite-size scaling for the integer partition problem. *Random Structures and Algorithms*, 19:247–208, 2001.

[24] P. B. Borwein. *Computational excursions in analysis and number theory*. Springer, New York, 2002.

[25] A. Braunstein, M. Mézard, and R. Zecchina. Survey propagation: an algorithm for satisfiability. *Random Structures and Algorithms*, 27:201–226, 2005.

[26] T. Britton, M. Deijfen, and A. Martin-Löf. Generating simple random graphs with prescribed degree distribution. *J. of Statistical Physics*, 124:1377–1397, 2006.

[27] A. Z. Broder, A. M. Frieze, and E. Shamir. Finding hidden Hamiltonian cycles. *Random Structures and Algorithms*, 5:395–410, 1994.

[28] F. A. Brunacci. DB2 and DB2A: Two useful tools for constructing Hamiltonian circuits. *European Journal of Operational Research*, 34:231–236, 1988.

[29] P. Cheeseman, B. Kanefsky, and W. M. Taylor. Where the really hard problems are. In *IJCAI'91 Proceedings of the 12th international joint conference on Artificial intelligence*, volume 1, pages 331–337, 1991.

[30] K. Christensen and N. R. Moloney. *Complexity and Criticality*. Imperial College Press, London, UK, 2005.

[31] F. Chung, L. Lu, and V. Vu. Eigenvalues of random power law graphs. *Annals of Combinatorics*, 7:21–33, 2003.

[32] F. R. K. Chung. *Spectral Graph Theory*. American Mathematical Society, Providence, Rhode Island, 1994.

[33] F. R. K. Chung and L. Lu. *Complex Graphs and Networks*. American Mathematical Society, Providence, Rhode Island, 2004.

[34] R. Cohen and S. Havlin. Scale-free networks are ultrasmall. *Phys. Rev. Lett.*, 90:3682–3685, 2003.

[35] S. Cook. The P versus NP problem. http://www.claymath.org/milliennium.

[36] T. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms, Third Edition*. The MIT Press, Cambridge, Massachusetts, 2009.

[37] M. J. Coster, A. Joux, B. A. LaMacchia, A. M. Odlyzko, C. Schnorr, and J. Stern. Improved low-density subset sum algorithms. *Computational Complexity*, 2:111–128, 1992.

[38] B. Derrida. Random-energy model: Limit of a family of disordered models. *Phys. Rev. Lett. 45*, pages 79–82, 1980.

[39] B. Derrida. Random-energy model: An exactly solvable model of disordered systems. *Phys. Rev. Lett. 24*, pages 2513–2626, 1981.

[40] R. Diestel. *Graph Theory, Electronic Edition 2005*. Springer, New York, 2005.

[41] I. Eliazar and J. Klafter. On the extreme flights of one-sided lévy processes. *J. Phys. A*, 330:8–17, 2003.

[42] P. Erdös and A. Rényi. On random graphs. *Publ. Math Debrecen*, 6:290–297, 1959.

[43] P. Erdös and A. Rényi. On the evolution of random graphs. *Publ. Inst. Math Hungar. Acad. Sci.*, 5:17–61, 1960.

[44] P. Erdös and A. Rényi. On the evolution of random graphs. *Bull. Inst. Int. Statist. Tokyo*, 38:343–347, 1961.

[45] I. J. Farkas, I. Derényi, A. Barabási, and T. Viscek. Spectra of 'real-world' graphs: Beyond the semicircle law. *Physical Review E.*, 64, 2001.

[46] W. Feller. *An Introduction to Probability Theory and its Applications*, volume 2. John Wiley and Sons, New York, second edition, 1971.

[47] H. R. P. Ferguson, D. H. Bailey, and S. Arno. Analysis of pslq, an integer relation finding algorithm. *RNR: Technical Report RNR-91-032*, 1992.

[48] L. Fortnow. The status of the P versus NP problem. *Communications of the ACM*, 52:78–86, 2009.

[49] E. Friedgut and G. Kalai. Every monotone graph property has a sharp threshold. In *Proceedings of the American Mathematical Society*, volume 124, pages 2993–3002, 1996.

[50] A. M. Frieze. An algorithm for finding Hamilton cycles in random directed graphs. *Journal of Algorithms*, 9:181–204, 1988.

[51] Y. T. Fu and P. W. Anderson Y. T. Application of statistical mechanics to np-complete problems in combinatorial optimization. *J. Phys. A*, 19:1605–1620, 1985.

[52] M. R. Garey and D. S. Johnson. *Computers and Intractability, A Guide to the Theory of NP-Completeness*. W.H. Freeman and Company, San Francisco, 1979.

[53] I. Gent and T. Walsh. Phase transition and annealed theories: Number partitioning as a case study. *12th European Conference on AI*, pages 171–174, 1996.

[54] I. Gent and T. Walsh. The TSP phase transition. *Artificial Intelligence*, 88:349–358, 1996.

[55] I. Gent and T. Walsh. Analysis of heuristics for number partitioning. *Computational Intelligence*, 14:430–451, 1998.

[56] P. M. Gleiss, P. F. Stadler, A. Wagner, and D. A. Fell. Small cycles in small worlds. *Adv. Complex Systems*, 4:207–226, 2001.

[57] O. Goldreich. *Computational Complexity: A Conceptual Perspective*. Cambridge University Press, 32 Avenue of the Americas, New York, NY, 2006.

[58] A. K. Hartmann and M. Weigt. Statistical mechanics perspective on the phase transition of vertex covering of finite-connectivity random graphs. *Theoretical Computer Science*, 265:199–225, 2001.

[59] A. K. Hartmann and M. Weigt. *Phase Transitions in Combinatorial Optimization Problems: Basics, Algorithms and Statistical Mechanics*. Wiley-VCH, 2005.

[60] B. Hayes. The easiest hard problem. *American Scientist*, 90(2), 2002.

[61] B. Hayes. On the threshold. *American Scientist*, 91(1), 2003.

[62] H. J. Jensen. *Self-Organized Criticality: Emergent Complex Behavior in Physical and Biological Systems*. Cambridge University Press, 1998.

[63] N. Karmarkar, R. Karp, J. Lueker, and A. Odlyzko. Probabilistic analysis of optimum partitioning. *J. Appl. Prob.*, 23:626–645, 1986.

[64] H. Kesten. What is percolation? *Notices of the AMS*, 53:572–574, 2006.

[65] Subhash Khot. Hardness of approximating the shortest vector problem in lattices. *Journal of the ACM*, 52:789–808, 2005.

[66] S. Kirkpatrick and B. Selman. Critical behavior in the satisfiability of random boolean expressions. *Science*, 264:1297–1301, 1994.

[67] W. Kocay. An extension of the multi-path algorithm for finding Hamilton cycles. *Discrete Mathematics*, 101:171–188, 1992.

[68] W. Kocay and P. Li. An algorithm for finding a long path in a graph. *Utilitas Mathematica*, 45:169–185, 1994.

[69] J. Komlós and E. Szemerédi. Limit distributions for the existence of Hamilton cycles in a random graph. *Discrete Math*, 43:55–63, 1983.

[70] R. Korf. From approximate to optimal solutions: A case study of number partitioning. In *Proceedings of the 14th IJCAI*, pages 266–272, 1995.

[71] A. D. Korshunov. A solution of a problem of p. Erdös and a. Rényi about Hamilton cycles in non-oriented graphs. *metody Diskr. Anal. Teoriy Ubr. Syst. Sh. Trudov Novasibirisk*, 31:17–56, 1977.

[72] D. C. Kozen. *Automata and Computability.* Springer, New York, NY, 1997.

[73] J. C. Lagarias and A. M. Odlyzko. Solving low-density subset sum problems. *J. of the Ass. for Computing Machinery*, 32:229–246, 1983.

[74] A.K. Lenstra, H. W. Lenstra Jr., and L. Lovasz. Factoring polynomials with rational coefficients. *Math. Ann.*, 261:515–534, 1982.

[75] C. M. Lie and S. Gérard. On the limit of branching rules for hard random unsatisfiable 3-SAT. In *Proceedings of 14th European Conference on Artificial Intelligence*, Berlin, Germany, 2000.

[76] S. Martello. Algorithm 595: An enumerative algorithm for finding Hamiltonian circuits in a directed graph. *ACM Transactions on Mathematical Software*, 9:131–138, 1983.

[77] O. C. Martin, R. Monasson, and R. Zecchina. Statistical mechanics methods and phase transitions in optimization problems. *Theoretical Computer Science*, 265:3–67, 2001.

[78] S. Merkle and M. Hellman. Hiding information and signatures in trapdoor knapsacks. *Information Theory IEEE Transactions*, 24 no. 5:525–530, 1978.

[79] S. Mertens. Phase transition in the number partition problem. *Physical Review Letters*, 81:4281–4284, 1998.

[80] S. Mertens, M. Mézard, and R. Zecchina. Threshold values of random K-SAT from the cavity method. *Random Structures and Algorithms*, 28:340.

[81] M. Mézard and A. Monatanari. *Information, Physics and Computation.* Oxford University Press, 2009.

[82] M. Mézard, G. Parisi, and R. Zecchina. Analytic and algorithmic solution of random satisfiability problems. *Science*, 297:812–815, 2002.

[83] M. Mihail and C. Papadimitriou. On the eigenvalue power law. In *Proceedings of RANDOM 2002*, pages 254–262. Berlin-Heidelberg: Springer-Verlag, 2002.

[84] D. Mitchell, B. Selman, and H. J. Levesque. A new method for solving hard satisfiability problems. In *Proceedings of the Tenth National Conference on Artificial Intelligence (AAAI-92)*, pages 440–446, 1992.

[85] M. Molloy and B. Reed. A critical point for random graphs with a given degree sequence. *Random Struct. Algorithms*, 6:161, 1995.

[86] R. Monasson, R. Zecchina, S. Kirkpatrick, B. Selman, and L. Troyansky. Determining computation complexity from characteristic 'phase transitions'. *Nature*, 400:133, 1999.

[87] M. E. J. Newman. The structure and function of complex networks. *SIAM Review*, 45:251–262, 1999.

[88] J. P. Nolan. *Stable Distributions, Models for Heavy Tailed Data, Chapter 1.* Unpublished, http://academic2.american.edu/ jpnolan/stable/chap1.pdf, 2000.

[89] C. H. Papadimitriou. On the complexity of the parity argument and other inefficient proofs of existence. *J. of Computer And System Science*, 48:498–532, 1994.

[90] C. H. Papadimitriou. *Computational Complexity.* Addison Wesley Publishing Co., Reading, MA, 1995.

[91] M. Perman. Order statistics for jumps of normalized subordinators. *Stochastic Processes and their Applications*, 46:267–281, 1993.

[92] J. Pitman and M. Yor. The two-parameter poisson-dirichlet distribution derived from a stable subordinator. *University of California Technical Report No. 433*, 1995.

[93] L. Pósa. Hamiltonian circuits in random graphs. *Discrete Mathematics*, 14:358–364, 1976.

[94] W. J. Reed and B. D. Hughes. From gene families and genera to incomes and internet file sizes: Why power laws are so common in nature. *Physical Review E*, 66:067103–1–067103–4, 2002.

[95] C. P. Schnorr and M. Euchner. Lattice basis reduction: Improved practical algorithms and solving subset sum problems. In *Proceedings of Fundamentals of Computation Theory '91, L. Budach, ed., Lecture Notes in Computer Science*, volume 529, pages 68–85, 1991.

[96] E. Shamir. How many random edges make a graph Hamiltonian. *Combinatorica*, 3:123–132, 1983.

[97] J. A. Shufelt and H. J. Berliner. Generating Hamiltonian circuits without backtracking from errors. *Theoretical Computer Science*, 132:347–375, 1994.

[98] J. Spencer. *The Strange Logic of Random Graphs*. Springer, 2000.

[99] D. Stauffer and Amnon Aharony. *Introduction to Percolation Theory*. CRC Press, 2000 N.W. Corporate Blvd., Boca Raton, Florida 33431, 1994.

[100] A. Thomason. A simple linear expected time algorithm for finding a Hamilton path. *Discrete Mathematics*, 75:373–379, 1989.

[101] H. van den Esker, R. van der Hofstad, G. Hooghiemstra, and D. Znamenski. Distances in random graphs with infinite mean degrees. *Extremes*, 8:111–141, 2006.

[102] R. van der Hofstad, G. Hooghiemstra, and P. van Mieghem. Distances in random graphs with finite variance degrees. *Random Structures and Algorithms*, 27:76–123, 2005.

[103] R. van der Hofstad, G. Hooghiemstra, and D. Znamenski. Distances in random graphs with finite mean and infinite variance degrees. *Electronic Journal of Probability*, 12:703–766, 2007.

[104] B. Vandegriend. Finding Hamiltonian cycles: Algorithms, graphs and performance. Master's thesis, University of Alberta, Dept. of Computer Science, 1998.

[105] B. Vandegriend and J. Culberson. The $G_{n,m}$ phase transition is not hard for the Hamiltonian cycle problem. *J. of AI Research*, 9:219–245, 1998.

[106] C. K. Yap. *Fundamental Problems of Algorithmic Algebra*. Oxford University Press, New York, 2000.

[107] M. Zimand. *Computation Complexity: A Quantitative Perspective*. Elsevier, Sara Burgerhartstraat 25, P.O. Box 211, 1000 AE Amsterdam, The Netherlands, 2004.

[108] V. M. Zolotarev. *One-dimensional Stable Distributions*. American Mathematical Society, 1983.

Appendix A

LLL Worked Example

A small basis is provided as input the LLL algorithms and the example run is provided below. For convenience all vectors are taken to be row vectors.

Our example basis, $\mathbf{B}$, is as follows:

$$\mathbf{B} = \begin{bmatrix} b_0 \\ b_1 \\ b_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 130 \\ 0 & 1 & 0 & 240 \\ 0 & 0 & 1 & 440 \end{bmatrix}$$

Initially, the matrices as a result of the Gram-Schmidt orthogonalization step, $\mathbf{M}$ and $\mathbf{B}^*$ are:

$$\mathbf{B} = \mathbf{M}\mathbf{B}^* = \begin{bmatrix} 1 & 0 & 0 \\ \frac{31200}{16901} & 1 & 0 \\ \frac{57200}{16901} & \frac{105600}{74501} & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 130 \\ -\frac{31200}{16901} & 1 & 0 & \frac{240}{16901} \\ -\frac{57200}{74501} & -\frac{105600}{74501} & 1 & \frac{440}{74501} \end{bmatrix}$$

with

$$|b_0^*|^2 = 16901$$

$$|b_1^*|^2 = \frac{74501}{16901}$$

$$|b_2^*|^2 = \frac{268101}{74501}$$

Initially, a **weak reduction** step is performed, to reduce the lower triangular elements in the M array, via a series of integer linear combinations, to be absolutely less than or equal to $1/2$. i.e. $|\mu_{i,j}| \leq (1/2)$ for $i < j$. This results in:

$$\begin{bmatrix} 1 & 0 & 0 & 130 \\ -2 & 1 & 0 & -20 \\ -2 & -1 & 1 & -60 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ -\frac{2602}{16901} & 1 & 0 \\ -\frac{7802}{16901} & \frac{31099}{74501} & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 130 \\ -\frac{31200}{16901} & 1 & 0 & \frac{240}{16901} \\ -\frac{57200}{74501} & -\frac{105600}{74501} & 1 & \frac{440}{74501} \end{bmatrix}$$

Since a weak reduction step leaves $\mathbf{B}^*$ unaltered, $|b_k^*|^2$ ($k \in \{0, 1, 2\}$) need not be changed. Recalling that a reduced basis is one where $|b_i^*|^2 \leq 2|b_i^*|^2$ for $0 \leq i < n$, we notice that for the basis above:

$$|b_0^*|^2 = 16901 > 2|b_1^*|^2 = (2)(\frac{74501}{16901})$$

Here, b_0^* fails the reduction criteria and is a candidate for the **strong reduction** step. Thus we swap b_0 with b_1 resulting in:

$$\begin{bmatrix} -2 & 1 & 0 & -20 \\ 1 & 0 & 0 & 130 \\ -2 & -1 & 1 & -60 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ -\frac{2602}{405} & 1 & 0 \\ \frac{401}{135} & -\frac{29604}{74501} & 1 \end{bmatrix} \begin{bmatrix} -2 & 1 & 0 & -20 \\ -\frac{4799}{405} & \frac{2602}{405} & 0 & \frac{122}{81} \\ -\frac{57200}{74501} & -\frac{105600}{74501} & 1 & \frac{440}{74501} \end{bmatrix}$$

$$|b_0^*|^2 = 405$$

$$|b_1^*|^2 = \frac{74501}{405}$$

$$|b_2^*|^2 = \frac{268101}{74501}$$

A **weak reduction** step is performed, resulting in:

$$\begin{bmatrix} -2 & 1 & 0 & -20 \\ -11 & 6 & 0 & 10 \\ 4 & -4 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ -\frac{172}{405} & 1 & 0 \\ -\frac{4}{135} & -\frac{29604}{74501} & 1 \end{bmatrix} \begin{bmatrix} -2 & 1 & 0 & -20 \\ -\frac{4799}{405} & \frac{2602}{405} & 0 & \frac{122}{81} \\ -\frac{57200}{74501} & -\frac{105600}{74501} & 1 & \frac{440}{74501} \end{bmatrix}$$

We choose b_0 and b_1 as the basis vectors for the strong reduction step as b_0^* is again in violation of the reduction criteria:

$$|b_0^*|^2 = 405 > 2|b_1^*|^2 = (2)(\frac{74501}{405})$$

After a **strong reduction** step:

$$\begin{bmatrix} -11 & 6 & 0 & 10 \\ -2 & 1 & 0 & -20 \\ 4 & -4 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ -\frac{172}{257} & 1 & 0 \\ -\frac{68}{257} & -\frac{14780}{74501} & 1 \end{bmatrix} \begin{bmatrix} -11 & 6 & 0 & 10 \\ -\frac{2406}{257} & \frac{1289}{257} & 0 & -\frac{3420}{257} \\ -\frac{57200}{74501} & -\frac{105600}{74501} & 1 & \frac{440}{74501} \end{bmatrix}$$

$$|b_0^*|^2 = 257$$

$$|b_1^*|^2 = \frac{74501}{257}$$

$$|b_2^*|^2 = \frac{268101}{74501}$$

After a **weak reduction** step is performed, we have:

$$\begin{bmatrix} -11 & 6 & 0 & 10 \\ -13 & 7 & 0 & -10 \\ 4 & -4 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ \frac{85}{257} & 1 & 0 \\ -\frac{68}{257} & -\frac{14780}{74501} & 1 \end{bmatrix} \begin{bmatrix} -11 & 6 & 0 & 10 \\ -\frac{2406}{257} & \frac{1289}{257} & 0 & -\frac{3420}{257} \\ -\frac{57200}{74501} & -\frac{105600}{74501} & 1 & \frac{440}{74501} \end{bmatrix}$$

We now choose b_1 and b_2 as the two basis vectors for the strong reduction step, as b_1^* and b_2^* now violate the reduction criteria:

$$|b_1^*|^2 = \frac{74501}{257} > 2|b_2^*|^2 = (2)(\frac{268101}{74501})$$

After a **strong reduction** step of swapping b_1 with b_2, the result is:

$$\begin{bmatrix} -11 & 6 & 0 & 10 \\ 4 & -4 & 1 & 0 \\ -13 & 7 & 0 & -10 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ -\frac{68}{257} & 1 & 0 \\ \frac{85}{257} & -\frac{14780}{3857} & 1 \end{bmatrix} \begin{bmatrix} -11 & 6 & 0 & 10 \\ \frac{280}{257} & -\frac{620}{257} & 1 & \frac{680}{257} \\ -\frac{2858}{551} & -\frac{16311}{3857} & \frac{14780}{3857} & -\frac{12220}{3857} \end{bmatrix}$$

$$|b_0^*|^2 = 257$$

$$|b_1^*|^2 = \frac{3857}{257}$$

$$|b_2^*|^2 = \frac{268101}{3857}$$

A **weak reduction** is performed resulting in:

$$\begin{bmatrix} -11 & 6 & 0 & 10 \\ 4 & -4 & 1 & 0 \\ -8 & -3 & 4 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ -\frac{68}{257} & 1 & 0 \\ \frac{70}{257} & \frac{648}{3857} & 1 \end{bmatrix} \begin{bmatrix} -11 & 6 & 0 & 10 \\ \frac{280}{257} & -\frac{620}{257} & 1 & \frac{680}{257} \\ -\frac{2858}{551} & -\frac{16311}{3857} & \frac{14780}{3857} & -\frac{12220}{3857} \end{bmatrix}$$

b_0 and b_1 are swapped for the **strong reduction** step as:

$$|b_0^*|^2 = 257 > 2|b_1^*|^2 = (2)(\frac{3857}{257})$$

resulting in:

$$\begin{bmatrix} 4 & -4 & 1 & 0 \\ -11 & 6 & 0 & 10 \\ -8 & -3 & 4 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ -\frac{68}{33} & 1 & 0 \\ -\frac{16}{33} & \frac{1222}{3857} & 1 \end{bmatrix} \begin{bmatrix} 4 & -4 & 1 & 0 \\ -\frac{91}{33} & -\frac{74}{33} & \frac{68}{33} & 10 \\ -\frac{2858}{551} & -\frac{16311}{3857} & \frac{14780}{3857} & -\frac{12220}{3857} \end{bmatrix}$$

$$|b_0^*|^2 = 33$$

$$|b_1^*|^2 = \frac{3857}{33}$$

$$|b_2^*|^2 = \frac{268101}{3857}$$

A **weak reduction** step is performed:

$$\begin{bmatrix} 4 & -4 & 1 & 0 \\ -3 & -2 & 2 & 10 \\ -8 & -3 & 4 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ -\frac{2}{33} & 1 & 0 \\ -\frac{16}{33} & \frac{1222}{3857} & 1 \end{bmatrix} \begin{bmatrix} 4 & -4 & 1 & 0 \\ -\frac{91}{33} & -\frac{74}{33} & \frac{68}{33} & 10 \\ -\frac{2858}{551} & -\frac{16311}{3857} & \frac{14780}{3857} & -\frac{12220}{3857} \end{bmatrix}$$

This basis is now reduced as:

$$|b_0^*|^2 = 33 \leq 2|b_1^*|^2 = (2)(\frac{3857}{33})$$

and

$$|b_1^*| = \frac{3857}{33} \leq 2|b_2^*| = (2)\frac{268101}{3857}$$

resulting in the final reduced basis:

$$\begin{bmatrix} 4 & -4 & 1 & 0 \\ -3 & -2 & 2 & 10 \\ -8 & -3 & 4 & 0 \end{bmatrix}$$