1-1-2011

Factoring Semiprimes Using PG2N Prime Graph Multiagent Search

Keith Eirik Wilson
Portland State University

Factoring Semiprimes Using PG_N^2 Prime Graph Multiagent Search

by

Keith Eirik Wilson

A thesis submitted in partial fulfillment of the
requirements for the degree of

Master of Science
in
Computer Science

Thesis Committee:
Bryant York, Chair
Bart Massey
Thomas Shrimpton

Portland State University

Abstract

In this thesis a heuristic method for factoring semiprimes by multiagent depth-limited search of PG_N^2 graphs is presented. An analysis of PG_N^2 graph connectivity is used to generate heuristics for multiagent search. Further analysis is presented including the requirements on choosing prime numbers to generate 'hard' semiprimes; the lack of connectivity in PG_N^1 graphs; the counts of spanning trees in PG_N^2 graphs; the upper bound of a PG_N^2 graph diameter and a conjecture on the frequency distribution of prime numbers on Hamming distance.

We further demonstrated the feasibility of the HD2 breadth first search of PG_N^2 graphs for factoring small semiprimes. We presented the performance of different multiagent search heuristics in PG_N^2 graphs showing that the heuristic of most connected seedpick outperforms least connected or random connected seedpick heuristics on small PG_N^2 graphs of size $N \leq 26$. The contribution of this small scale research was to develop heuristics for seed selection that may extrapolate to larger values of N.

This Thesis is Dedicated to Paula Holm Jensen

Acknowledgements

I would like to thank Dr. Bryant York, my adviser these past few years for his patience and ability to clearly explain mathematical ideas.

I would also like to thank Dr. Bart Massey and Dr. Thomas Shrimpton for serving on the thesis committee. Their enthusiasm and insightful comments brought a focus to this work which I would not have found on my own.

In addition I'd like to acknowledge the contributions of the students who have worked on this material with Dr. York over the years; Chandler York, Qing Yi, Adam Ingram-Goble, Yan Chen and David Rosenbaum.

I am glad to have had the opportunity to carry this research a little further forward.

I would also like to thank Paula Holm Jensen for her support and encouragement.

Others who have supported me in this work include Dr. William Black of Colorado State University for his help with statistical theory; Alan Bahm, Jamie Sharp, Josh Triplett for their curiosity and questions about prime number graphs and all my friends who have patiently listened to me talk about 'that prime number thing' over the past year.

Table of Contents

List of Tables

List of Figures

1 Introduction

> *"God does not play dice with the universe, but something strange is going on with the prime numbers."* -Paul Erdős [16]

Counting is arguably one of the most enthusiastic and earliest mathematical activities in human history. We count in music, in trade, in games asking how many?–how long?–how far?–how often? And other questions too many to enumerate. The counts of things are represented by whole numbers, called integers. Someone, somewhere, started asking questions about counting and integers: how many?–how long?–how far?–how often? One of these questions about the integers remains a puzzle to this day: Is there a pattern to the prime numbers?

Most integers are equal to some set of smaller integers multiplied together. We call these smaller integers factors. Some integers are not the result of multiplying any set of smaller integers. These integers are called prime numbers. As it turns out, all integers that are not prime numbers can be found by multiplying together two or more prime numbers. Each of these prime numbers is called a prime factor of an integer. This discovery has a name called *The Fundamental Theorem of Arithmetic.* (See Gauss [8, p.6])

In this thesis we will define and discuss two graph structures, integer graphs and prime number graphs. These are graphs constructed by an adjacency relation based on Hamming distance between the binary representation of the vertex sets.

This thesis describes and discusses the performance of an algorithm for finding the factors of integers comprising exactly two prime numbers. These integers are called semiprimes. The method used by this algorithm is computa-

tional graph search over prime number graphs. In this thesis we are concerned with the problem of factoring semiprimes using multiagent parallel neighborhood graph search. We present heuristics for choosing the start point or seed for each neighborhood search; the connectivity analysis motivating these heuristics and the experimental results using multiagent neighborhood search to find factors of semiprimes.

In the remainder of this section we will summarize the definitions and symbols used throughout this work. In section 2 we cover the background ideas used in this research; in section 3 the search methodology and the multiagent search algorithms at the core of this thesis. In section 4 we describe our experimental results; in section 5 we present our conclusions and in section 6 our plan for future work.

1.1 Hamming distance (HD)

Hamming distance[9] is a metric from information theory. The Hamming distance (HD) between two strings is the number of positions in which the bit strings differ. It is a method to calculate how *far* one string is from another.

A geometric interpretation of Hamming distance can be seen in the example of the binary hypercube. In Figure 1 each of the vertices is assigned a three digit binary number. The numbers are assigned in such a way that each vertex is $\mathrm{HD} = 1$, away from every adjacent vertex. Vertex **101** is HD=3 away from vertex **010**.

A step of $\mathrm{HD} = 1$ is equivalent to traversing an edge of the hypercube. A path between **101** and **010**, stepping by $\mathrm{HD} = 1$ is shown in red in Figure 2.

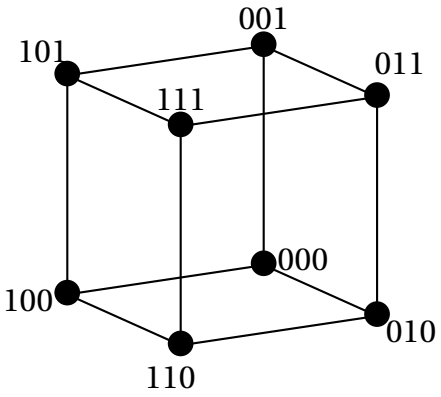

Figure 1: Three Dimensional Binary Hypercube[19]

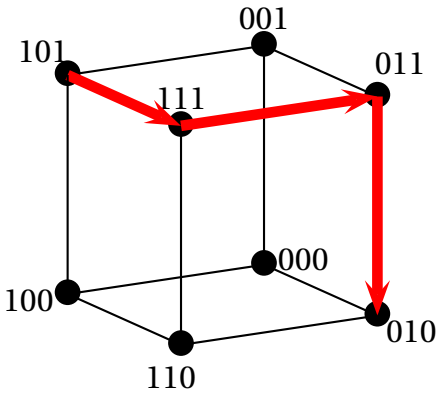

Figure 2: Path in steps of HD $=$ 1[19]

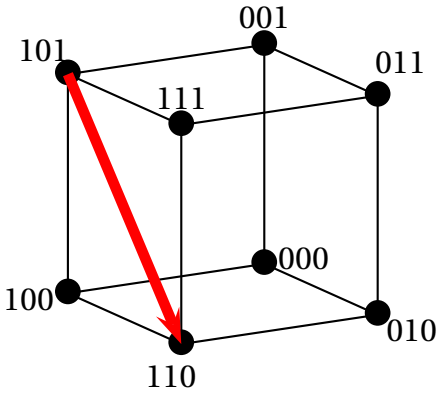

Figure 3: Step of HD $=$ 2[19]

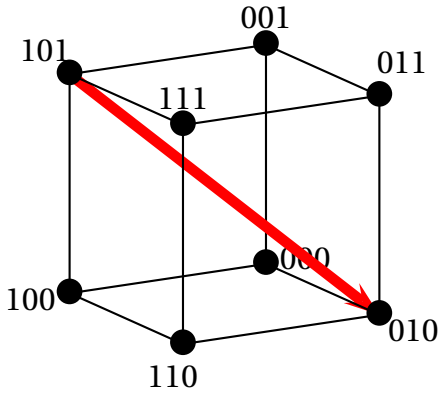

Figure 4: Step of HD $=$ 3[19]

However, if the step is HD $=$ 2 then each step traverses a plane on the hypercube. For instance traveling between **101** and **110**, stepping by HD $=$ 2 is shown in red in Figure 3.

If a step is HD $=$ 3 then each step traverses through the volume of the hypercube as shown in Figure 3 between points **101** and **010**.

In higher dimensions, similar arguments are made. Using a gray code, an n-digit binary number is assigned to each vertex of an n-dimensional binary hypercube.

1.2 Definitions

A **semiprime,** $S = p_1 p_2$, is a **composite** integer comprising two prime numbers (p_1 and p_2) multiplied together.

There are several abbreviations regarding Hamming distance.

- HD refers to the concept of Hamming distance.
- $\mathrm{HD1}$ refers to a Hamming distance = 1.
- $\mathrm{HD2}$ refers to a Hamming distance = 2.
- $\mathrm{HD2}^*$ refers to a Hamming distance ≤ 2.

A more detailed discussion of **Hamming distance** appears in section 1.1.

A **graph** is a mathematical structure defined as an ordered pair $G = (V, E)$ of disjoint sets, V and E. Set V is the set of vertices or nodes in the graph and E is the set of edges connecting vertices as unordered pairs from the set $V \times V$.

In a graph each pair of connected vertices (a, b) can be either ordered or unordered. In a **directed** graph each pair of vertices is ordered with an edge in the direction **from** a **to** b. The edges between connected vertices in a graph are also called arrows. In an **undirected** graph each pair of vertices is unordered, and the edges have no direction.

A **path** in a graph is a sequence of vertices where each vertex has an edge connecting it to the next vertex in the sequence.

A **complete** graph is a graph in which every vertex is *adjacent* to every other vertex in the graph — i.e. connected to every other vertex by a path of length 1. In

a **connected** graph there exists a path (not necessarily of length 1) from every vertex to every other vertex in the graph. An **unconnected** graph is a graph which is not connected — i.e. there exists at least one pair of vertices ($v1$ and $v2$) such that there is no path (sequence of edges in the graph) connecting $v1$ and $v2$. A **connected component** is a subset of the vertices and subset of the edges in a graph that form a connected graph. The number of connected components in an unconnected graph is at least two and is less than or equal to the number of vertices in the graph.

The **degree** of a vertex in an undirected graph is the number of vertices to which it is connected.

> "On a graph G, the distance between two points is the length of a shortest path joining them. If no points join them then the distance is $\inf$. For a connected graph, for all points u, v and w,
>
> 1. $d(u,v) \geq 0,\ with\ d(u,v) = 0 \iff u = v$
> 2. $d(u,v) = d(v,u)$
> 3. $d(u,v) + d(v,w) = d(u,w)$" (Harary [11])

The **diameter** d(G) of a connected graph G is the length of the longest shortest path between two vertices.

In this thesis we use integer graphs, prime number graphs and associated notation developed by Dr. Bryant York[22]. They are defined here for reference.

1.2.1 Integer Graphs

ZG_N^k denotes the graph formed by connecting every integer in the open interval $(2, 2^N)$ with every other integer that is within Hamming distance k.

An integer graph ZG_N^k is a graph:

$ZG_N^k = (V, E)$

where:

$k, N \in \mathbb{Z},\ k < N$

$V = \left\{v \mid v \in \left(2, 2^N\right),\ v \in \mathbb{Z}\right\}$

$E = \{(a, b) \mid a \in V,\ b \in V,\ 0 < Hamming\ distance(a, b) \leq k\}$

Example: ZG_{20}^2 is the graph of integers on the interval $(2, 2^{20})$ connected by a Hamming distance ≤ 2.

1.2.2 Prime Number Graphs

PG_N^k denotes the graph formed by connecting every prime number in the open interval $(2, 2^N)$ with every other prime that is within Hamming distance k.

A prime number graph PG_N^k is a graph:

$PG_N^k = (V, E)$

where:

$k, N \in \mathbb{Z}, ; k < N$

$V = \left\{v \mid v \in \left(2, 2^N\right), v\ is\ prime.\right\}$

$E = \{(a, b) \mid a \in V,\ b \in V,\ 0 < Hamming\ distance(a, b) \leq k\ \}$

Example: PG_{20}^2 is the graph of prime numbers on the interval $(2, 2^{20})$ connected by a Hamming distance ≤ 2.

1.2.3 Additional Notation and Summary Table

The $\mathrm{HD2}^*$ **Corona** is the first ply of prime numbers connected to a prime in a PG_N^2 graph. The **Prime Index**, i, is the index of the prime in the sequence of prime numbers. For example, prime number **3** has index **1**.

The probability of search success is γ (gamma) which is the result of searching a PG_N^k graph for a factor of a semiprime. Gamma is the number of successful factor searches divided by the number of attempted factor searches.

Table 1: Summary of Notation

Symbol	Definition
m	Number of Agents
γ	Probability of Search Success
S	The Semiprime to be Factored
S_{csq}	The Ceiling of the Square Root of S
S_{fsq}	The Floor of the Square Root of S
N	Number of Bits in S_{csq}: $N = log_2\ S_{csq}$
$D(G)$	Graph Diameter
D_{ub}	PG_N^2 Graph Diameter Upper Bound
P_i	The Prime Number at Prime Index i
HD	Hamming distance
HD2*	Hamming distance ≤ 2
ZG_N^k	Graph of Integers on the open interval $\left(2, 2^N\right)\ and\ \mathrm{HD} = k$
PG_N^k	Graph of Prime Numbers on the open interval $\left(2, 2^N\right)\ \ and\ \mathrm{HD} = k$

1.3 Previous Work on Prime Number Graphs and Factoring

Professor Bryant York has been researching prime number graphs for over a decade. The people who have worked with Professor York and their contributions are listed here in chronological order.

- The idea of HD Prime Number Graphs was invented by Professor Bryant York in 1999 while homeschooling his son, Chandler York.
- The initial histograms of Hamming distances of all prime numbers in $(2, 2^{10})$ from a given prime in $(2, 2^{10})$ were generated by Chandler York in an APL program in 2000.
- In 2000 Professor York wrote the first HD BFS search in APL and factored many small semiprimes $(< 25\ bits)$.
- In 2002 Professor York rewrote the HD BFS search in Lisp to take advantage of bignums.
- In 2002 Qing Yi (a graduate student of Professor York) converted Professor York's HD BFS search code to C and verified the factoring of small semiprimes $(< 25\ bits)$.
- In 2004 Adam Ingram-Goble (a graduate student of Professor York) re-implemented Qing Yi's C code in Java. He was unable to demonstrate that this code factored small semiprimes successfully.
- In 2006 Yan Chen (while a graduate student at PSU) generated the first set of small challenge numbers. Yan Chen rewrote Qing Yi's C code using the gmp library and also rewrote it in R. He verified that the technique factored small semiprimes $(< 25\ bits)$. Yan Chen also suggested the use of Bloom filter as the hashing mechanism and implemented it.
- In 2007 Yan Chen conducted the first analysis of RSA numbers in terms of HD1 and HD2 coronas.
- In 2009 David Rosenbaum (while an undergraduate at PSU) implemented a parallel HDBFS in lisp on the multiprocessor Sun Solaris machine, but did not complete factoring experiments.

1.4 Related Work

Contemporary factoring techniques rely on analytic methods based on the mathematics of number theory. They are much more mature and effective than factoring by graph search. Factoring by graph search has been introduced as a novel method of factoring which with further research may become useful as a factoring method or provide insights into the PG_N^k graph family.

Two contemporary methods of factoring are considered sub-exponential methods (graph search is an exponential method). Examples of the first methods are the Quadratic Sieve (QS) and Number Field Sieves (NFS) [18][6] methods. The second is the Elliptic Curve Method [14]. In addition to these sub-exponential methods there are Monte Carlo methods such as the Pollard rho method[17] for factoring.

1.4.1 Number Field Sieves

One way to find a prime factor of an integer *n*, is to divide *n* by all the prime numbers less than the square root of *n*. Now the problem is to find all these prime numbers. Sieving is one way to do this. For example, the sieve of Eratosthenes is a method of finding primes starting from the prime number 2. First 'write down' all the integers from 2 to some maximum integer (such as less than the square root of *n*). Then circle 2 and cross off all the multiples of 2 (4, 6, 8 and so on). Then circle the first unmarked number, 3, and cross off all the multiples of 3. Find the next unmarked number, circle it and cross off all of its multiples. Continue until there are no more unmarked numbers. The circled numbers are the prime numbers.

This could be a lengthy procedure if n is a very large integer. We may have to check all the prime numbers less than the square root of n. What if we only needed to check some subset of these prime numbers, some subset less than Y? An integer is called *Y-smooth* if it is a multiple of prime numbers less than some number Y.

Sieving methods such as the Quadratic Sieve and Number Field Sieves begin by trying to find a value for Y that is as small as possible and significantly less than the square root of a number n to reduce the number of candidates to test as a factor. Other techniques are applied to further reduce the size of the set of factors less than Y. An analogy for sieving is to imagine we are pushing the integers less than the square root of n through finer and finer sieves until relatively few prime numbers remain for trial division of n.

For a discussion of sieving methods methods please see Crandall and Pomerance [6, p261] and Pomerance [18].

1.4.2 Elliptic Curve Method

Another way to factor an integer is to use the Elliptic Curve Method (ECM). [10] [23]

"The **Lenstra elliptic curve factorization** (Lenstra [14]) or the **elliptic curve factorization method (ECM)** is a fast, sub-exponential running time algorithm for integer factorization which employs elliptic curves. For general purpose factoring, ECM is the third-fastest known factoring method. The second fastest is the multiple polynomial quadratic sieve and the fastest is the general number field sieve. It is named after Hendrik Lenstra." [21]

The elliptic curve method works using the following basic idea. For a given prime number *p* and the set of integers modulo *p*, define a field (F_p) with addition and multiplication defined modulo *p*. An elliptic curve E over F_p is defined by an equation of the form $y^2 = x^3 + ax + b$, where $a, b \in F_p$ satisfy $4a^3 + 27b^2 \neq 0 \pmod{p}$. A pair (x, y) where $x, y \in F_p$ is a point on the curve where (x, y) satisfies the equation $y^2 = x^3 + ax + b$. The point at infinity, ∞, is also said to be on the curve. The set of all points on E is denoted by $E(F_p)$ and they form an additive Abelian group with a suitable addition defined, with ∞ serving as the identity element.

Lenstra elliptic curve factorization (to factor an integer n) works by picking a random elliptic curve over Z/nZ of the form $y^2 = x^3 + ax + b \pmod{n}$; then picking a non-trivial point $P = (x, y)$ on the curve with random non-zero coordinates. Next, pick a random non-zero $a \pmod{n}$ and compute $b = y^2 - x^3 - ax \pmod{n}$. The next step of the algorithm is to compute certain k multiples of P using the elliptic curve group addition rule. The formulas for the group addition of two points P and Q on the elliptic curve effectively encode the "slope" of the line joining P and Q and thus involve division between residue classes modulo n. This division is implemented by an extended Euclidean algorithm for the greatest common divisor computation. If the result is a slope of the form $\frac{u}{v}$ where $gcd(u, n) = 1$ and $v = 0 \pmod{n}$, this means that the result of addition is the point at infinity and thus the elliptic curve is **not** a group $\pmod{n}$. More importantly, $gcd(v, n)$ is a non-trivial factor of n. [23]

This algorithm basically works by trying random elliptic curves and starting points until the above condition is met.

ECM differs in complexity from Number Field Sieves in that the complexity is

related primarily to the size of the least prime factor of the number we are attempting to factor, and only weakly on the size of the number itself [6, p335]. For further explanation please see Hankerson et al. [10] or Crandall and Pomerance [6].

1.4.3 Pollard rho method for factoring

There are also "heuristic methods using deterministic sequences"[6][17] where a random function f is created from a set $S \rightarrow S$. A sequence is created by iterating the function a number of times. Since the set S is finite, a repetition or cycle in the sequence eventually appears, even if the initial seed of the function is random. Choice of the function and length of the iteration aims to create a sequence that reveals a non-trivial factor of some number n. Again, Crandall and Pomerance [6, p229] has an excellent discussion of these types of Monte Carlo methods.

1.4.4 Hamming Graphs

The author conducted an extensive literature search and was unable to find reference to the graph structures for PG_N^k or ZG_N^k presented herein.

The nearest idea was that of Hamming graphs as defined in Jamison and Matthews [13]. The authors define Hamming graphs as "Cartesian powers of complete graphs." Their notation is to "...let $H(q,n)$ denote the n-fold Cartesian product of a complete graph K_q with itself...$H(q,n)$ is an *n*-tuple whose entries come from a fixed set of *q* symbols-the vertices of K_q. Two words are adjacent if and only if they differ in exactly one place."

The author has extended the idea to integer Hamming graphs, similar to $H(2,n)$

in the notation of Jamison and Matthews [13]. The distinction between $H(2,n)$ graphs and ZG_N^k is that ZG_N^k graphs are not complete as is required of $H(2,n)$ graphs. The author adopted the notation, ZG_N^k, for the family of integer graphs with adjacency relation $\mathrm{HD} \leq k$ for consistency with Professor York's PG_N^k notation.

2 Background

This section presents some background ideas for multiagent prime number graph search. Several topics are covered including a deeper discussion of prime graphs with examples; a discussion of spanning trees in graphs and why they are important in this work; a discussion of the Prime Number Theorem and how we will apply it; a conjecture on the frequency distribution of prime numbers by Hamming distance, a conjecture on the connectivity of PG_N^2 graphs and a proof of the upper bound of the diameter of PG_N^2 graphs.

2.1 Prime Number Graphs

Every PG_N^2 graph is non-empty and unique due to Bertrand's postulate[7][1] which states that if n is an integer, $n > 3$, then there always exists at least one prime number $p,\ n < p < 2n-2$. Since each prime number graph adds prime numbers from an interval $(2^N, 2^{N+1})$ and $(2N-2-N) < (2^{N+1}-2^N)$ at least one prime is added to each successive PG_N^2 graph.

Figures 5 and 6 are examples of PG_5^1 and PG_5^2 [24]

All PG_N^1 graphs are subgraphs of PG_N^2 graphs. The qualitative difference is that PG_N^2 graphs have many more edges. Figure 7[24] shows the adjacency matrices for the PG_5^1 and PG_5^2 graphs as an example.

[1] Proven by Chebyshev as the Bertrand-Chebyshev theorem in 1850 and by Ramanujan in 1919.

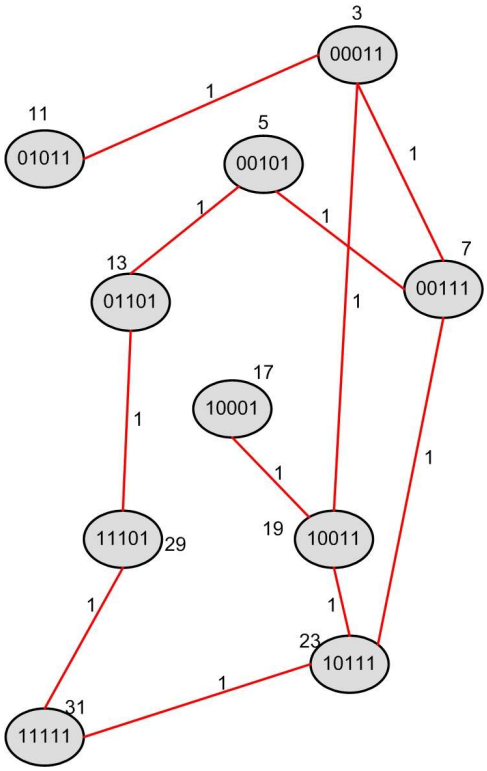

Figure 5: PG_5^1 [19]

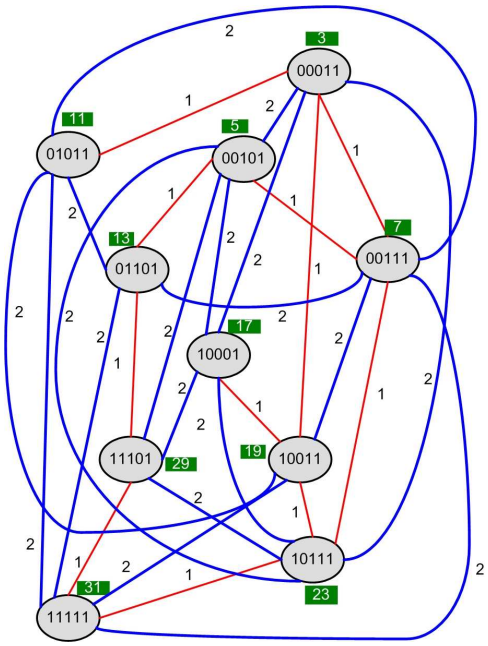

Figure 6: PG_5^2 [19]

$$PG_5^1$$

$$A = \begin{bmatrix} 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \end{bmatrix}$$

$$PG_5^2$$

$$A = \begin{bmatrix} 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 \end{bmatrix}$$

Figure 7: Adjacency Matrix for PG_5^1 and PG_5^2 [19]

2.1.1 Connectivity in Prime Number Graphs

PG_N^1 graphs may have isolated nodes. The first non-trivial[2] unconnected PG_N^1 graph is PG_7^1 which contains an isolated vertex representing prime 127.[3]

N	Smallest Isolated Prime
2	3 *(trivial)*
3	-
4	-
5	-
6	-
7	127
8	127
9	173
10	173
11	173
12	251
13	251
14	373
15	373

Table 2: Smallest Isolated Prime in PG_N^1 Graphs

An unconnected graph is a graph with more than one connected component. An unconnected graph may have components numbering from two to the number of vertices in the graph, each component containing one vertex to the number of vertices in the graph minus one or $|V| - 1$.

[2] PG_2^k graphs contain only one vertex and are in a sense trivially unconnected.
[3] Prime curio: 127 is also a Mersenne prime of the form $2^N - 1$

As an example, the figure 8 [24] shows the average size of the connected components of PG_N^1 graphs as N increases. It appears that the average size of an component in a PG_N^1 graph may be converging to a number near 6.6 as N gets large. While this plot is introduced here an empirical observation about PG_N^1 graphs there are no hypotheses about why it has the shape that it does. This is an area for future research.

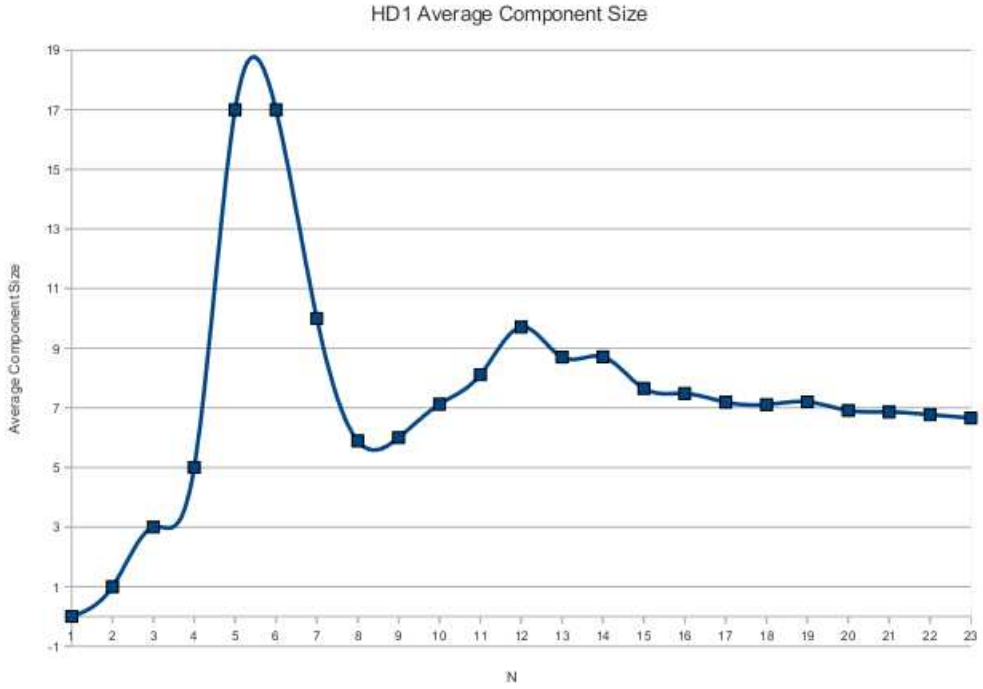

Figure 8: PG_N^1 Average Component Size

Figure 9 [24] shows statistics for PG_N^1 graphs. The column 'Bits' represents the number of bits in the binary representation N. Also shown are the number of vertices in each graph size, the number of edges, the number of graph components, the maximum and minimum component size and the average component size plotted above. Through $N = 6$ the number of graph components remains constant at one, that in this range the graphs are connected. After $N = 6$ the number of graph components increases for every increase in N. An implication from this

trend is that we don't expect the PG_N^1 graphs to begin to become connected at $N > 23$. The only way to show this trend continues for all N is to find an analytic proof, which is a area for future research. Another implication is that if we wish to search this graph for a prime factor, beginning from one vertex represented by a prime number and following a path through an PG_N^1 graph, we would only be able to reach vertices (prime numbers) in the component where we started. What we would like is to have a connected graph of prime numbers, so we could search from a vertex to any other vertex in a connected prime number graph, something that isn't possible for PG_N^1 graphs.

HD1 graph: primes connected components using Tarjan's algorithm							
Bits	Vertices in graph	Edges in graph	Number of elements	Number of graph components	max component size	min component size	average component size
1	0	0	0	0	-1	-1	0
2	1	0	1	1	1	1	1
3	3	4	3	1	3	3	3
4	5	8	5	1	5	5	5
5	10	22	10	1	10	10	17
6	17	38	17	1	17	17	17
7	30	66	30	3	27	1	10
8	53	108	53	9	44	1	5.89
9	96	190	96	16	80	1	6
10	171	342	171	24	144	1	7.12
11	308	638	308	38	237	1	8.11
12	563	1180	563	58	459	1	9.71
13	1027	2086	1027	118	768	1	8.7
14	1899	3862	1899	218	1484	1	8.71
15	3511	7046	3511	459	2551	1	7.65
16	6541	12988	6541	875	4823	1	7.48
17	12250	24252	12250	1704	8594	1	7.19
18	22999	45442	22999	3236	16419	1	7.11
19	43389	85846	43389	6029	29495	1	7.2
20	82024	161504	82024	11869	56879	1	6.91
21	155610	306128	155610	22700	102391	1	6.86
22	295496	582138	295496	43695	199070	1	6.77
23	564162	1108014	564162	84686	359853	1	6.66

Figure 9: PG_N^1 Statistics

PG_N^2 graphs are more richly connected. We investigated by exhaustive enumera-

tion graphs up to size PG^2_{27} and found no graph to have isolated vertices. Figure 10 [24] shows comparison of the relative connectedness of PG^1_N and PG^2_N graphs. For example PG^2_{20} has an order of magnitude more edges than PG^1_{20}. Is it possible that the PG^2_N graphs are connected graphs? The idea that PG^2_N graph are connected is one we will address after we introduce spanning trees.

PG^1_N

hd1		
N	Total Vertices	Total Edges
4	5	4
5	10	11
6	17	19
7	30	33
8	53	54
9	96	95
10	171	171
11	308	319
12	563	590
13	1027	1034
14	1899	1931
15	3511	3523
16	6541	6494
17	12250	12126
18	22999	22721
19	43389	42923
20	82024	80752
21	155610	153064

PG^2_N

hd2*		
N	Total Vertices	Total Edges
4	5	8
5	10	29
6	17	64
7	30	147
8	53	329
9	96	676
10	171	1348
11	308	2666
12	563	5333
13	1027	10635
14	1899	21262
15	3511	42086
16	6541	84014
17	12250	168160
18	22999	334231
19	43389	666981
20	82024	1330754
21	155610	2652761

Figure 10: PG^1_N and PG^2_N Edgecount Statistics

2.2 Spanning Trees in Prime Number Graphs.

A tree is a special graph in which every two vertices are connected by exactly one simple path between them.[11] A path is simple when there are no repeated vertices on the path.

A spanning tree $t(G)$ of a graph G is a tree that contains all the vertices in G and therefore has $|E| = |V| - 1$. In other words the edges of a spanning tree are a subset of the edges of the graph that connect all the vertices. There can be many spanning trees in a graph.

Connected graphs always have at least one spanning tree. Breadth first search (BFS) from any vertex in a spanning tree in a connected graph is complete, in that the search will visit every vertex in the graph. If the search visits every vertex in a PG_N^2 graph containing all the factors of a semiprime S, then a factor of S will be found. This algorithm for searching a prime number graph is guaranteed to visit every prime in the graph.

The growth in the number of spanning trees in PG_N^2 graphs, by Kirchhoff's Matrix-Tree Theorem [11] are shown in Table 3. [4]

In Figure 11, the graph for PG_{10}^2 is shown as an adjacency matrix. The prime numbers are indexed from 1 to 171, as there are 171 prime numbers in $(2, 2^{10})$. Each '$\triangle$' represents a connection between two prime numbers. Each '$*$' is a edge of the spanning tree starting at $P_i = 1$. In Figure 12, the spanning tree begins at $P_i = 171$. These examples show that starting from a different root vertex, it is possible to find a different spanning tree.

[4] The PG_N^2 graphs are not complete graphs so we may not use Cayley's formula[3]: $|t(G)| = |V|^{|V|-2}$

Table 3: PG_N^2 Spanning Tree Counts

N	Number of Spanning Trees	Vertices in PG_N^2
3	3	3
4	45	5
5	132775	10
6	6.46E12	17
7	6.02E26	30
8	6.85E53	53
9	4.58E104	96
10	4.58E196	171

Table 4 shows the path length through the graph starting from either P_1 or P_{171}. If we were to search this graph for a factor of a semiprime, where would we want to begin a search for a factor in this case? Starting a search from P_1 results in a shorter path length which is less work. We will address searching a graph for a factor further in Section 3.

Table 4: PG_N^2 Path Lengths to Factor 523 in Graph For S=523*541=282,943

Root	P_i	Factor	HD	Path Length
3	1	523	2	1
1021	171	523	7	4

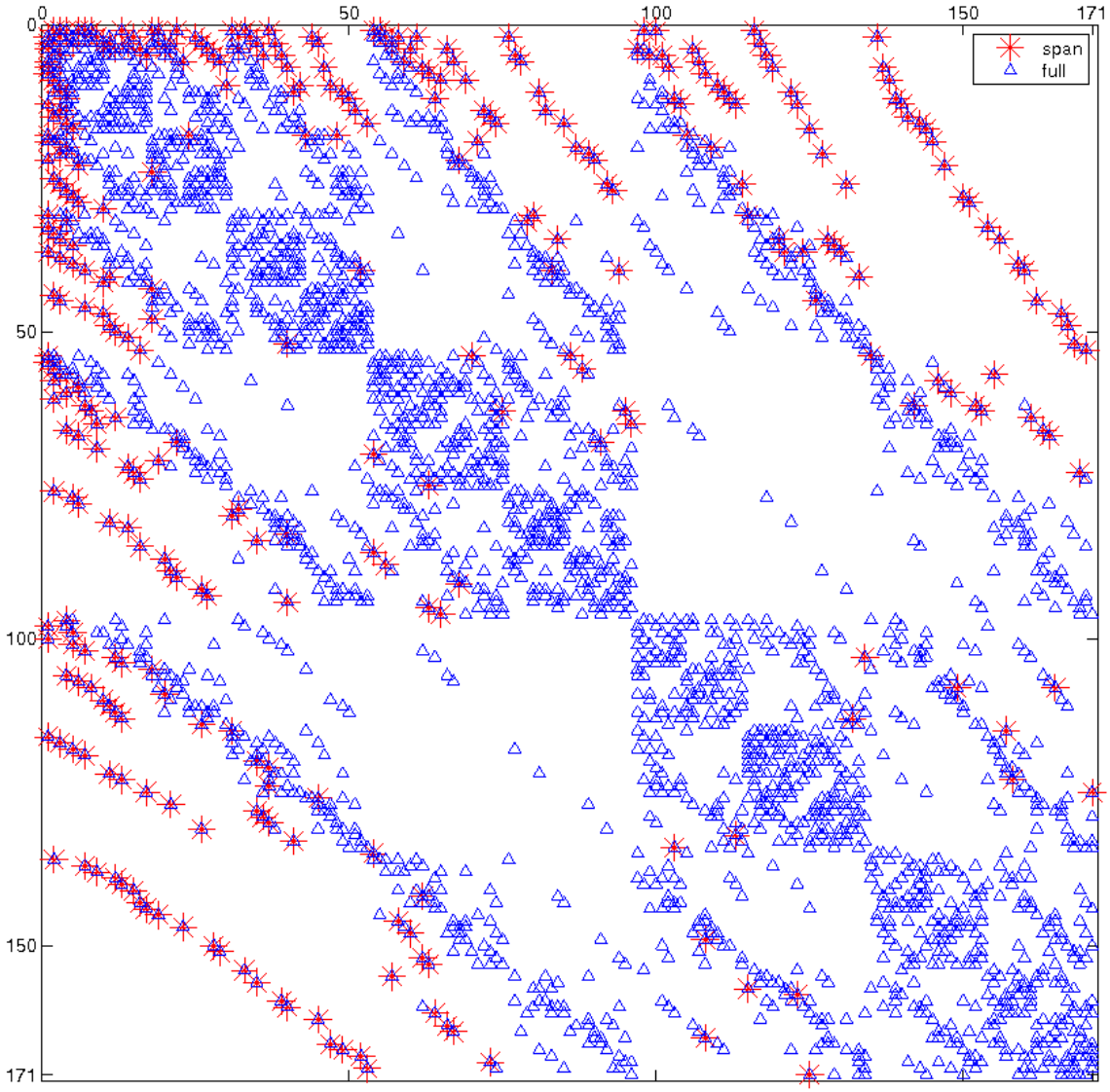

Figure 11: PG_{10}^2 Spantree Starting at 3

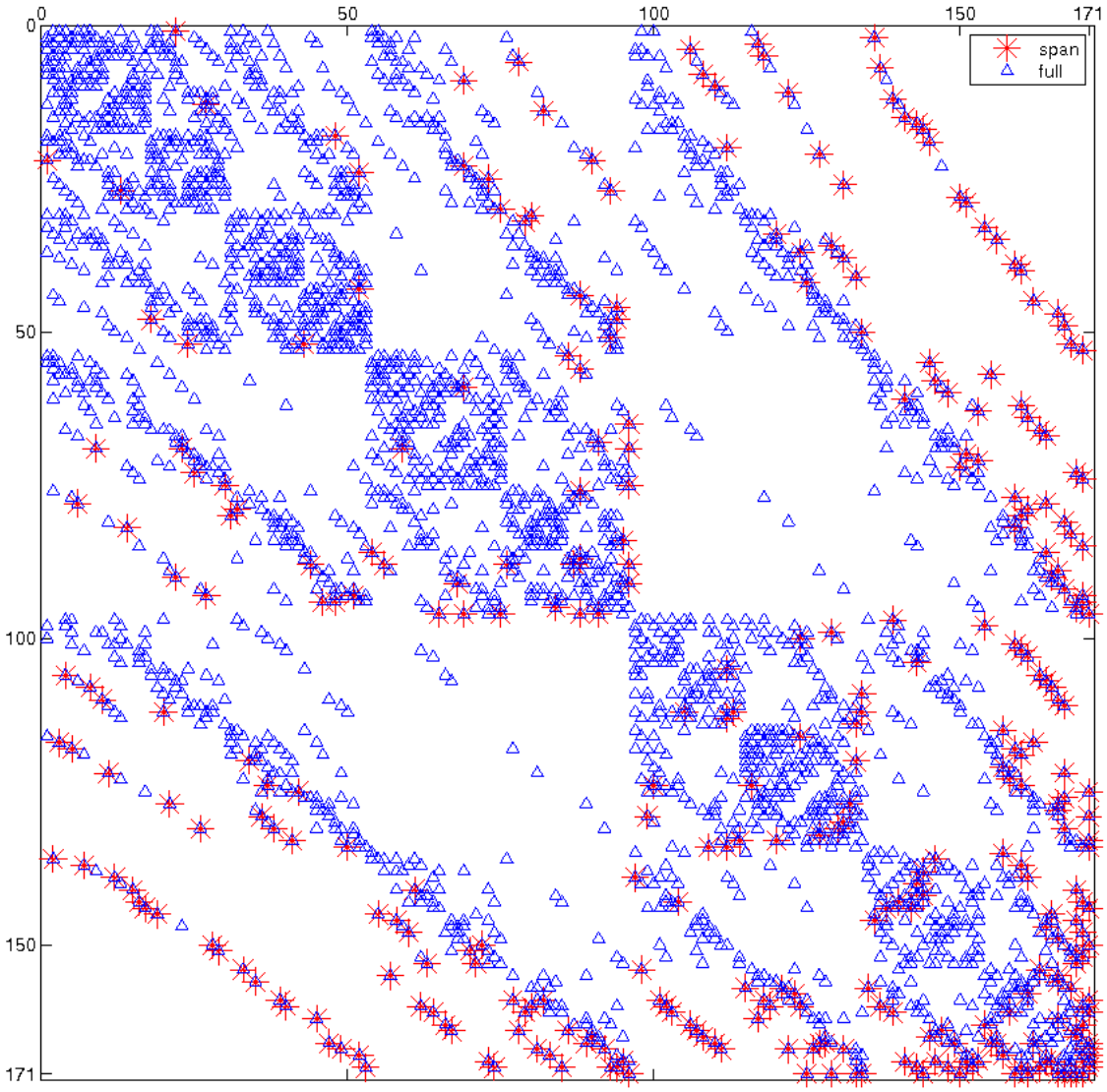

Figure 12: PG_{10}^2 Spantree Starting at 171

2.3 York's Conjecture

York's Conjecture: PG_N^2 graphs are connected.

Dr. Bryant York has conjectured[24] that PG_N^2 graphs are connected (See Section 1.2) for all $N \geq 2$. One implication of this is that a complete spanning tree search of a PG_N^2 graph is guaranteed to find a factor of a semiprime of size $2N$ bits.

More formally this conjecture is that the set of prime numbers $\mathcal{P}$ on the open interval $(2, 2^N), N \geq 2$ form a connected graph PG_N^2 where adjacency is determined by the relation $Hamming\ Distance \leq 2$ when the prime numbers are represented in base 2.

2.4 Counting Prime Numbers: PNT-The Prime Number Theorem

The exact count of prime numbers less than some integer **x** is called the prime counting function $\pi(x)$. The **Prime Number Theorem** [12] shows that the complexity of $\pi(x)$ is $O(\ln(x))$. [7]

$$\pi(x) \sim \frac{x}{\ln(x)} \tag{1}$$

In the limit this is the exact count of prime numbers:

$$\lim_{x\to\infty} \frac{\pi(x)}{\frac{x}{\ln(x)}} = 1 \tag{2}$$

Thus, $\pi(2^N) - 1$ gives us the number of prime numbers in the interval $(2, 2^N)$ which are the number of vertices, $|V|$, in a PG_N^2 graph. We will need to approximate the number of vertices in PG_N^2 in order to develop graph search algorithms for very large N. Using the Prime Number Theorem, the approximate number of vertices, $|V|$, in a PG_N^2 graph is:

$$|V| \sim \frac{2^N}{\ln(2^N)} \sim \frac{2^N}{N} \tag{3}$$

Equation 3 shows the growth in the size of PG_N^2 graphs is $O(2^N)$.

2.5 On the Distribution of Prime Numbers based on Hamming distance

An integer is binomially distributed from the other integers in a range $[0, 2^N)$ by Hamming distance.

Let $N \in \mathbb{Z}^+$ represent the string length of the binary representation of an integer.

The count of integers from $HD = 0$ to $HD = N$ from a given integer on the interval $[0, 2^N)$ follows the following binomial sequence:

$$\left\{ \binom{N}{0}, \binom{N}{1}, \binom{N}{2}, \ldots, \binom{N}{N-1}, \binom{N}{N} \right\}$$

This median of the distribution is near $\lceil \frac{N}{2} \rceil$, since the largest binomial term in the sequence is $\binom{N}{\lceil \frac{N}{2} \rceil}$.

If we define an event, z_k, as the event where we choose an integer $HD = k$ away from some reference integer (e.g. the N-bit integer represented by all $1s$ in base 2), the probability of z_k such that $\sum_0^N z_i = 1$, is:

$$z_k = \frac{\binom{N}{k}}{2^N}, k \in [0, N] \tag{4}$$

As an example, consider the case of ZG_N^2 graphs. These comprise all the integers on the range $(2, 2^N)$. The probability of picking an integer in ZG_N^2 , $HD \leq 4$ from some reference integer (e.g. the N-bit integer which is all $1s$ described above) is:

$$z_4 \sim \frac{\binom{N}{4}}{2^N}, N \geq 4$$

This is the approximate probability because ZG_N^2 graphs do not include the integers 0,1 or 2, but the error of this approximation decreases as N gets large.

Conjecture: The prime numbers, $\mathbb{P}+ \subset \mathbb{Z}+$, are a uniformly scaled distribution of the integer binomial distribution by Hamming distance on the interval $(2, 2^N)$, from a given prime number. There are $\pi(2^N)$ prime numbers numbers in the same range.

This conjecture implies a scaled version of Equation 4.

If we define an event, p_k, as the event where we choose an prime $HD = k$ away from some reference prime (e.g. the prime number 3 in N-bits), the probability of p_k such that $\sum_0^N p_i = 1$ is:

$$p_k \sim \frac{\binom{N}{k}}{\pi(2^N)}, k \in [0, N] \tag{5}$$

As an example, consider the case of PG_N^2 graphs. These comprise all the prime numbers on the range $(2, 2^N)$. The probability of picking a prime in PG_N^2 , $HD \leq 4$ from some reference prime (e.g. the prime number 3 in N-bits) is:

$$p_4 \sim \frac{\binom{N}{4}}{\pi(2^N)}, N \geq 4$$

This median of the distribution is near $\lceil \frac{N}{2} \rceil$, the same as the median for the integer binomial distribution. This follows from our conjecture that the prime numbers are a uniformly scaled version of the integer binomial distribution.

Figure 13 shows how prime numbers are distributed based on Hamming distance from one prime, P. For a given prime P, there is one prime $\mathrm{HD} = 0$ from P, the prime itself. At $\mathrm{HD} = N$, with N the number of bits in the binary representation, there is no prime, which is the one's complement of the number P, an even number. The largest concentration of prime numbers lies in the blue shaded area centered around $\lceil \frac{N}{2} \rceil$.

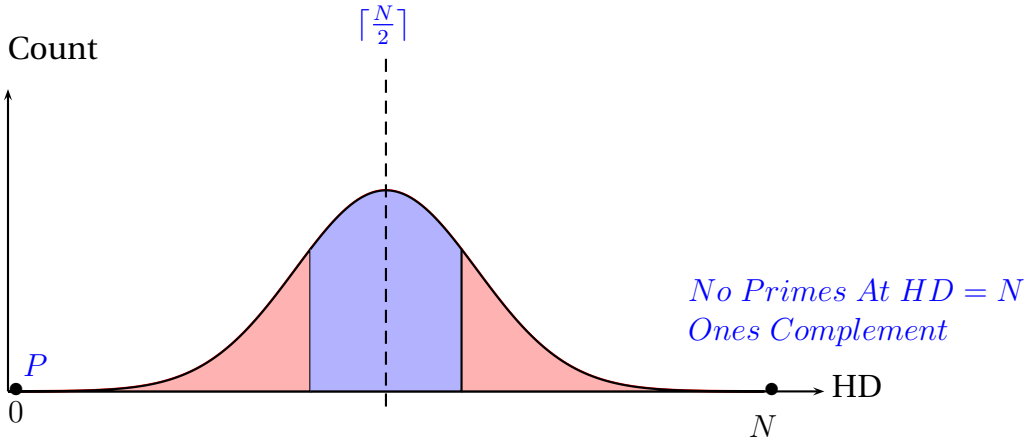

Figure 13: Hamming Distance vs. Number of Primes

If $S = p_1 p_2$ is an N-bit semiprime, $\lceil \sqrt{S} \rceil$ represents the approximate center of the conjectured binomial distribution of prime numbers in the interval $(2, 2^N)$. One factor will be less than $\lceil \sqrt{S} \rceil$ and one will be greater.[5] Since we are choosing prime numbers from this distribution at random Hamming distances, we can use a normal approximation to this conjectured binomial distribution. If we use a normal approximation to this binomial distribution, approximately 68% of prime numbers are within one standard deviation (σ) of the center of the distribution.

In Figure 14 we show the surface created from the actual HD histograms of each of the 6541 prime numbers on the interval $(2, 2^{16})$ from every other prime number in $(2, 2^{16})$. The surface appears visually to be approximating a normal distribution which empirically supports the conjecture that the prime numbers are a scaled binomial distribution by Hamming distance on the integers.

[5] The degenerate cases of semiprimes that are perfect squares are excepted.

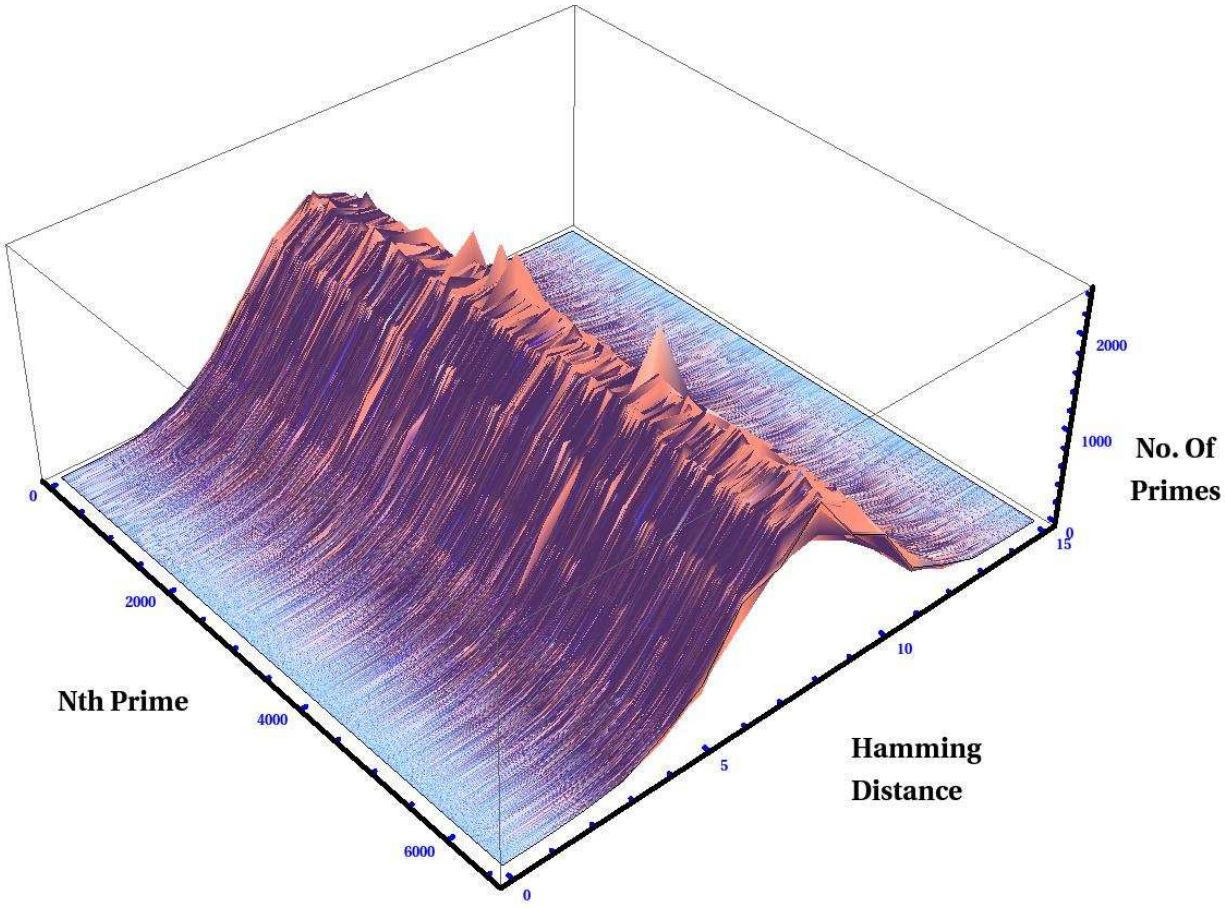

Figure 14: Histograms of Prime Numbers in $\left\{(2, 2^N) \mid N\epsilon\ [2, 16]\right\}$

2.6 Upper Bound for PG_N^2 Diameter

When searching a graph, it is useful to know the diameter of the graph in order to know the maximum number of plies to expand. Expanding plies beyond the diameter of the graph requires redundant work.

Every two prime numbers connected to a third prime by $\mathrm{HD} = 1$ are connected by $\mathrm{HD} = 2$. This connectivity rule is not true in the general case.

As can be seen in Figure 15, every pair of $\mathrm{HD} = 1$ adjacent vertices is equivalent to one $\mathrm{HD} = 2$ hop.

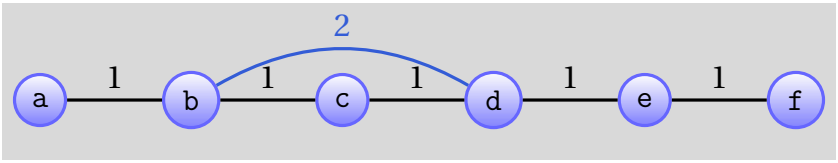

Figure 15: Hamming distance one connections.

Not all prime number graphs are HD1 connected as shown in section 1.2.2 and in Table 2. For example, PG_8^1 is not HD1 connected because prime 127 is not connected to any prime in PG_8^1 by HD1.

Assuming PG_N^2 graphs are connected, our argument for the upper bound on diameter is as follows. Some vertices in the graph are not connected by $\mathrm{HD} = 1$. The worst case odd length path appears in Figure 16. The worst case even length path appears in Figure 17. Note in the example *b* and *c* are connected by $\mathrm{HD} = 1$, and *a and b* are connected by $\mathrm{HD} = 3$, which is outside the adjacency condition. In this case, the only path available is the $\mathrm{HD} = 2$ path on *a* and *c*. A longest shortest path—the graph diameter—would alternate between $\mathrm{HD} = 2$ hops and $\mathrm{HD} = 1$ hops.

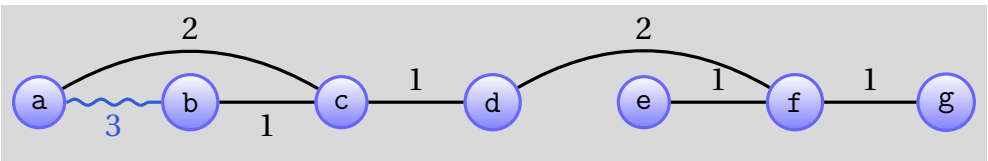

Figure 16: Odd worst case longest path section (N=7).

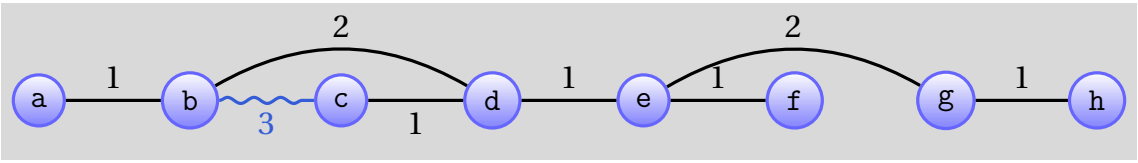

Figure 17: Even worst case longest path section (N=8).

Since all prime numbers in the graph are odd numbers (the prime '2' is not a member of the PG_N^k interval) the least significant bit must be '1' and inverting this bit would create an even number which is not in any PG_N^2 graph. This implies that in PG_N^2 graphs the maximum Hamming distance between any two vertices (prime numbers) is $N-1$.

An upper bound on the graph diameter would be a path with the maximum possible number of $\mathrm{HD}=2$ hops (edges) in $N-1$ plus the number of $\mathrm{HD}=1$ hops. This leads to the following formula for the PG_N^2 graph diameter upper bound D_{ub} :

$$D_{ub} = \lceil \frac{N}{2} \rceil + ((N+1)\ mod\ 2) \qquad (6)$$

Some typical values are:

N	D_{ub}	Actual D(PG_N^2)
5	3	2
6	4	3
7	4	3
8	5	3
9	5	4
10	6	5
11	6	6
12	7	7
13	7	7
14	8	7
15	8	7

Table 5: D_{ub} versus Actual PG_N^2 Graph Diameter

2.7 Implications

In multiagent search we start multiple agents in different neighborhoods of a PG_N^2 graph by starting each agent at the root of separate spanning trees. Each spanning tree root represents the $\mathrm{HD} = 0$ point of a distribution from the spanning tree root to one of two factors. As each neighborhood search explores plies from the spantree root, it increases the probability of finding prime numbers (See Figure 18). Each agent only searches this distribution to a limited depth before beginning a new search in a different neighborhood of the graph from a different spantree root.

This neighborhood search method may represent a viable way to scale prime factor search to large PG_N^2 graphs without incurring the resource costs of complete search. The reason for this is the size of the PG_N^2 graphs grow exponentially in N and searching the entire distribution would be equivalent to visiting every vertex in the graph. Because different spantree root vertices are at different path lengths in the graph from a factor, starting a search in a new graph neighborhood after an unsuccessful limited depth search in the current graph neighborhood may reveal a path to a factor within the search depth from the new spantree root. This process can be repeated over many agents and many starting vertices, up to the number of vertices in a PG_N^2 graph. We have already shown the number of spanning trees in a PG_N^2 graph is equal to or larger than the number of vertices in a PG_N^2 graph (See Table 3).

If the PG_N^2 graphs are connected, then theoretically complete BFS search will succeed in factoring a semiprime given enough resource (just as in sieving). The issue is how much resource (space and time) will be required. Prime graph multiagent limited depth search offers the possibility of substituting neighborhood

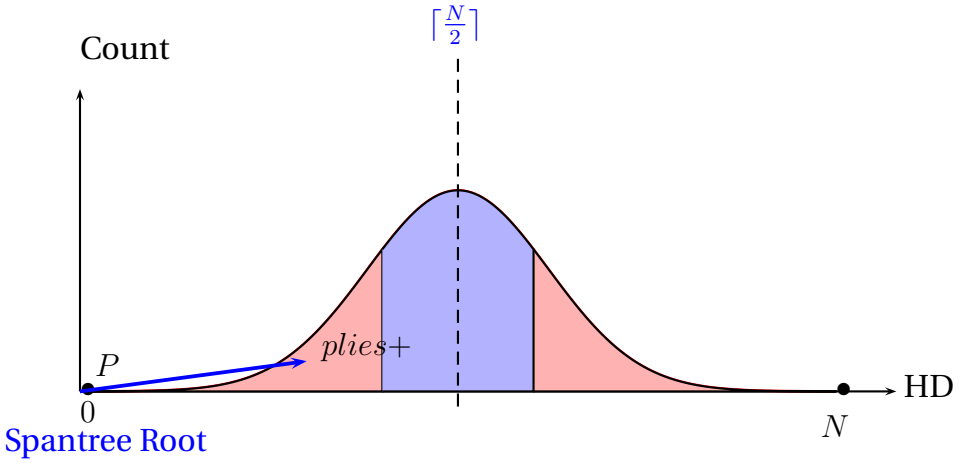

Figure 18: Exploring PG_N^2 Graph Plies From Spantree Root At HD=0

search (or many parallel neighborhood searches) for global search. The critical issue is to determine if there are heuristic methods for identifying seed locations (spanning tree roots) for neighborhood search that increase the probability of success based on features of the prime number graphs.

In the next section we will discuss the details of our search methodology.

3 Search Methodology

3.1 The Basic Search Paradigm

For our prime number graph search we follow a basic generate-and-test paradigm. Starting from the binary representation of a given prime we generate all of the integers which are adjacent to the given prime by $HD2^*$. The test component requires that we apply a primality test to each of these candidates. In our case we use the Rabin-Miller[5][6] algorithm. Rabin-Miller is known to produce false positives with some probability. However, our search is robust enough to tolerate this type of redundancy. The search algorithm builds a BFS tree from the candidates. Non-prime numbers are eliminated from the candidate list by the Rabin-Miller test. The time complexity of Rabin-Miller is $O(k \log^3 n)$ where k is the number of trials and n is the size of the integer under test. The accuracy of the test increases with the number of trials k. The deterministic primality test referred to as AKS [1], has complexity of AKS is approximately $O(\log^6(n))$(See Lenstra [15]). This is slower than Rabin-Miller. Further information on the Rabin-Miller algorithm is in Appendix B.

In Figure 19 we give the sequential version of the basic algorithm. This is a basic BFS search modified for prime number graph search where the initial search begins near the $\lceil\sqrt{S}\rceil$. This approach does not scale to large N so we have decided to pursue parallel search techniques in the development of our algorithm.

INPUT: S a semiprime:
$primeFound \leftarrow FALSE$
$frontierQueue \leftarrow EMPTY$
$S_{csq} = \lceil \sqrt{S} \rceil$
if $isProbablePrime(S_{csq})$ **then**
  if S *modulo* $S_{csq} = 0$ **then**
    $factora \leftarrow S_{csq}$
    $factorb \leftarrow \frac{S}{factora}$
    $primeFound \leftarrow TRUE$
  end if
end if
if $primeFound = FALSE$ **then**
  $N = log_2(S_{csq})$
  $seedList = \{P_i \mid log_2(P_i) \leq N,\ P_i\ is\ ProbablePrime, \mathrm{HD}(S_{csq}, P_i) \leq 2\}$

  $push(frontierQueue, seedList)$
  repeat
    $nut = dequeue(frontierQueue)$
    if $NOT\ visited(nut)$ **then**
      if S *modulo* $nut = 0$ **then**
        $factora \leftarrow nut$
        $factorb \leftarrow \frac{S}{factora}$
        $primeFound \leftarrow TRUE$
      else
        $nextGen = \{P_i \mid log_2(P_i) \leq N,\ P_i\ is\ Prime, \mathrm{HD}(nut, P_i) \leq 2\}$
        $push(frontierQueue, nextGen)$
      end if
    end if
  until $primeFound\ OR\ isEmpty(frontierQueue)$
end if
$Exit$

Figure 19: Basic Algorithm for Prime Factoring

3.2 Complete Search

Complete search of a connected graph is a search that visits every vertex in the graph. Complete searches of PG_N^2 graphs will find factors of a semiprime, assuming the graphs are connected. As has been shown however (See Equation 3, the size of the PG_N^2 graphs grows exponentially with N. The goal of this thesis is to present an algorithm that may not depend on visiting every vertex in a graph or even a majority of the vertices in a PG_N^2 graph to reveal a factor of a semiprime.

Because the complete sequential search approach does not scale to large N, we chose to investigate a parallel heuristic search approach, *Multiagent Depth Limited Search.*

3.3 Multiagent Depth Limited Search

Multiagent search begins multiple searches in parallel at different start vertices, each searching to a limited depth, choosing a new start or seed vertex and searching again, until a prime factor is found or some search limit has been reached. Members of each new ply of the graph search are distributed throughout the decimal space $(2, 2^N)$. Each agent would have a maximum search depth less than or equal to the diameter of the graph (See Section 2.6). A BFS depth limited search with search depth equal to the diameter of the graph and perfect cycle checking would be equivalent to a complete search.

Using a multiagent BFS depth limited search (parallel neighborhood searches) may alleviate the memory costs associated with large frontier queues and visited lists. By searching to a limited depth in one neighborhood of the graph, then starting a new search in another neighborhood of the graph, the size of the fron-

tier queues are limited proportional to $|V|$ in the graph neighborhood.[6] The minimization of resource usage using multiagent search may allow the algorithm to scale to a large numbers of agents in order to explore prime number graphs on the order of PG^2_{64} or larger.

The implication of multiagent search is that each agent searches a tail of the distribution of prime numbers by Hamming distance in the PG^2_N graph. Each new seed is the root of a specific spanning tree and will search only some depth across the distribution from $HD = 0$ (the root) to $HD = search\ depth$. The larger the search depth, the more prime numbers will be searched in a spanning tree associated with the root vertex and the greater the resources will be needed in terms of time and memory (space).

In addition, an implementation may choose to have the different concurrent agents share information about vertices that have already been searched. This would reduce redundant search efforts as the search neighborhood of each agent may overlap.

Choosing the start points or *seeds* of search wisely may increase our chances of beginning a search at the root of a spanning tree close to a factor of a semiprime or to find adjacent prime numbers more quickly. In this thesis we will explore the idea of choosing seeds based on the connectivity of a vertex in a graph as an adjunctive part of our multiagent prime number graph search algorithm.

The next section discusses the analysis of graph vertex degrees for PG^2_N graphs and presents the connectivity features which inspired our connectivity based heuristics.

[6] The number of vertices searched $|V|$ is equal to b^d where d is the depth and b is the branching factor.

3.4 Feature Analysis for Development of Heuristics

3.4.1 PG_N^2 Vertex Degree Frequency Analysis

Each vertex in a PG_N^2 graph has an associated vertex degree.[7]

As an experiment, the degree of each vertex of PG_N^2 graphs from $N = 3$ to $N = 34$ was counted. This is possible because we are able to enumerate every prime in each of these graphs. The results are in Table 6. This table shows that the average degree of a vertex is increasing over N. We will use tables like this, on prime number graph sizes we can realize, to build a basis for extrapolation to prime number graph sizes where we cannot physically enumerate every vertex.

Table 6: PG_N^2 **Graph Node Degree Analysis**

N	mean	median	std.	min	max	(N+1)/(2ln2)
3	2.0	2.0	0.0	2	2	2.89
4	3.2	3.0	0.4	3	4	3.61
5	5.8	6.0	0.75	5	7	4.33
6	7.53	7.0	1.19	5	9	5.05
7	9.8	10.0	2.14	5	14	5.77
8	12.42	12.0	3.01	5	18	6.49
9	14.08	14.0	3.12	6	21	7.21
10	15.77	16.0	3.54	6	23	7.93
11	17.45	17.0	4.21	6	28	8.66
12	18.94	19.0	4.74	3	33	9.38
13	20.71	21.0	4.98	4	38	10.1
14	22.39	22.0	5.21	6	40	10.82
15	23.97	24.0	5.58	6	44	11.54
16	25.69	26.0	5.96	7	48	12.26
17	27.45	27.0	6.16	6	56	12.98
18	29.06	29.0	6.41	5	58	13.71

Continued on next page...

[7] See section 1.2.

Table 6 – continued						
N	mean	median	std.	minbin	maxbin	(N+1)/(2ln2)
19	30.74	31.0	6.65	2	63	14.43
20	32.45	32.0	6.94	2	65	15.15
21	34.09	34.0	7.18	4	69	15.87
22	35.77	36.0	7.41	5	75	16.59
23	37.45	37.0	7.64	4	79	17.31
24	39.1	39.0	7.85	6	84	18.03
25	40.76	41.0	8.04	6	90	18.76
26	42.45	42.0	8.25	3	96	19.48
27	44.13	44.0	8.45	3	102	20.2
28	45.81	46.0	8.64	4	103	20.92
29	47.48	47.0	8.83	4	106	21.64
30	49.17	49.0	9.02	3	111	22.36
31	50.83	51.0	9.19	3	118	23.08
32	52.53	52.0	9.38	4	119	23.8
33	54.18	54.0	9.55	4	122	24.53
34	55.82	56.0	9.72	4	126	25.25
						End

A histogram of the vertex degrees for PG_{30}^2 is plotted in Figure 20.

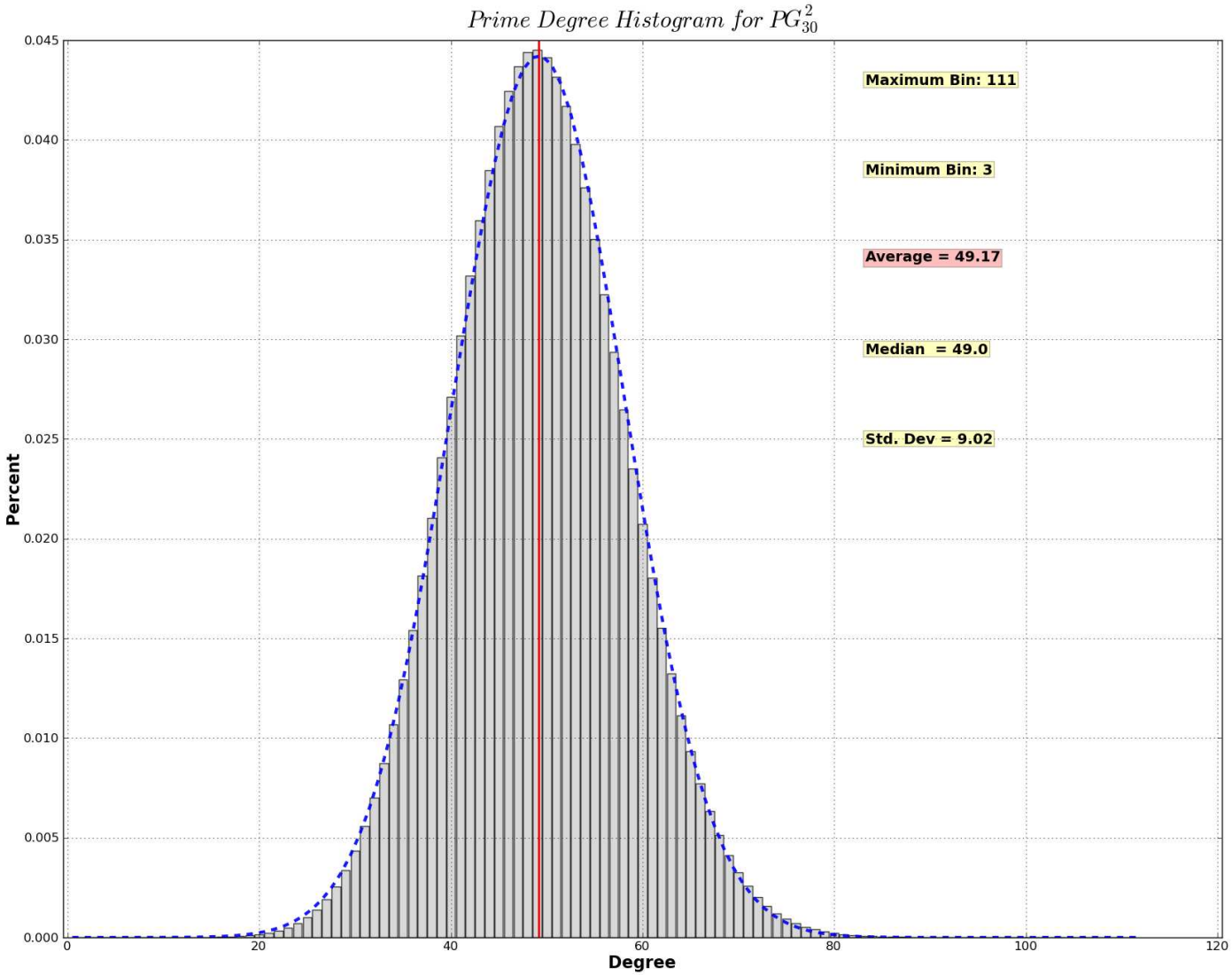

Figure 20: Graph Node Degree Histogram For PG_{30}^2

The average degree of a vertex in a PG_N^2 graph from $N = 3$ to $N = 30$ is plotted in Figure 21. A least squares linear fit to this data returns a function to be used in extrapolation to the average vertex degree for PG_N^2 for larger graphs. This equation in N is:

$$average\ degree = 1.726 * N - 2.209 \qquad (7)$$

We are developing this algorithm as a basis for extrapolation. By removing the data for $N = 31$ to $N = 34$ from the curve fit data we will be able to test our extrapolation methods based on data for $N \leq 30$ and measure our extrapolated performance on PG_N^2 graphs from $N = 31$ to $N = 34$. Without this 'reserve data' we would not be able to test how well our ideas for extrapolation perform. This thesis presents a basis for this future extrapolation work. [8]

The average vertex degree information will allow us to create heuristics for seed or start point selection in multiagent neighborhood search beyond the size of PG_N^2 graphs we can enumerate. The question that we will be asking is whether starting a neighborhood search with vertices that have higher than average connections will lead to a shorter path than those with less than average connections.

[8] This is similar to the technique in adaptive systems of having both a training set and a test set before extrapolating to intractably large data sets.

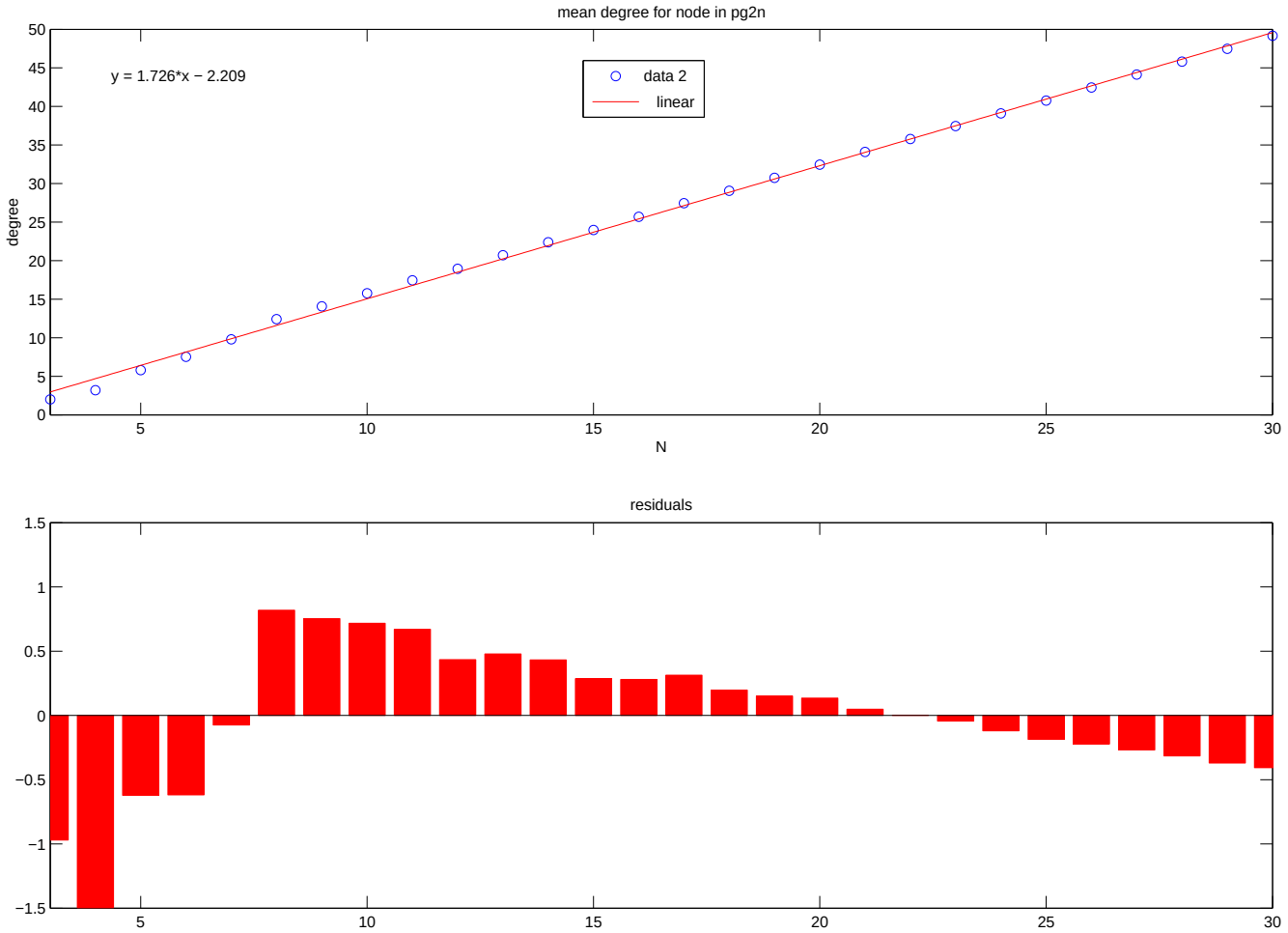

Figure 21: Plot and Fit of Mean Graph Node Degree For PG^2_{30}

3.4.2 Connectivity Analysis

In Table 7 are shown samples of the vertex degrees for sample vertices in PG_{30}^2. Which of these prime numbers is a better choice for the start of a neighborhood search? This differential between the most and least connected seeds is the basis of *most connected, least connected and random connected* seed pick heuristics described in the next section. The most connected vertex in PG_{30}^2 has 111 neighbors, while the least connected vertex has only 3 neighbors. The difference in vertex degree between the most connected and least connected vertices is compelling because it may indicate there exist densely connected regions of the graph and sparsely connected regions of the graph.

Table 7: Sample Connected Prime Numbers in PG_{30}^2

Degree	Prime Index	Prime
111	723	5479
109	658	4933
108	8370	86113
⋮	⋮	⋮
5	37027930	715795123
4	19631044	366302549
3	37029520	715827883

How many prime numbers would we expect to find if we chose integers from at random from an interval $(2, 2^N)$?

The number of integers connected to a node in a ZG_N^2 graph is:

$$|W| = N + \binom{N}{2} = \frac{N(N+1)}{2}$$

The probability of randomly selecting a prime number, $pr(\mathcal{P})$ from a set of integers on the interval $[0, 2^N]$ is:

$$pr(\mathcal{P}) \sim \frac{\pi(2^N)}{2^N}$$

If a set of the same size as $|W|$ of integers above were chosen randomly, the expected number of prime numbers in the set would be:

$$pr(\mathcal{P}) \times |W| = \frac{\frac{\pi(2^N)}{\ln 2^N} \times N(N+1)}{2^{N+1}}$$

Note, from the prime number theorem,

$$\pi(x) \sim \frac{x}{\ln(x)}$$

Using this approximation:

$$\pi(2^N) \sim \frac{2^N}{\ln 2^N} = \frac{2^N}{N \ln 2}$$

$$pr(\mathcal{P}) \times |W| \sim \frac{2^N(N+1)}{2^{N+1} \ln 2} = \frac{N+1}{2 \ln 2} \sim \frac{N}{2}$$

Table 6 shows the observation that the average degree of a node in a PG_N^2 graph is significantly larger than this number, the result of choosing integers randomly. This observation implies that PG_N^2 graphs are highly connected and supports the idea of using a search heuristic which is a function of the degree of a node in a PG_N^2 graph.

3.5 Heuristic Multiagent Search for Prime Factors

Search heuristics for multiagent search are concerned with choosing the seeds for each depth limited search. The goal of these heuristics is to minimize the path length to a factor, from the start vertex of a search.

Table 6, lists the results of an analysis of the graph vertex degree of the vertices in PG_N^2 graphs. The higher the degree of a vertex the greater the connectivity of the vertex in the graph. Creating seed selection heuristics based on connectivity is a direct result of this analysis. Since this analysis is limited to PG_N^2 graphs where all the prime numbers are known, techniques for extrapolation of vertex degree are necessary when N is too large to calculate the degree of every vertex in the graph. In the case where the graph is too large these heuristics would have to base their selection criteria on the extrapolated average connectivity and choose seeds based on whether they were larger or smaller then the averages. The linear curve fit from Equation 7 and Figure 21 could be used to accomplish this extrapolation.

Most Connected

In the most connected heuristic, vertices are chosen as seeds beginning with the most connected, or highest vertex degree. The hypothesis here is that the more connected vertices would search more prime numbers in a single ply, and reduce the depth of plies to search for a factor.

Least Connected

In the least connected heuristic, vertices are chosen as seeds beginning with the least connected, or least vertex degree. The hypothesis here is that the least connected prime numbers would search more sparsely connected regions of the graph, increasing the chance of investigating prime numbers not yet seen in the

search.

Random Connected

In the random connected heuristic, prime numbers are chosen as seeds at random from from the interval $(2, 2^N)$. If neither most or least connected heuristics can outperform this method of choosing random vertices, then the heuristics are not helpful.

In the next section we will discuss the analysis we performed in order to develop 'hard' semiprimes, that is semiprimes that are hard to factor, in order to test the performance of our algorithm.

3.6 Preparation of Testcases for Experimentation

In order to test our algorithms we needed to construct semiprimes which are difficult to factor but on a scale where we could get results in a reasonable time. We studied the features of RSA Laboratories challenge semiprimes so that we could construct smaller semiprimes of comparable difficulty.

3.6.1 Analysis of Factored RSA Semiprimes

The RSA Laboratories Challenge numbers[9] were designed to be the hardest type of semiprime to factor. We analyzed the factored semiprimes in order to discover some common features and connectivity characteristics of the factors in order to design difficult test cases for our algorithm. Figure 22 shows some of the features of these numbers.

[9] http://www.rsa.com/rsalabs/node.asp?id=2093

Analysis of Solved RSA Numbers								
			Actual					
						%		%
RSA	Bits	RSA Factor	Nbits	C1	C2	C2/Nbits	HD(x,sqrt)	|x-sqrt|/sqrt
100	330	RSA 100 F1	165	0	111	67	80	2.75
100	330	RSA 100 F2	165	2	127	77	85	2.67
110	364	RSA 110 F1	182	2	66	36	82	2.33
110	364	RSA 110 F2	182	0	60	33	94	2.28
120	397	RSA 120 F1	198	4	60	30	111	31.28
120	397	RSA 120 F2	199	4	154	77	110	45.52
129	426	RSA 129 F1	212	5	72	34	103	67.36
129	426	RSA 129 F2	215	5	147	68	101	206.39
130	430	RSA 130 F1	215	3	143	67	105	6.64
130	430	RSA 130 F2	215	1	170	79	116	7.11
140	463	RSA 140 F1	231	4	166	72	105	26.34
140	463	RSA 140 F2	232	3	85	37	112	35.76
150	496	RSA 150 F1	248	3	88	35	124	11.63
150	496	RSA 150 F2	248	3	83	33	126	13.16
155	512	RSA 155 F1	256	1	78	30	134	1.87
155	512	RSA 155 F2	256	4	107	42	132	1.91
160	530	RSA 160 F1	265	5	193	73	127	2.09
160	530	RSA 160 F2	265	2	172	65	136	2.13
200	663	RSA 200 F1	331	1	244	74	173	33.24
200	663	RSA 200 F2	332	2	126	38	171	49.79
576	576	RSA 576 F1	288	3	115	40	138	8.23
576	576	RSA 576 F1	288	3	93	32	148	8.97
640	640	RSA 640 F1	320	6	130	41	162	7.26
640	640	RSA 640 F2	320	3	97	30	162	7.83

Figure 22: Analysis of Solved RSA Challenge Numbers[20]

In the figure C1 and C2 are the $\mathrm{HD1}$ and $\mathrm{HD2}$ prime coronas of each of the factors of the numbers listed in the RSA column. One feature to note are the small $\mathrm{HD1}$ coronas of these numbers, some of which are isolated $\mathrm{HD1}$ prime numbers. The $\mathrm{HD2}$ coronas are much larger, however they are less than what we would expect from the average degree formula in Equation 7. For example, the extrapolated average degree for RSA640, a 640 bit number, with factors of 320 bits, is 561. Both factors for RSA640 are in PG^2_{320} and have degrees of 136 or fewer.

Also note that the factors are frequently within ten percent of the square root of the semiprime S number. Using factors close in magnitude to the square root of

S is intended to make factoring by sieving difficult, by using factors as large as possible.

Picking large factors, with relatively low connectivity as in the RSA640 example are two aspects of designing difficult to factor semiprimes. We have to design our testcases with these ideas in mind.

3.6.2 Analysis of Smartly Generated Small Challenge Numbers

Table 8 lists the test cases we used in exploring PG_N^2 graph factoring algorithms.

Table 3 shows a sample of small challenge numbers we used. We generated our small challenge numbers using the features from our analysis of the RSA numbers. The resulting analysis appears in Figure 23.

The analysis in Figure 23 is a similar to the analysis done for the RSA numbers. As in the RSA numbers, our numbers are near S_{csq} and have relatively low connectivity in the graph. The data in Figure 23 shows that we approximate the quality of the RSA Laboratories analysis Figure 22, but there are relatively few factored RSA numbers. In this sense, the RSA factor analysis is a guide to how we design our hard to solve semiprimes.

Table 8: **Smartly Generated Small Test Numbers**

N	Semiprime	Factor A	Factor B	Filename
24	9990157	3119	3203	sm1-24
25	32639989	5507	5927	sm2-25
26	55127929	7247	7607	sm3-26
27	119094961	10559	11279	sm4-27
28	220023049	14303	15383	sm5-28
Continued on next page...				

Table 8 – continued				
N	Semiprime	Factor A	Factor B	Filename
29	526612501	22739	23159	sm6-29
30	571077781	23159	24659	sm7-30
31	1298487901	34583	37547	sm8-31
32	3130236121	55103	56807	sm9-32
64	17017141319541425869	4293845903	3963146723	sm10-64

Analysis of Smartly-Generated Small Challenge Numbers								
						%		%
Small Challenge Numbers	Bits	Factors	Nbits	C1	C2	C2/Nbits	HD(x,sqrt)	|x-sqrt|/sqrt
SM1	24	SM1 F1	12	1	20	1.666667	6	1.3
SM1	24	SM1 F2	12	1	21	1.75	6	1.36
SM2	25	SM2 F1	13	1	13	1	6	3.61
SM2	25	SM2 F2	13	3	17	1.307692	6	3.75
SM3	26	SM3 F1	13	2	10	0.769231	6	2.38
SM3	26	SM3 F2	13	4	18	1.384615	6	2.46
SM4	27	SM4 F1	14	3	14	1	7	3.24
SM4	27	SM4 F2	14	1	26	1.857143	7	3.35
SM5	28	SM5 F1	14	0	13	0.928571	7	3.57
SM5	28	SM5 F2	14	3	17	1.214286	7	3.71
SM6	29	SM6 F1	15	2	22	1.466667	7	0.91
SM6	29	SM6 F2	15	2	17	1.133333	7	0.92
SM7	30	SM7 F1	15	2	17	1.133333	7	3.09
SM7	30	SM7 F2	15	2	18	1.2	7	3.19
SM8	31	SM8 F1	16	2	24	1.5	8	4.03
SM8	31	SM8 F2	16	0	24	1.5	8	4.2
SM9	32	SM9 F1	16	0	13	0.8125	8	1.51
SM9	32	SM9 F2	16	2	23	1.4375	8	1.54
SM10	64	SM10 F1	32	2	66	2.0625	17	4.09
SM10	64	SM10 F2	32	0	57	1.78125	18	3.93
SM11	128	SM11 F1	64	2	113	1.765625	27	8.01
SM11	128	SM11 F1	64	1	97	1.515625	27	7.42
SM12	256	SM12 F1	128	1	190	1.484375	70	8.01
SM12	256	SM12 F2	128	2	204	1.59375	64	7.42

Figure 23: Analysis of Smartly Generated Challenge Numbers[20]

3.6.3 Designing 'Hard' Semiprimes for $PG_N^2\,\mathrm{HD2}^*$ Search

As in the generation of the smartly generated semiprimes, similar thought has gone into the generation of the testcases for designing semiprimes for multiagent nearest neighbor search. We call these prime numbers 'hard' semiprimes.

The following describes the processes we used to construct hard semiprimes.

- Choose factors beyond a Hamming distance of one or two from a factor near S_{csq}. Choosing factors within a Hamming distance of one or two from a factor near S_{csq}, will require searching only the first ply of the graph from the S_{csq}.
- A 'hard' semiprime has factors that are not too close in HD to S_{csq} but not so far that the factors are in the 'tails' of the distribution.
- The test cases created for this thesis chose semiprimes with factors from the intervals, where D_h represents Hamming distance:
  $(\frac{D_h}{4}, \frac{D_h}{4} + (0.25 * \frac{D_h}{4}))$
  and
  $(\frac{3D_h}{4} - (0.25 * \frac{3D_h}{4}), \frac{3D_h}{4})$.

Choosing semiprimes with factors as described above creates a reasonable set of testcases for factoring. We have generated at least one hundred testcases for each graph size for $10 < N < 34$.

4 Experimental Results

An exploratory experiment demonstrated the viability of multiagent search using the small set of smart semiprimes in Table 8. Factors up to 32 bits were successfully factored in this early exploration.

4.1 Independent Multiagent Limited Depth BFS

This was a preliminary experiment to test the viability of using multiple agents to search a prime number graph for factors of a semiprime. For each graph size only one semiprime was tested with only one seed per agent, while the depth of search and the number of agents were varied. The question we were trying to answer was whether this type of search would work at all.

The purpose of this experiment is to explore factoring semiprimes of size $N = 24$ to $N = 32$ bits from the set of test semiprimes (See Table 8). Using independent agents, which have no shared data structures, search in the space with each agent starting at a unique, maximally connected seed prime.

The variables for each run are, the test semiprime, the maximum number of plies to expand during the BFS and the number of search agents to create.

For the purposes of BFS search, cycle checking was implemented with a local Bloom filter [2]. A Bloom filter was used to gain experience with the behavior and implementation of this data structure with the intention that it would be useful in scaling to large PG_N^k graphs. Each agent maintained an independent frontier queue.

The largest graph available at the time of this preliminary experiment from which

to choose seeds was PG_{22}^2. Further work has been done since (See 6). This was sufficient to test the idea that is was possible to use multiagent search to find factors of a semiprime.

As can be seen from the spanning tree graph (Figures 11 and 12), choosing a seed creates an entry into a new spanning tree. Choosing the most connected seeds involves the following two steps.

The search begins by creating one seed per agent using depth-limited BFS neighborhood search. Each agent maintains a local frontier queue. Cycle checking is done through the use of a global probabilistic hash table. Before each prime number in the next ply is added to the frontier queue, it is checked against the global hash table. If the prime number has already been visited, then it will not be added to the frontier queue. Either this agent or another has already searched the graph from this vertex.

Our preliminary experiment performed a limited BFS over semiprimes from 24 to 32 bits in length. The number of agents ranged from 1 to 5 and number of search levels ranged from 1 to 5. The seeds were chosen from PG_{22}^2 , which is the largest graph vertex degree dataset available at the time.

The only search heuristic used was the choice of starting seeds. In this experiment, the prime with the highest degree was chosen first from our list of most connected primes in PG_{22}^2 followed by the next highest degree prime, up to the number of agents searching the graph.

Result: Each of the test prime numbers were successfully factored with this simple seed choice heuristic and no pruning of the search space during breadth first search. All of the factors were found at four plies of the initial search seed or fewer.

If every factor were to have been found at the fourth ply, given an vertex degree average of 39 (See Table 6) then the search would have visited approximately 14% of the graph for PG_{24}^2 .

The results shown in Table 9 using five agents support the conjecture that search based factoring algorithms over PG_N^k graphs can be successful for small N. The full table can be found in Appendix C.

Table 9: **Selected Independent Limited BFS Results**

N	Num. Agents	Max. Ply	Found Ply	Seed	HD (sd,$F_{\boldsymbol{s}}$)	A	B	S = A * B
24	5	2	2	59	4	3119	3203	9990157
25	5	5	2	7573	4	5507	5927	32639989
26	5	5	1	7573	2	7607	7247	55127929
27	5	5	2	59	4	10559	11279	119094961
28	5	5	2	7573	4	15383	14303	220023049
29	5	5	3	7573	6	22739	23159	526612501
30	5	5	4	7573	8	23159	24659	571077781
31	5	5	3	7573	6	34583	37547	1298487901
32	5	5	3	7573	6	56807	55103	3130236121
								End

Next we discuss the results of testing the three search heuristics over different graph sizes, using the hard semiprimes we created as testcases. (See Section 3.6.3)

4.2 Results for Connectivity Heuristics

This section summarizes the results of testing the connectivity heuristics we proposed in Section 3.5. In contrast to the preliminary test of the previous section, this test factored more than one semiprime at each graph size and heuristic.

Each of the heuristics were tested at a fixed number of seeds, agents and depth of search (plies) on PG_N^2 graphs from $N = 10$ to $N = 26$. The results of each test are averaged over 40 samples (attempted factorings). Studying the behavior of this algorithm with graphs we can physically realize may help us extrapolate to larger graph sizes, where the graph is too large to enumerate.

Figure 24 shows the relative performance of the three connectivity heuristics in average seconds aggregated for successful or failed factoring. The top portion of the figure shows the percentage of successful factoring attempts ('Gamma', γ). As in Figure 26 the most connected heuristic is showing a trend of outperforming the other connectivity heuristics. This result is aggregated for successful and failed factoring. Here, the most connected heuristic is attractive because if the search is going to fail, the most connected heuristic will fail faster on average.

The search does succeed 100% of the time until we get to a $N > 23$. At this point we start to see a decline in the performance of the multiagent graph search factoring algorithm. We expect the algorithm to begin to fail at some size because we have fixed the number of start seeds, the depth of search and the number of agents. When these limits have been reached, the search stops.

Observation: In Figure 24, in the region of $N > 23$ the heuristic of most connected pick outperforms random pick and least connected pick by 30% in terms of seconds at $N = 26$.

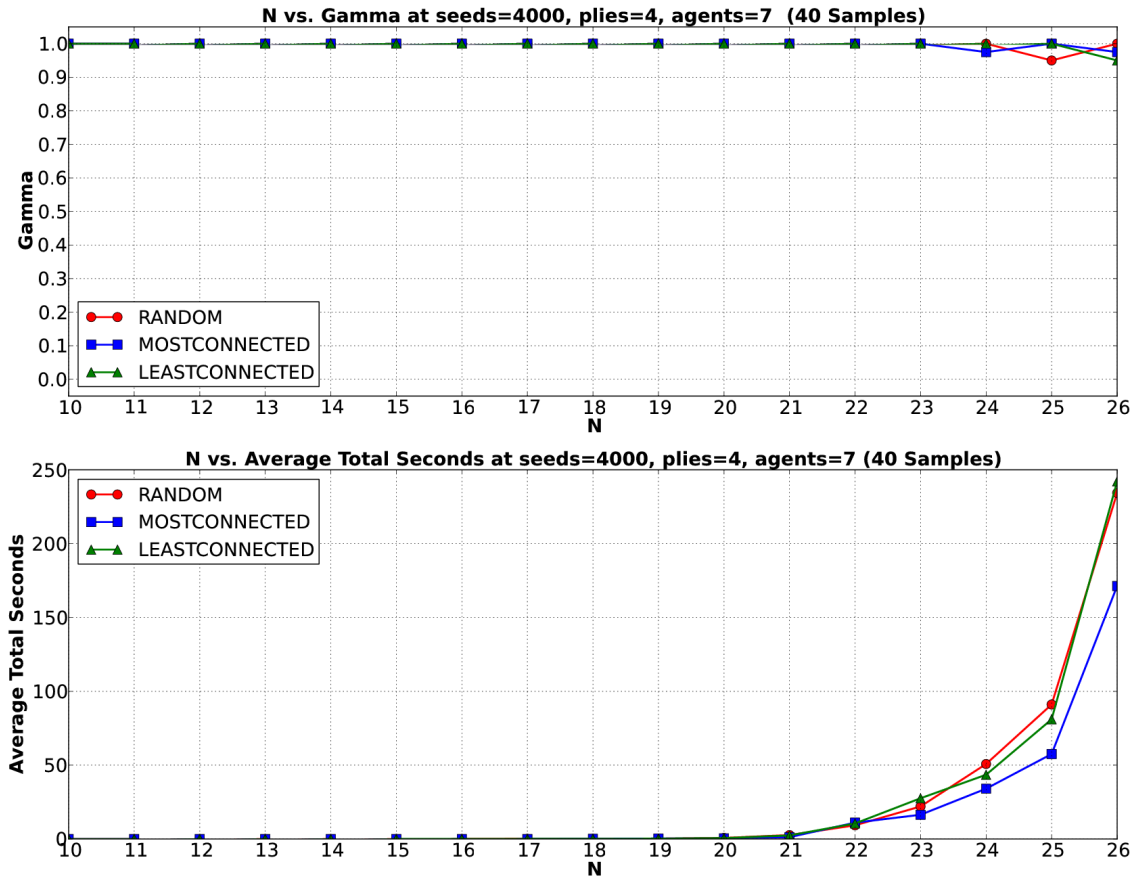

Figure 24: Execution Time

Next, we compare the heuristics based on what ply on average a factor is found during search. One of the goals of these heuristics is to bias the search towards finding a spantree root within a neighborhood equal to the search depth, of a factor. Figure 25 compares the average ply searched and a trend is again seen for most connected pick. It is not a large advantage but it is consistent. The correlation between the two plots of the figure support consistent results overall as the most connected pick heuristic runs faster and searches to a smaller depth overall. This correlation increases confidence that the experiments for comparing heuristics for PG_N^2 graph search were executed in a consistent manner.

Observation: The heuristic of most connected pick shows a slight performance advantage over the other heuristics in terms of average depth of search.

Observation: Visually, there is a correlation between time of search and depth of search in Figure 25. The most connected pick heuristic also outperforms at larger N in Figure 26. This consistent result increases confidence in our test methodology and execution.

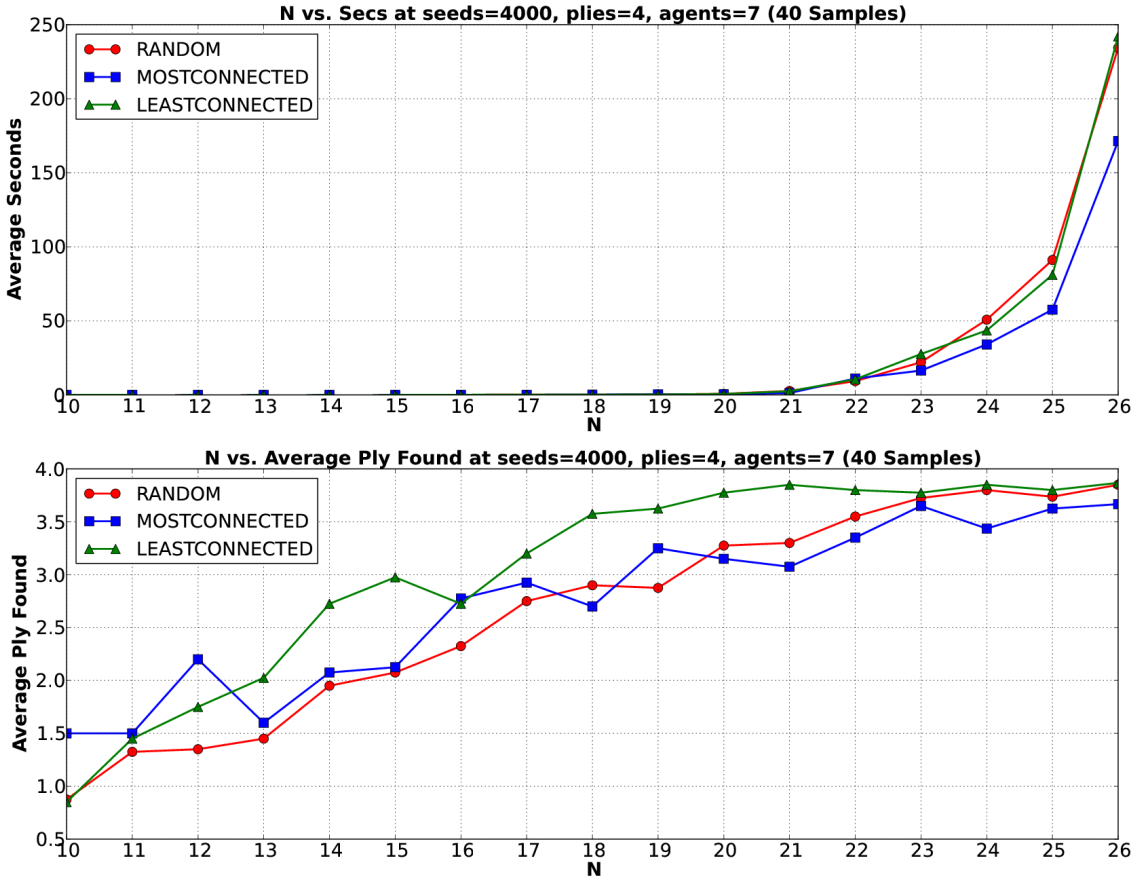

Figure 25: Depth of Search

Next we examine the number of vertices visited as a percentage of the total graph vertices. This is the best performance metric for this algorithm. It is not affected

by other processes running on the machine which degraded the accuracy of our metric of seconds and has a larger range than the depth of search metric. There are two two plots, successful factoring totals and failed factor totals.

Any heuristic that searches a quantity of vertices more than fifty percent of the prime numbers in the graph is less efficient than just searching for a factor by trial division of all prime numbers either less than or greater than the square root of the semiprime.

As can be seen in the Figure 26, which is the percentage of total vertices searched for successful factorings, a performance difference begins to appear between each of the heuristics chosen as N increases. The most connected heuristic is consistently searching a smaller percentage of the graph than either least connected or random connected. This figure, however, is only accounting for successful search.

Observation: In Figure 26, in the region of $N > 23$ the heuristic of most connected pick outperforms random pick and least connected pick while searching between fifteen and twenty percent of the total vertices in a PG_N^2 graph.

Figure 27 is the plot of percentage of total vertices searched for failed searches. At $N > 23$ is where search begins to fail for some samples. At $N = 26$ the percentage of vertices searched for failed factoring attempts reaches as high as 60%. Factoring is still succeeding 95% of the time, so this only represents a few data points.

This observation that failed factoring attempts search many more vertices in the graph is possibly due to the small sample size (forty samples) for the experiment which is a small sample relative to the size of the search space, $\binom{\pi(2^N)}{2}$. It may be

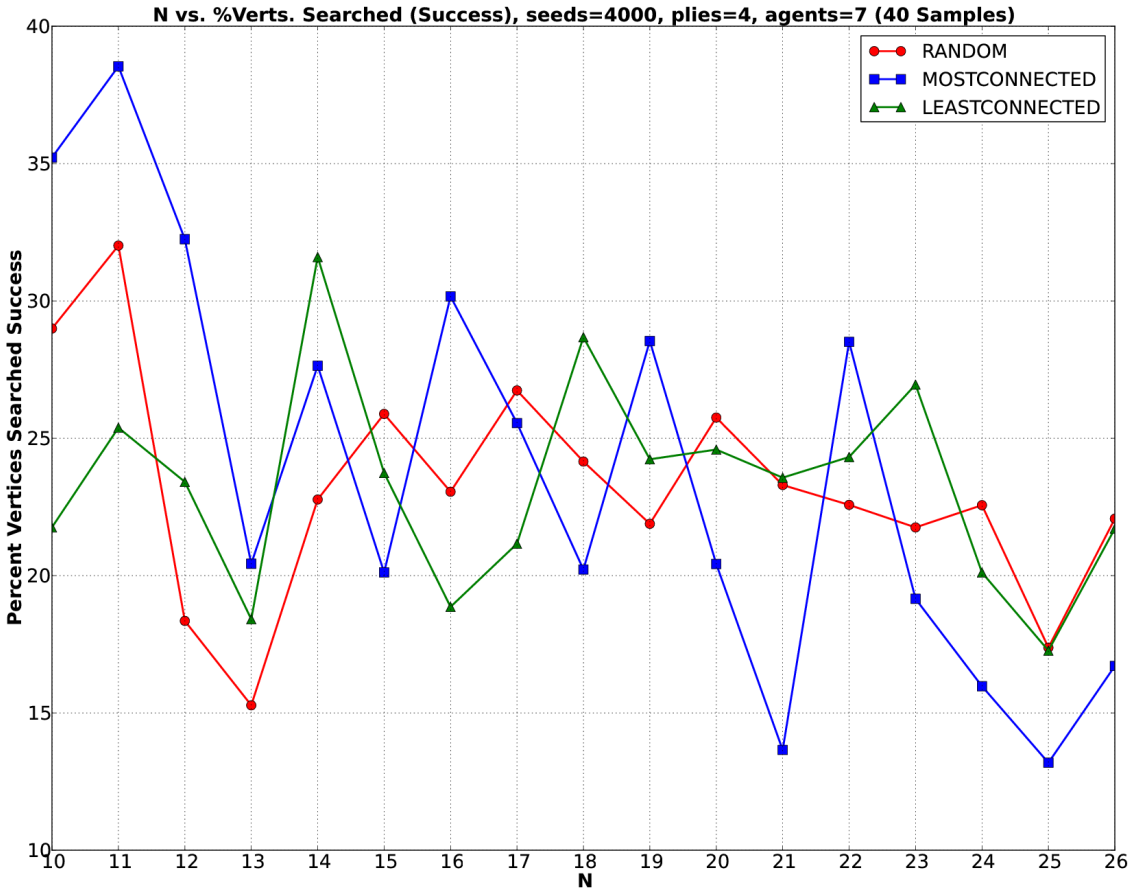

Figure 26: Percent of Vertices Searched for Successful Factoring

that when search fails, a much larger number of vertices are searched than when search succeeds. It may be that these few failures are the result of some especially difficult to factor semiprimes. This will have to be investigated and to do so will require many more samples of failed factorings. If these failures are due to something unique about the sample semiprimes, the investigation of these samples may provide insight useful for extrapolation to larger PG_N^2 graphs. Working with graphs of a size we can realize allows us to perform these investigations where with larger graphs will be difficult or impossible to reconstruct the sequence of

events leading to the result.

Search succeeds approximately better than 95% of the time in this region of the plot, at a fixed number of seeds, agents and depth of search. Figure 26 does show a possible trend for most connected search having the best performance. Figure 27 is a result that will need further research before a clear conclusion can be made.

Observation: When search fails, the number of vertices visited shows a marked relative increase as a percentage of the vertices in the graph, relative to successful searches.

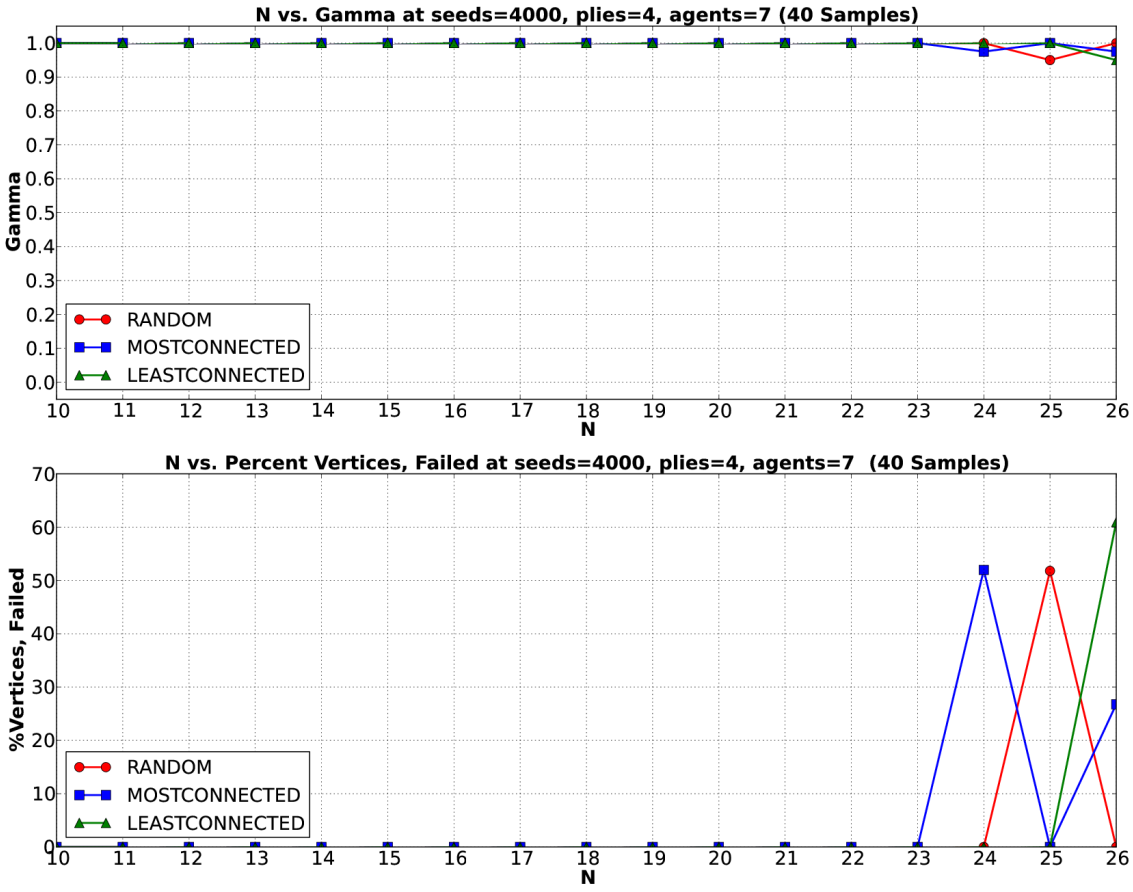

Figure 27: Percent of Vertices Searched Failed Factoring

5 Conclusions

- An analysis of the PG_N^2 graph vertex degrees showed that the average vertex degree fit a linear function over N:

$$average\ degree = 1.726 * N - 2.209 \qquad (8)$$

- In a preliminary experiment testing factoring by neighborhood search, each of the test prime numbers were successfully factored with a simple seed choice heuristic and no pruning of the search space during breadth first search. All of the factors were found within four plies of the initial search seed (Section 4).
- The heuristic of most connected pick outperforms random pick and least connected pick while searching between fifteen and twenty percent of the total vertices in a PG_N^2 graph (Figure 26).
- The heuristic of most connected pick outperforms random pick and least connected pick by 30% in terms of seconds at $N = 26$ (Figure 24).
- The heuristic of most connected pick shows a performance advantage over least connected and random connected pick heuristics in terms of average depth of search (Figure 25).
- When search fails, the number of vertices visited increases as a percentage of the vertices in the graph (Figure 27), relative to successful searches (Figure 26).
- Visually, there is a correlation between time of search and depth of search in Figure 25. The most connected pick heuristic also outperforms at larger N in Figure 26. This consistent result increases confidence in our test methodology and execution.
- The upper bound for the diameter of a PG_N^2 graph is:

$$D_{ub} = \lceil \frac{N}{2} \rceil + ((N+1)\ mod\ 2) \qquad (9)$$

6 Future Work

6.1 Extrapolation

All the experiments so far have been run using lists of prime numbers contained in each PG_N^2 under investigation. We would like to run experiments beyond PG_{34}^2 which will require estimations of average vertex degree in the graph, the number of start seeds required to reach a minimum factor performance and the number of plies each neighborhood search should expand.

In our experiments, the best performing seed pick heuristic was found to be most connected. In order to use this heuristic in searching larger graphs $N > 34$, we will need to choose seed vertices in the graph based on an average graph vertex degree. Using the linear fit from Equation 7, we will consider choosing prime numbers in these PG_N^2 graphs with greater than the average vertex degree or some factor greater than the average degree.

Future work would explore the performance of this algorithm by changing the value of the number of seeds (m) at different values of N. The resulting table would be similar to the table of Appendix C without changing the number of plies expanded. This new table would represent γ as a function of m, and N. The number of experiments required would be the number of values of m times the number of values of N times the number of experiments at each (m, N) pair. Further work would include changing the value of the number of plies expanded pl creating (m, N, pl) triples, increasing the total number of experiments by pl. A sensitivity analysis could be performed to investigate which of these parameters has the greatest effect on the success of factoring. Resources beyond what were available for this thesis would have to be available.

Compute resources will also have to be extrapolated from further experiments to determine the number of processors, cores, threads (agents) and depending on the machine architecture how to manage local and global memory. The purpose of the experiments in this thesis was to build a foundation for this type of extrapolation by working from relatively small N.

If we wanted to factor a number of 128 bits, a search of PG_{64}^2 would be necessary. In order to use the degree based seedpick heuristics, extrapolation of what the average degree in a PG_{64}^2 would be would be necessary. For example, choosing seeds less than this value would be similar to the least connected seedpick heuristic, since is is not realistic to search the graph to find a complete order of the seeds by connectivity.

6.2 Exploring Alternate Compute Architectures

This thesis demonstrates an algorithm based on shared memory multiprocessor and multicore machines.

One possibility is to explore data parallel solutions (or SIMD) on contemporary GPU architectures. An example is the operation of expanding the next ply of search through xor masking of the current prime. This operation can be done using xor function with the current prime and a vector of pre-calculated masks as the function input. This would create a new vector, which could be input to a data parallel version of Rabin-Miller or equivalent to create the next ply of the search. Now, each ply is a vector which can be input to a data parallel function which performs trial division of the current semiprime which we are attempting to factor.

Any implementation of this type of algorithm will need to manage a frontier queue and a visited queue. It is possible that these queues will need to be managed in host memory, not in the local GPU memory and any computational performance gains may be lost to data management overhead. Any future work in this area will need to investigate this issue.

6.3 Pruning

The upper bound diameter of a PG_N^2 graph is been shown by Equation 6. We hypothesize that an algorithm could use this to reduce redundant search. By adding the distance from the first selected seed prime (or anchor prime) to the current seed prime the search could be limited not to search beyond the diameter of the graph from this first selected seed (or anchor). As neighborhood search progresses, each expanded ply increases the distance towards or away from the maximum diameter of the current graph, as measured from the anchor prime. Not allowing any neighborhood search to proceed beyond the distance from the anchor prime to the upper bound of the PG_N^2 graph may allow an opportunity for pruning the search.

7 References

[1] Manindra Agrawal, Neeraj Kayal, and Nitin Saxena. Primes is in p. *Ann. of Math*, 2:781–793, 2002. 34

[2] Burton H. Bloom. Space/Time Trade-offs in Hash Coding with Allowable Errors. *Communications of the ACM*, 13(7):422–426, 1970. ISSN 0001-0782. doi: http://doi.acm.org/10.1145/362686.362692. 51

[3] Arthur Cayley. A Theorem on Trees. *Quarterly Journal of Pure and Applied Mathematics*, 23:376–378, 1889. 20

[4] Rosetta Code. Millerrabin primality test — wikipedia, the free encyclopedia, 2011. URL http://rosettacode.org/wiki/Miller-Rabin_primality_test. [Online; accessed 5-April-2011]. 70

[5] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, second edition*. The MIT Press, Cambridge, MA, USA, 2001. 34, 70

[6] Richard Crandall and Carl B. Pomerance. *Prime Numbers: A Computational Perspective*. Springer Science+Business Media, Inc., New York, NY, USA, 2005. springeronline.com. 9, 10, 12, 34, 70

[7] John Derbyshire. *Prime Obsession: Bernhard Riemann and the Greatest Unsolved Problem in Mathematics*. Penguin, New York, NY, USA, 2004. 14, 25

[8] Carl Friedrich Gauss. *Disquisitiones arithmeticae*. Springer, New York, NY,

USA, 1986. Translated by Waterhouse, William C. From the Latin edition of 1870 edited by E.C.J. Schering. 1

[9] R W. Hamming. Error Detecting and Error Correcting Codes. *The Bell System Technical Journal*, 29(2):147–160, 1950. 2

[10] D. Hankerson, A. Menezes, and S. Vanstone. *Guide to Elliptic Curve Cryptography*. Springer-Verlag, New York, 2004. 10, 12

[11] Frank Harary. *Graph Theory*. Addison-Wesley Publishing Company, Reading, MA, USA, 1969. 5, 20

[12] Julian Havil. *Exploring Euler's Constant*. Princeton University Press, Princeton, NJ, USA, 2003. www.pupress.pnnceton.edu. 25

[13] Robert E. Jamison and Gretchen L. Matthews. Distance k Colorings of Hamming Graphs. In *Proceedings of the Thirty-Seventh Southeastern International Conference on Combinatorics, Graph Theory and Computing. Congr. Numer. 183*, 2006. 12, 13

[14] Hendrik W. Lenstra, Jr. Factoring Integers with Elliptic Curves. *The Annals of Mathematics*, 126:649–673, 1987. 9, 10

[15] Hendrik W. Lenstra, Jr. Primality testing with gaussian periods. In *Proceedings of the 22nd Conference Kanpur on Foundations of Software Technology and Theoretical Computer Science*, FST TCS '02, pages 1–, London, UK, 2002. Springer-Verlag. ISBN 3-540-00225-1. URL http://portal.acm.org/citation.cfm?id=646840.708803. 34

[16] Dana Mackenzie. Mathematics: Homage to an itinerant master. *Sci-*

ence, 275(5301):759, 1997. doi: 10.1126/science.275.5301.759. URL http://www.sciencemag.org/content/275/5301/759.short. 1

[17] J. Pollard. A Monte Carlo method for factorization. *BIT Numerical Mathematics*, 32:918–924, 1975. 9, 12

[18] Carl Pomerance. A Tale of Two Sieves. *Notices Amer. Math. Soc*, 43:1473–1485, 1996. 9, 10

[19] ©2007 Bryant York. Used by permission. ix, 3, 15

[20] ©2009 Bryant York. Used by permission. x, 47, 49

[21] Wikipedia. Lenstra Elliptic Curve Factorization, 2011. URL https://secure.wikimedia.org/wikipedia. 10

[22] Bryant York. Notes on Prime Number Graphs. 1999-2011. 5

[23] Bryant York, 2011. Personal correspondence and notes of Bryant York. 10, 11

[24] Bryant York and Keith Wilson. Prime number graphs, self similarity, and factoring (unpublished). Ninety Minute Technical Seminar at Portland State University, January 2010. 14, 17, 19, 24

Appendices

A Size of PG_N^2 Graphs

Table 10: **Vertices in** PG_N^2

N	$\mid PG_N^2 \mid$
3	3
4	5
5	10
6	17
7	30
8	53
9	96
10	171
11	308
12	563
13	1027
14	1899
15	3511
16	6541
17	12250
18	22999
19	43382
20	82024
21	155610
22	295946
23	564162
24	1077870
25	2063688
26	3957808
27	7603552
28	14630842
29	28192749
30	54400027
31	105097564
32	203280220
33	400072673
34	803536681
	End

B Rabin-Miller Algorithm

The Rabin-Miller test is more accurately called a 'compositeness' test in that it is trying to find a witness that divides the input number. If it does, then the algorithm returns false. [4] [6] [5]

INPUT: $N > 3$, an odd integer to be tested for primality;
INPUT: k, a parameter that determines the accuracy of the test;
OUTPUT: composite if n is composite, otherwise probably prime
write $n-1$ as $2^s t$ with t odd by factoring powers of 2 from $n-1$
loop
  repeat k times:
  pick a random integer a in the range $[2, n-2]$
  $x \leftarrow a^t \ mod \ n$
  if $x = 1$ or $x = n-1$ **then**
    do next loop
  end if
  for $r = 1 \ldots s-1$ **do**
    $x \leftarrow x^2 \ mod \ n$
    if $x = 1$ **then**
      return composite
    end if
    if $x = n-1$ **then**
      do next loop
    end if
  end for
  return composite
end loop
return probably prime

Figure 28: Rabin-Miller Primality Test

C Independent Limited BFS Results with Five Agents

Table 11: **Independent Limited BFS Results with Five Agents**

N	Num. Clusters	Max. Ply	Found Ply	Seed	HD (seed,F_s)	A	B	S = A * B
24	1	1	nf	nf	nf	nf	nf	9990157
24	2	1	nf	nf	nf	nf	nf	9990157
24	3	1	nf	nf	nf	nf	nf	9990157
24	4	1	nf	nf	nf	nf	nf	9990157
24	5	1	nf	nf	nf	nf	nf	9990157
24	1	2	nf	nf	nf	nf	nf	9990157
24	2	2	nf	nf	nf	nf	nf	9990157
24	3	2	2	41	4	3119	3203	9990157
24	4	2	2	41	4	3119	3203	9990157
24	5	2	2	59	4	3119	3203	9990157
24	1	3	nf	nf	nf	nf	nf	9990157
24	2	3	nf	nf	nf	nf	nf	9990157
24	3	3	2	41	4	3119	3203	9990157
24	4	3	2	41	4	3119	3203	9990157
24	5	3	2	41	4	3119	3203	9990157
24	1	4	nf	nf	nf	nf	nf	9990157
24	2	4	nf	nf	nf	nf	nf	9990157
24	3	4	2	41	4	3119	3203	9990157
24	4	4	2	41	4	3119	3203	9990157
24	5	4	2	59	4	3119	3203	9990157
24	1	5	nf	nf	nf	nf	nf	9990157
24	2	5	nf	nf	nf	nf	nf	9990157
24	3	5	2	41	4	3119	3203	9990157
24	4	5	2	41	4	3119	3203	9990157
24	5	5	2	59	4	3119	3203	9990157
25	1	1	nf	nf	nf	nf	nf	32639989
25	2	1	nf	nf	nf	nf	nf	32639989
25	3	1	nf	nf	nf	nf	nf	32639989
25	4	1	nf	nf	nf	nf	nf	32639989
25	5	1	nf	nf	nf	nf	nf	32639989
25	1	2	nf	nf	nf	nf	nf	32639989
25	2	2	2	7573	4	5507	5927	32639989
25	3	2	2	7573	4	5507	5927	32639989
25	4	2	2	7573	4	5507	5927	32639989
25	5	2	2	7573	4	5507	5927	32639989

Continued on next page...

Table 11 – continued								
N	Num. Clusters	Max. Ply	Found Ply	Seed	HD (seed,F_s)	A	B	S = A * B
25	1	3	nf	nf	nf	nf	nf	32639989
25	2	3	2	7573	4	5507	5927	32639989
25	3	3	2	7573	4	5507	5927	32639989
25	4	3	2	7573	4	5507	5927	32639989
25	5	3	2	7573	4	5507	5927	32639989
25	1	4	nf	nf	nf	nf	nf	32639989
25	2	4	2	7573	4	5507	5927	32639989
25	3	4	2	7573	4	5507	5927	32639989
25	4	4	2	7573	4	5507	5927	32639989
25	5	4	2	7573	4	5507	5927	32639989
25	1	5	nf	nf	nf	nf	nf	32639989
25	2	5	2	7573	4	5507	5927	32639989
25	3	5	2	7573	4	5507	5927	32639989
25	4	5	2	7573	4	5507	5927	32639989
25	5	5	2	7573	4	5507	5927	32639989
26	1	1	nf	nf	nf	nf	nf	55127929
26	2	1	1	7573	2	7607	7247	55127929
26	3	1	1	7573	2	7607	7247	55127929
26	4	1	1	7573	2	7607	7247	55127929
26	5	1	1	7573	2	7607	7247	55127929
26	1	2	nf	nf	nf	nf	nf	55127929
26	2	2	1	7573	2	7607	7247	55127929
26	3	2	1	7573	2	7607	7247	55127929
26	4	2	1	7573	2	7607	7247	55127929
26	5	2	1	7573	2	7607	7247	55127929
26	1	3	nf	nf	nf	nf	nf	55127929
26	2	3	1	7573	2	7607	7247	55127929
26	3	3	1	7573	2	7607	7247	55127929
26	4	3	1	7573	2	7607	7247	55127929
26	5	3	1	7573	2	7607	7247	55127929
26	1	4	nf	nf	nf	nf	nf	55127929
26	2	4	1	7573	2	7607	7247	55127929
26	3	4	1	7573	2	7607	7247	55127929
26	4	4	1	7573	2	7607	7247	55127929
26	5	4	1	7573	2	7607	7247	55127929
26	1	5	nf	nf	nf	nf	nf	55127929
26	2	5	1	7573	2	7607	7247	55127929
26	3	5	1	7573	2	7607	7247	55127929
26	4	5	1	7573	2	7607	7247	55127929
26	5	5	1	7573	2	7607	7247	55127929

Continued on next page...

Table 11 – continued								
N	Num. Clusters	Max. Ply	Found Ply	Seed	HD (seed,F_s)	A	B	S = A * B
27	1	1	nf	nf	nf	nf	nf	119094961
27	2	1	nf	nf	nf	nf	nf	119094961
27	3	1	nf	nf	nf	nf	nf	119094961
27	4	1	nf	nf	nf	nf	nf	119094961
27	5	1	nf	nf	nf	nf	nf	119094961
27	1	2	nf	nf	nf	nf	nf	119094961
27	2	2	nf	nf	nf	nf	nf	119094961
27	3	2	nf	nf	nf	nf	nf	119094961
27	4	2	2	59	4	10559	11279	119094961
27	5	2	2	59	4	10559	11279	119094961
27	1	3	nf	nf	nf	nf	nf	119094961
27	2	3	nf	nf	nf	nf	nf	119094961
27	3	3	3	41	6	10559	11279	119094961
27	4	3	3	41	6	10559	11279	119094961
27	5	3	2	59	4	10559	11279	119094961
27	1	4	nf	nf	nf	nf	nf	119094961
27	2	4	4	7573	7	10559	11279	119094961
27	3	4	3	41	6	10559	11279	119094961
27	4	4	3	41	6	10559	11279	119094961
27	5	4	3	41	6	10559	11279	119094961
27	1	5	nf	nf	nf	nf	nf	119094961
27	2	5	4	7573	7	10559	11279	119094961
27	3	5	3	41	6	10559	11279	119094961
27	4	5	3	41	6	10559	11279	119094961
27	5	5	2	59	4	10559	11279	119094961
28	1	1	nf	nf	nf	nf	nf	220023049
28	2	1	nf	nf	nf	nf	nf	220023049
28	3	1	nf	nf	nf	nf	nf	220023049
28	4	1	nf	nf	nf	nf	nf	220023049
28	5	1	nf	nf	nf	nf	nf	220023049
28	1	2	nf	nf	nf	nf	nf	220023049
28	2	2	2	7573	4	15383	14303	220023049
28	3	2	2	7573	4	15383	14303	220023049
28	4	2	2	7573	4	15383	14303	220023049
28	5	2	2	7573	4	15383	14303	220023049
28	1	3	nf	nf	nf	nf	nf	220023049
28	2	3	2	7573	4	15383	14303	220023049
28	3	3	2	7573	4	15383	14303	220023049
28	4	3	2	7573	4	15383	14303	220023049
28	5	3	2	7573	4	15383	14303	220023049

Continued on next page...

Table 11 – continued								
N	Num. Clusters	Max. Ply	Found Ply	Seed	HD (seed,F_s)	A	B	S = A * B
28	1	4	nf	nf	nf	nf	nf	220023049
28	2	4	2	7573	4	15383	14303	220023049
28	3	4	2	7573	4	15383	14303	220023049
28	4	4	2	7573	4	15383	14303	220023049
28	5	4	2	7573	4	15383	14303	220023049
28	1	5	nf	nf	nf	nf	nf	220023049
28	2	5	2	7573	4	15383	14303	220023049
28	3	5	2	7573	4	15383	14303	220023049
28	4	5	2	7573	4	15383	14303	220023049
28	5	5	2	7573	4	15383	14303	220023049
29	1	1	nf	nf	nf	nf	nf	526612501
29	2	1	nf	nf	nf	nf	nf	526612501
29	3	1	nf	nf	nf	nf	nf	526612501
29	4	1	nf	nf	nf	nf	nf	526612501
29	5	1	nf	nf	nf	nf	nf	526612501
29	1	2	nf	nf	nf	nf	nf	526612501
29	2	2	nf	nf	nf	nf	nf	526612501
29	3	2	nf	nf	nf	nf	nf	526612501
29	4	2	nf	nf	nf	nf	nf	526612501
29	5	2	nf	nf	nf	nf	nf	526612501
29	1	3	nf	nf	nf	nf	nf	526612501
29	2	3	3	7573	6	22739	23159	526612501
29	3	3	3	7573	6	22739	23159	526612501
29	4	3	3	7573	6	22739	23159	526612501
29	5	3	3	7573	6	22739	23159	526612501
29	1	4	nf	nf	nf	nf	nf	526612501
29	2	4	3	7573	6	22739	23159	526612501
29	3	4	3	7573	6	22739	23159	526612501
29	4	4	3	7573	6	22739	23159	526612501
29	5	4	3	7573	6	22739	23159	526612501
29	1	5	nf	nf	nf	nf	nf	526612501
29	2	5	3	7573	6	22739	23159	526612501
29	3	5	3	7573	6	22739	23159	526612501
29	4	5	3	7573	6	22739	23159	526612501
29	5	5	3	7573	6	22739	23159	526612501
30	1	1	nf	nf	nf	nf	nf	571077781
30	2	1	nf	nf	nf	nf	nf	571077781
30	3	1	nf	nf	nf	nf	nf	571077781
30	4	1	nf	nf	nf	nf	nf	571077781
30	5	1	nf	nf	nf	nf	nf	571077781
Continued on next page...								

Table 11 – continued								
N	Num. Clusters	Max. Ply	Found Ply	Seed	HD (seed,F_s)	A	B	S = A * B
30	1	2	nf	nf	nf	nf	nf	571077781
30	2	2	nf	nf	nf	nf	nf	571077781
30	3	2	nf	nf	nf	nf	nf	571077781
30	4	2	nf	nf	nf	nf	nf	571077781
30	5	2	nf	nf	nf	nf	nf	571077781
30	1	3	nf	nf	nf	nf	nf	571077781
30	2	3	nf	nf	nf	nf	nf	571077781
30	3	3	nf	nf	nf	nf	nf	571077781
30	4	3	nf	nf	nf	nf	nf	571077781
30	5	3	nf	nf	nf	nf	nf	571077781
30	1	4	nf	nf	nf	nf	nf	571077781
30	2	4	4	7573	8	23159	24659	571077781
30	3	4	4	7573	8	23159	24659	571077781
30	4	4	4	7573	8	23159	24659	571077781
30	5	4	4	7573	8	23159	24659	571077781
30	1	5	nf	nf	nf	nf	nf	571077781
30	2	5	4	7573	8	23159	24659	571077781
30	3	5	4	7573	8	23159	24659	571077781
30	4	5	4	7573	8	23159	24659	571077781
30	5	5	4	7573	8	23159	24659	571077781
31	1	1	nf	nf	nf	nf	nf	1298487901
31	2	1	nf	nf	nf	nf	nf	1298487901
31	3	1	nf	nf	nf	nf	nf	1298487901
31	4	1	nf	nf	nf	nf	nf	1298487901
31	5	1	nf	nf	nf	nf	nf	1298487901
31	1	2	nf	nf	nf	nf	nf	1298487901
31	2	2	nf	nf	nf	nf	nf	1298487901
31	3	2	nf	nf	nf	nf	nf	1298487901
31	4	2	nf	nf	nf	nf	nf	1298487901
31	5	2	nf	nf	nf	nf	nf	1298487901
31	1	3	nf	nf	nf	nf	nf	1298487901
31	2	3	3	7573	6	34583	37547	1298487901
31	3	3	3	7573	6	34583	37547	1298487901
31	4	3	3	7573	6	34583	37547	1298487901
31	5	3	3	7573	6	34583	37547	1298487901
31	1	4	nf	nf	nf	nf	nf	1298487901
31	2	4	3	7573	6	34583	37547	1298487901
31	3	4	3	7573	6	34583	37547	1298487901
31	4	4	3	7573	6	34583	37547	1298487901
31	5	4	3	7573	6	34583	37547	1298487901

Continued on next page...

Table 11 – continued								
N	Num. Clusters	Max. Ply	Found Ply	Seed	HD (seed,F_s)	A	B	S = A * B
31	1	5	nf	nf	nf	nf	nf	1298487901
31	2	5	3	7573	6	34583	37547	1298487901
31	3	5	3	7573	6	34583	37547	1298487901
31	4	5	3	7573	6	34583	37547	1298487901
31	5	5	3	7573	6	34583	37547	1298487901
32	1	1	nf	nf	nf	nf	nf	3130236121
32	2	1	nf	nf	nf	nf	nf	3130236121
32	3	1	nf	nf	nf	nf	nf	3130236121
32	4	1	nf	nf	nf	nf	nf	3130236121
32	5	1	nf	nf	nf	nf	nf	3130236121
32	1	2	nf	nf	nf	nf	nf	3130236121
32	2	2	nf	nf	nf	nf	nf	3130236121
32	3	2	nf	nf	nf	nf	nf	3130236121
32	4	2	nf	nf	nf	nf	nf	3130236121
32	5	2	nf	nf	nf	nf	nf	3130236121
32	1	3	nf	nf	nf	nf	nf	3130236121
32	2	3	3	7573	6	56807	55103	3130236121
32	3	3	3	7573	6	56807	55103	3130236121
32	4	3	3	7573	6	56807	55103	3130236121
32	5	3	3	7573	6	56807	55103	3130236121
32	1	4	nf	nf	nf	nf	nf	3130236121
32	2	4	3	7573	6	56807	55103	3130236121
32	3	4	3	7573	6	56807	55103	3130236121
32	4	4	3	7573	6	56807	55103	3130236121
32	5	4	3	7573	6	56807	55103	3130236121
32	1	5	nf	nf	nf	nf	nf	3130236121
32	2	5	3	7573	6	56807	55103	3130236121
32	3	5	3	7573	6	56807	55103	3130236121
32	4	5	3	7573	6	56807	55103	3130236121
32	5	5	3	7573	6	56807	55103	3130236121
								End

D Heuristic Comparison Table Experiment Results

$numsamples$: 40

$maxply$: 4

$maxN$: 26

$maxagents$: 7

seedpickH: RANDOMPICK

Table 12: PG_N^2 **Random Pick Heurisitic Sweep**

N	Plys	Seeds	Agents	SuccProb	AvgVerts	AvgSecs	AvgSeeds	AvgPly
10	4	4000	7	1.0	49.58	0.0	15.88	0.875
11	4	4000	7	1.0	98.6	0.0	11.23	1.325
12	4	4000	7	1.0	103.3	0.0	8.275	1.35
13	4	4000	7	1.0	156.89	0.025	10.55	1.45
14	4	4000	7	1.0	432.35	0.0	9.55	1.95
15	4	4000	7	1.0	908.78	0.025	12.225	2.075
16	4	4000	7	1.0	1507.73	0.025	12.6	2.325
17	4	4000	7	1.0	3275.6	0.1	26.6	2.75
18	4	4000	7	1.0	5554.23	0.1	7.35	2.9
19	4	4000	7	1.0	9494.83	0.25	22.675	2.875
20	4	4000	7	1.0	21121.45	0.7	70.075	3.275
21	4	4000	7	1.0	36249.33	2.65	11.65	3.3
22	4	4000	7	1.0	66799.45	9.25	9.875	3.55
23	4	4000	7	1.0	122712.5	22.025	12.65	3.725
24	4	4000	7	1.0	243172.15	50.775	15.775	3.8
25	4	4000	7	0.95	358584.42	91.075	14.342	3.737
26	4	4000	7	1.0	873473.05	234.325	52.8	3.85
								End

$numsamples$: 40

$maxply$: 4

$maxN$: 26

$maxagents$: 7

seedpickH: MOSTCONNECTEDPICK

Table 13: PG_N^2 **Most Connected Heuristic Sweep**

N	Plys	Seeds	Agents	SuccProb	AvgVerts	AvgSecs	AvgSeeds	AvgPly
10	4	4000	7	1.0	60.23	0.0	11.05	1.5
11	4	4000	7	1.0	118.7	0.0	8.9	1.5
12	4	4000	7	1.0	181.55	0.0	13.15	2.2
13	4	4000	7	1.0	209.8	0.0	9.2	1.6
14	4	4000	7	1.0	524.8	0.0	10.05	2.075
15	4	4000	7	1.0	706.23	0.0	7.825	2.125
16	4	4000	7	1.0	1972.78	0.025	14.075	2.775
17	4	4000	7	1.0	3130.0	0.025	13.325	2.925
18	4	4000	7	1.0	4649.85	0.125	28.775	2.7
19	4	4000	7	1.0	12382.0	0.325	39.25	3.25
20	4	4000	7	1.0	16749.98	0.425	103.25	3.15
21	4	4000	7	1.0	21238.95	1.225	80.175	3.075
22	4	4000	7	1.0	84364.48	11.05	41.775	3.35
23	4	4000	7	1.0	108049.13	16.4	46.075	3.65
24	4	4000	7	0.975	172139.46	34.05	165.846	3.436
25	4	4000	7	1.0	272057.08	57.525	66.475	3.625
26	4	4000	7	0.975	661163.67	171.35	99.59	3.667
								End

$numsamples$: 40

$maxply$: 4

$maxN$: 26

$maxagents$: 7

seedpickH: LEASTCONNECTEDPICK

Table 14: PG_N^2 **Least Pick Heurisitic Sweep**

N	Plys	Seeds	Agents	SuccProb	AvgVerts	AvgSecs	AvgSeeds	AvgPly
10	4	4000	7	1.0	37.2	0.0	8.775	0.85
11	4	4000	7	1.0	78.18	0.0	9.975	1.45
12	4	4000	7	1.0	131.8	0.0	9.375	1.75
13	4	4000	7	1.0	189.08	0.0	11.675	2.025
14	4	4000	7	1.0	600.025	0.0	7.725	2.725
15	4	4000	7	1.0	833.28	0.0	9.1	2.975
16	4	4000	7	1.0	1233.55	0.05	36.825	2.725
17	4	4000	7	1.0	2591.83	0.075	131.425	3.2
18	4	4000	7	1.0	6595.525	0.15	95.775	3.575
19	4	4000	7	1.0	10514.28	0.2	176.375	3.625
20	4	4000	7	1.0	20165.75	0.6	86.825	3.775
21	4	4000	7	1.0	36666.43	2.425	141.175	3.85
22	4	4000	7	1.0	71958.15	10.5	199.775	3.8
23	4	4000	7	1.0	152047.1	27.575	236.45	3.775
24	4	4000	7	1.0	216741.5	43.5	313.6	3.85
25	4	4000	7	1.0	356325.15	81.025	447.5	3.8
26	4	4000	7	0.95	859133.74	242.1	943.868	3.868
								End

E Development Tree For Prime Graph Investigation

Summary of structure for prime graph investigation development environment[10]

E.1 Source Code Management

The source code management system chosen is 'git.' This is a distributed source code mangement system in common use at Portland State.

Git Source Code Management. (http://git-scm.com)

The repository is called 'pgdev.git' and is located on my home machine 'magpie'. To clone a local copy:

git clone git+ssh://magpie/vc/git/pgdev.git

Access to magpie is controlled by ssh public key.

E.2 Development Tree - Goals

E.3 Grouping Source By Language Type, Not Task

Source code is grouped by language type, not task. This enables and encourages the reuse of components in multiple tools sharing the same language.

E.4 Separating Data from Code

Large files of source data are required in this investigation. Keeping this information out of the code develpment subtree will prevent duplication and allow future modifications to how the data is managed separate from the source code. For example, the file primes3-32.txt is a 2GB+ file of all the primes less than 2^{32}. Having this file available decreases the runtime of many analysis by not having to generate primes every execution. Other large files include complete graph representations for PG_N^2 in multiple formats.

[10]This repository is only for development. There is another repository for M.S. research and writing.

E.5 Development Tree - Structure

The root of the development tree is called 'pgdev' as shown in Figure 29. The diagram shows how the different source material is partitioned.

Each first level (gray) and second level (red) node of the tree are directories. Below that are directories or files as appropriate.

E.6 C/C++ Justification

C/C++[11] offers the widest application of the tools investigated[12] with clear access to multithreading and GPU programming. Other languages will also be used but the bulk of development exists today in C/C++.

E.7 Python

Python is used for:

- Scripting and batch control.
- Analysis using the numpy and scipy libraries.
- Plots and figures using the matplotlib libraries.
- Graph analysis using the python networkx libraries.

E.8 R

R is a contemporary statistical analysis package.

The R Project. (www.r-project.org)

E.9 C/C++ Status

Current work is increasing use of C++ container classes and iterators such as queue and vector, where previous work used C and custom data structures with pointers.

[11] Using gcc
[12] Python, Haskell and C/C++ were all considered

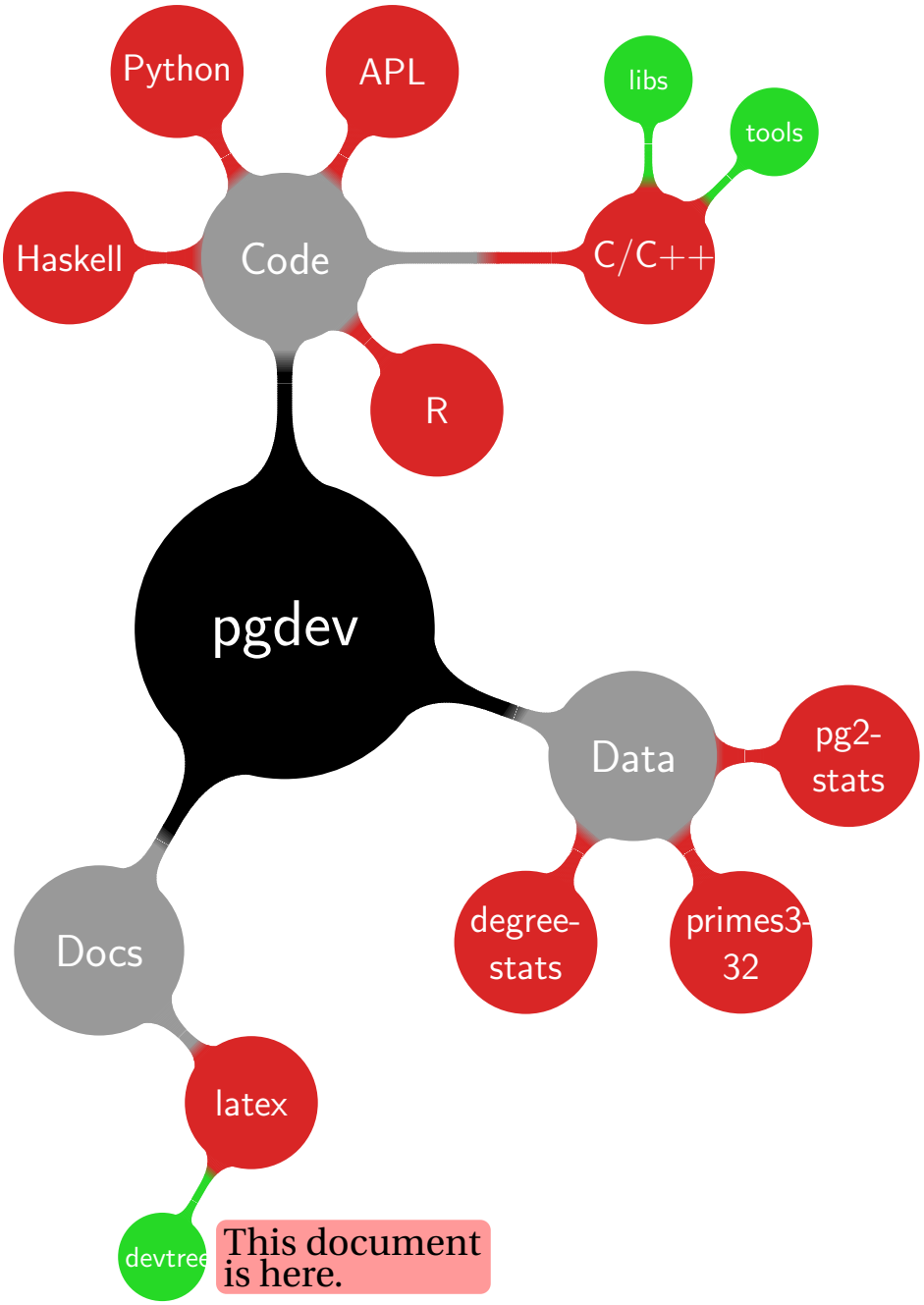

Figure 29: Directory Tree Structure for pgdev

The restrained use of C++ STL will create safer, more readable code and also be more portable over time.

The gmp header gmpxx.h is necessary when compiling C++ with the GMP Bignum library.

F Hardware

gottbrath.cs.pdx.edu:

- Four Intel® Xeon® E7310 CPUs @ 1.60GHz
- Each cpu has 4MB of L2 cache.
- Each cpu has four cores for a total of sixteen processors.
- Thirty-two gigabytes of RAM
- SAS Disk(s): 2 Mirrored Seagate® ST9146802SS/10,000 RPM/300MBPS/3.8 MS Seek

bailin.cs.pdx.edu:

- Four Intel® Xeon® E7310 CPUs @ 1.60GHz
- Each cpu has four cores for a total of sixteen processors.
- Sixteen gigabytes of RAM
- SAS Disk(s): 2 Mirrored Seagate® ST9146802SS/10,000 RPM/300MBPS/3.8 MS Seek

magpie:

- One Intel® Core™ i7 930 CPU @ 2.80GHz
- Each cpu has 8MiB of L2 cache and 64KiB of L1 cache.
- Each cpu has four cores for a total of four processors.
- Each core has hyperthreading capabilities for a total of eight logical processors.
- Six gigabytes of RAM.

G Disk Resources

A 100 gigabyte research space has been set up at /stash/yorkrsch on the cs.pdx.edu research data server. Members of group 'yorkrsch' have access to this space and it is backed up according to The CAT (http://cat.pdx.edu) policy. A copy of the code repository is regularly pulled to the kwilson/Projects/msthesis directory.

A 100 gigabyte partition has been created for group *yorkrsch* on each of the SAS disks on *gottbrath* and *bailin* (See Appendix F on page 84).