MATHEMATICAL ENGINEERING
TECHNICAL REPORTS

Computational Results for Gaussian Moat Problem

Nobuyuki TSUCHIMURA

METR 2004–13 March 2004

Computational Results for Gaussian Moat Problem

Nobuyuki TSUCHIMURA
Department of Mathematical Informatics
Graduate School of Information Science and Technology
The University of Tokyo
tutimura@mist.i.u-tokyo.ac.jp

March, 2004

Abstract

"Can one walk to infinity on Gaussian primes taking steps of bounded length?" We adopted computational techniques to probe into this open problem. We propose an efficient method to search for the farthest point reachable from the origin, which can be parallelized easily, and have confirmed the existence of a moat of width $k = \sqrt{36}$, whereas the best previous result was $k = \sqrt{26}$ due to Gethner et al. A refinement of Vardi's estimate for the farthest distance reachable from the origin is proposed. The proposed estimate incorporates discreteness into Vardi's that is based on percolation theory.

1 Introduction

The question addressed in this paper is whether one can walk to infinity on Gaussian primes taking steps of bounded length. More precisely, this problem may be formulated as follows. A Gaussian integer means a complex number $a + bi$ with integers a and b. A Gaussian prime means a Gaussian integer that cannot be decomposed into a product of two Gaussian integers in a nontrivial way, i.e., with factors distinct from $\pm 1, \pm i$, where $i = \sqrt{-1}$. Consider a graph G drawn on the complex plane, of which the vertex set is the set of all Gaussian primes augmented by the origin. Two vertices are connected by an edge if the distance between them is less than or equal to a specified parameter k, which we call the step size. Fig.1 illustrates this graph for $k = \sqrt{16}$ in the first octant. The question is whether the graph with a specified step size k contains a path from the origin that extends to infinity.

Nonexistence of such an infinite path implies the existence of a moat of width k surrounding the component of the origin. According to [1][2][4][5],

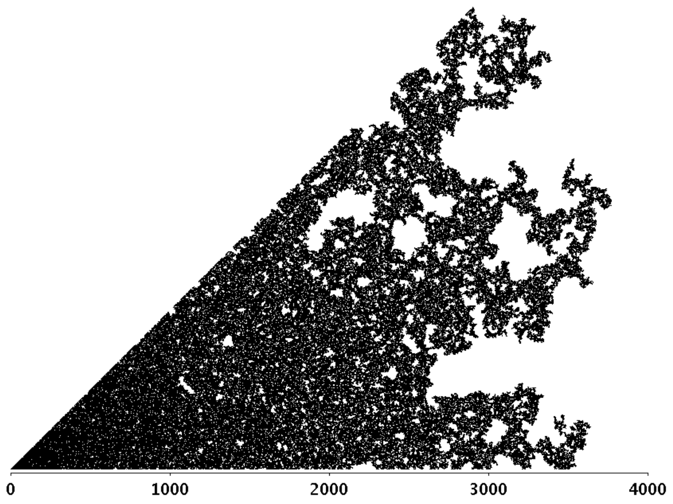


Figure 1: Illustration of the graph G for $k = \sqrt{16}$

this problem was posed by Basil Gordon in 1962 at International Congress of Mathematicians in Stockholm, and still remains open. It seems, however, that the opinions in the literature [1][2][4] are inclined to the negative answer.

Computational results on this problem are reported in [1][5], which can be summarized in the following table. We denote by $\xi(k)$ the farthest point reached from the origin for step size k.

Component of origin using distance k	Farthest point reached $\xi(k)$	Farthest distance reached $\lvert\xi(k)\rvert$	Total size of the component$*$
$\sqrt{1}$	$2+i$	2.23	2
$\sqrt{2}$	$11+4i$	11.70	14
$\sqrt{4}$	$42+17i$	45.31	92
$\sqrt{8}$	$84+41i$	93.47	380
$\sqrt{10}$	$976+311i$	1024.35	31221
$\sqrt{16}$	$3297+2780i$	4312.61	347638
$\sqrt{18}$	$8174+6981i$	10749.4	2386129
$\sqrt{20}$	$109677+64268i$	127120	Finite
$\sqrt{26}$	??	≤ 5586757	Finite

$*$ the number of points distinct from the origin

The 3rd row of the table, for example, shows that for $k = \sqrt{4}$ the farthest point reachable from the origin by a path on the graph is $42+17i$, with the distance from the origin being equal to 45.31, and that the component of the origin contains 92 vertices lying in the first octant $\{z \in \mathbf{C} \mid 0 \leq \arg z \leq \pi/4\}$. Note that we may restrict ourselves to the first octant by the symmetry of the problem. For $k = \sqrt{26}$ the farthest point reached and the total size of the component are not known, but only an upper bound of 5586757 on its distance from the origin is known. For $k = \sqrt{20}$ the exact value of the total size of the component is unknown but the known farthest point implies that it is finite.

The last two rows of the table have been completed and three more row has been added in the present work.

Component of origin using distance k	Farthest point reached $\xi(k)$	Farthest distance reached $\lvert\xi(k)\rvert$	Total size of the component*
$\sqrt{20}$	$120510 + 57857i$	133679.065	273791623
$\sqrt{26}$	$943460 + 376039i$	1015638.765	14542615005
$\sqrt{32}$	$2106442 + 1879505i$	2823054.542	103711268594
$\sqrt{34}$	??	< 24289452	Finite
$\sqrt{36}$	??	< 80015782	Finite

$*$ the number of points distinct from the origin

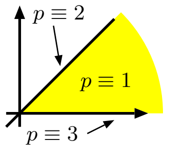
In Section 2 we summarize basic facts about Gaussian primes and describe how to generate them efficiently. Section 3 presents our method of computing the component of the origin. Section 4 presents the detail of the result and the statistical information which is necessary for tuning up the computation. Section 5 proposes an improve estimate for the farthest distance reachable from the origin.

2 Generating Methods of Gaussian Primes

A Gaussian integer is a number of the form $a + bi$, where a and b are integers and i is the square root of -1. A Gaussian prime is a Gaussian integer $a + bi$ which cannot be divided by any Gaussian integer, excepting for $\pm 1, \pm i, \pm(a + bi)$, and $\pm(b - ai)$. When $a + bi$ is a Gaussian prime, then $\pm a \pm bi$ and $\pm b \pm ai$ are also Gaussian primes. Hence we may restrict our attention to the first octant ($0 \leq b \leq a$).

The sieve of Eratosthenes for ordinary primes can be easily extended for Gaussian primes, but we need much more memory to get large Gaussian primes. Our generation methods both make use of the following fundamental fact [1][3]. Let p be an ordinary prime integer. If $p \equiv 3 \pmod 4$, then $p+0i$ is a Gaussian prime. If $p \equiv 1 \pmod 4$, there uniquely exist a and b such that $a^2 + b^2 = p$ and $0 \leq b \leq a$, for which $a + bi$ is a Gaussian prime.

ordinary prime p	(mod 4)	Gaussian prime $z = a + bi$		
$p = 5, 13, \ldots$	$p \equiv 1$	$a^2 + b^2 = p$	$0 < b < a$	$\lvert z\rvert^2 = p$
$p = 2$	$p \equiv 2$	$a = b = 1$	$0 < b = a$	$\lvert z\rvert^2 = p$
$p = 3, 7, 11, \ldots$	$p \equiv 3$	$a = p, b = 0$	$0 = b < a$	$\lvert z\rvert = p$



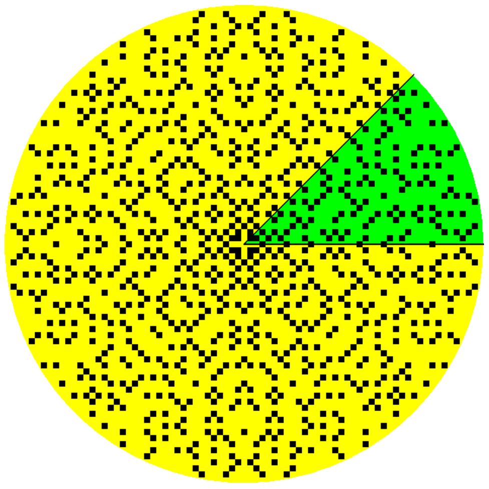


Figure 2: Gaussian primes with $|a + bi| \leq 39$ and the first octant $0 \leq b \leq a$

The first method is to test a Gaussian integer $a + bi$ with $0 \leq b \leq a$ for its primeness.

- If $b \neq 0$, $a^2 + b^2 \equiv 1 \pmod 4$ and $a^2 + b^2$ is an ordinary prime, then $a + bi$ is a Gaussian prime.
- If $b = 1$ and $a = 1$, then $a + bi$ is a Gaussian prime.
- If $b = 0$, $a \equiv 3 \pmod 4$ and a is an ordinary prime, then $a + bi$ is a Gaussian prime.
- Otherwise $a + bi$ is a Gaussian composite number.

This method will be used in combination with Gethner's method in our computational experiments in Section 3.1. A recent result [8] has shown that primality of an integer n can be tested in $O(\log^{12+\epsilon} n)$ time. This important finding, however, still remains to be theoretical. Randomized algorithms such as Miller-Rabin test [9] are more suitable for practical use.

The second method that we use in the method proposed in Section 3.2 is to generate a Gaussian prime from an ordinary prime p.

- If $p \equiv 1 \pmod 4$, we decompose p into a sum of two squares, $p = a^2 + b^2$, to get $a + bi$.
- If $p \equiv 2 \pmod 4$, i.e., $p = 2$, we get $1 + i$.
- If $p \equiv 3 \pmod 4$, we get $p + 0i$.

The decomposition in the first case can be done as follows efficiently [3]. Let x be an integer such that $x^2 \equiv -1 \pmod p$. Such x can be computed on the basis of the formula $x \equiv r^{(p-1)/4} \pmod p$, where r is the minimal quadratic non-residue modulo p. Apply the Euclidean algorithm to the pair of p and x, and let a to be the first in the residue sequence that is smaller than $\sqrt{p}$. It is known that $b = \sqrt{p - a^2}$ is an integer, and hence this pair (a, b) gives the decomposition $p = a^2 + b^2$.

The prime counting function $\pi(x)$ denotes the number of ordinary primes less than or equal to x. The prime number theorem (see. e.g., [6]) says

$$\pi(x) \sim \mathrm{Li}(x) = \int_2^x \frac{dt}{\log t} \sim \frac{x}{\log x}$$

for large x. We define $\pi_1(x)$ to be the number of primes p with $p \leq x$ and $p \equiv 1 \pmod 4$, and $\pi_3(x)$ to be the number of primes p with $p \leq x$ and $p \equiv 3 \pmod 4$. According to the Chebotarev density theorem [7], we have $\pi_1(x) \sim \pi_3(x) \sim \pi(x)/2$ for large x. The number of Gaussian primes of absolute value less than or equal to x lying in the first octant

$$\Big|\{z \in \mathbf{C} \mid |z| \leq x, 0 \leq \arg z \leq \pi/4\}\Big|$$

can be estimated as

$$\pi_1(x^2) + 1 + \pi_3(x) \sim \frac{\pi(x^2)}{2} + \frac{\pi(x)}{2} \sim \frac{x^2}{4\log x}.$$

3 Computational Method

For the computation of the farthest point of the connected component of the origin we employed two methods, Gethner's method [1] and another method that we propose. It turned out that the proposed method runs approximately 10 times faster than Gethner's.

3.1 Gethner's Method

The fundamental approach of Gethner's method is a breath-first search on the graph G introduced in Section 1. We classify Gaussian primes according to the number of steps to reach from the origin; the Gaussian primes of level n are those that can be reached from the origin in n steps, but not fewer.

A Gaussian prime within distance k from a Gaussian prime of level n is at the level of $n-1$, n, or $n+1$. This fact allows us to retain only those Gaussian prime at level $n-1$ and n when we search for Gaussian primes of level $n+1$. Suppose that we have found all the Gaussian primes of level n or less. For each Gaussian prime of level n, we collect Gaussian integers lying within distance k from that Gaussian prime. We then form the union of those sets of Gaussian integers over all Gaussian primes of level n. The Gaussian primes in this union, except for those of level $n-1$ or n, are exactly the Gaussian primes of level $n+1$.

Table 1: Comparison of the two methods

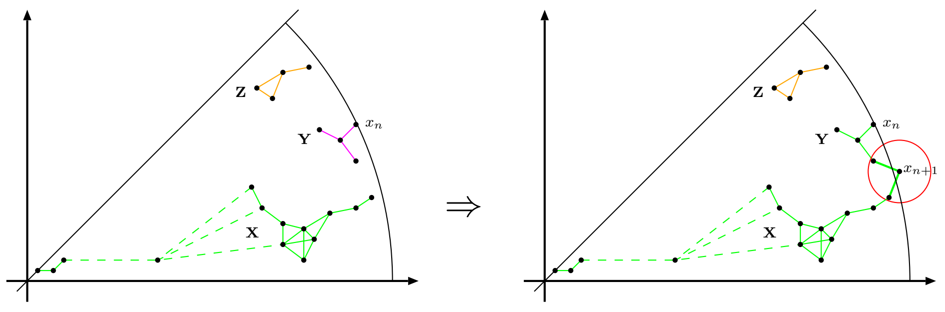
	Gethner's method	Our proposed method
Basic idea	breath-first search on graph G	sequential subgraph construction
Order of Gaussian prime generation	random	sequential
Generation of Gaussian primes	primality test (to save memory)	sieve of Eratosthenes + decomposition into sum of two squares
Generated Gaussian primes	connected component from the origin (50 $\sim$ 75% of right column)	$\{x_n \mid \|x_n\| \leq \|\xi(k)\| + k\}$
Maximum number of Gaussian primes needed to retain in memory	(750 for $k = 18$)	$\sim \dfrac{\sqrt{k}\|\xi(k)\|}{2 \log \|\xi(k)\|}$ (2600 for $k = 18$) (540000 for $k = 32$)
Number of generated Gaussian primes per second	10,000/sec	25,000–100,000/sec

We need primality test for Gaussian integers. This can be reduced to primality test for ordinary integers by the fact described in Section 2. As the sieve of Eratosthenes is quick but needs much memory to retain all the primes for the random order of primality test, we adopted Miller-Rabin test combined with Lucas-Lehmer test, which is implemented in `java.math.BigInteger.isProbablePrime()`.

3.2 Our Method

We generate Gaussian primes in the order of their norm by the method described in Section 2. In terms of the graph G introduced in Section 1, the vertices of G are numbered sequentially according to the norm of the associated Gaussian primes. As we generate Gaussian primes, we determine the connected components of the subgraph on G induced on the vertices generated so far. This can be done efficiently as follows.

Let $x_1, x_2, \ldots, x_n, x_{n+1}, \ldots$ denote the Gaussian primes, where $|x_1| < |x_2| < \cdots < |x_n| < |x_{n+1}| < \cdots$. We denote by G_n the subgraph of G induced on $\{x_1, x_2, \ldots, x_n\}$. A connected component is represented by an



G_n
Components X, Y, and Z are independent.

G_{n+1}
Components X and Y are connected by x_{n+1}.

Figure 3: Subgraphs G_n and G_{n+1}

arborescence (a directed tree) with arcs directed toward the unique root, which is the Gaussian prime with the largest norm in the component. Accordingly the connected component decomposition of G_n is represented by a family of such arborescence. We represent this family of arborescence by an array of pointers.

To add a new vertex x_{n+1}, we look for x_i with $i \leq n$ such that $|x_{n+1} - x_i| \leq k$. If no such x_i exists, we identify the singleton set $\{x_{n+1}\}$ as an isolated connected component of G_{n+1}. Otherwise, let $x_{i_1}, x_{i_2}, \ldots, x_{i_m}$ be the (exhaustive) list of such x_i, and let r_j be the root vertex of the component of G_n containing x_{i_j} for $j = 1, 2, \ldots, m$. We then add a pointer from r_j to x_{n+1} for $j = 1, 2, \ldots, m$. Note that this amounts to merging the components of $r_1, r_2, \ldots, r_m$ in G_n into a single component in G_{n+1} with root x_{n+1}.

In practical implementation, we do not need to keep the entire array of pointers. In fact, we may discard the pointers for x_i if $|x_i| < |x_n| - k$. This means that we maintain the Gaussian primes contained in the band region

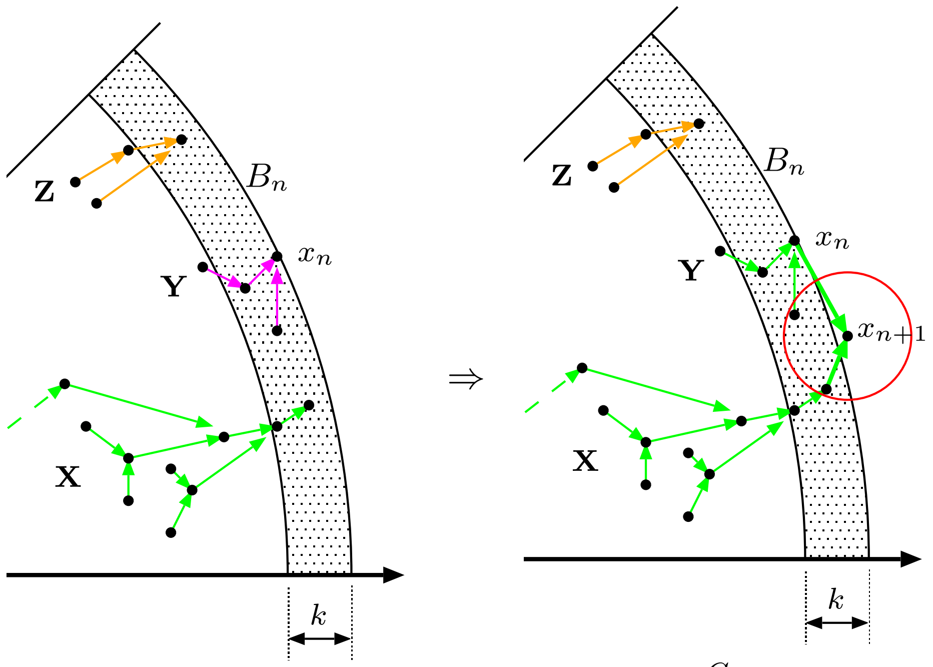
$$B_n = \{z \in \mathbf{C} \mid |x_n| - k \leq |z| \leq |x_n|, 0 \leq \arg z \leq \pi/4\}.$$

See the illustration in Fig.4. The number of Gaussian primes in the band region B_n is approximated by

$$|B_n| \sim \pi_1(|x_n|^2) - \pi_1((|x_n| - k)^2) \sim \frac{k|x_n|}{2\log|x_n|}.$$

The search process terminates if the root vertex, say, r_0 representing the component of the origin lies outside the band region, i.e., if $|r_0| < |x_n| - k$. Then we identify r_0 as $\xi(k)$, the farthest point reached from the origin for step size k.

An upper bound on $|\xi(k)|$ can be obtained by a slight variant of the above procedure. We choose a Gaussian prime y and generate all Gaussian



G_n
The root of component Y is x_n.

G_{n+1}
The root of component X and Y becomes x_{n+1}.

Figure 4: Arborescences representing G_n and G_{n+1} and the band regions B_n

primes lying in the region $\{z \in \mathbf{C} \mid |y| - k \leq |z| \leq |y|, 0 \leq \arg z \leq \pi/4\}$. By assuming (fictitiously) that all the Gaussian primes within this region are connected to the origin we run our search procedure. Note that this assumption is tantamount to assuming that all the Gaussian primes with the absolute value $\leq |y|$ are connected to the origin. If $|y|$ is chosen to be large enough, the search procedure is likely to terminate before long, providing an upper bound on $|\xi(k)|$.

4 Computational Result

The main discovery of our computational experiment is that one can reach $2106442 + 1879505i$ by step size $k = \sqrt{32}$. The point is at the distance of 2823054.542 from the origin. The number of Gaussian primes we generated is 138994584350. The total size of the component is 103711268594, which amounts to 75% of the generated Gaussian primes. The maximum number of Gaussian primes lying in the band region illustrated in Fig.4, which we need to retain in memory, is about 540000. (This number is very close to our estimate $k|\xi(k)|/2\log|\xi(k)| = 537577$.) We need 16Mbytes for them. The computational time we need is about 80 hours by parallel computing with 38 CPUs. It would take 70 days by one computer.

Another discovery is that one cannot walk to infinity with step size $k = \sqrt{36}$. A bound of 80015782 on the distance reachable from the origin has been found.

Out computational environment consists of 19 machines connected with Gigabit Ethernet (Intel Pentium III 1.4GHz x 2 (SMP), 1GByte memory, Red Hat Linux 7.1 (kernel 2.4.18), Java J2SE 1.4.2, Java HotSpot Server VM, gcc 2.96).

Component of origin using distance k	Farthest point reached $\xi(k)$	Farthest distance reached $\lvert\xi(k)\rvert$	Total size of the component$*$	Computation time (1CPU/38CPUs)
$\sqrt{1}$	$2+i$	2.236	2	
$\sqrt{2}$	$11+4i$	11.705	14	
$\sqrt{4}$	$42+17i$	45.310	92	
$\sqrt{8}$	$84+41i$	93.472	380	
$\sqrt{10}$	$976+311i$	1024.352	31221	
$\sqrt{16}$	$3297+2780i$	4312.610	347638	5sec/ —
$\sqrt{18}$	$8174+6981i$	10749.355	2386129	25sec/ —
$\sqrt{20}$	$120510+57857i$	133679.065	273791623	4hour/ 8min
$\sqrt{26}$	$943460+376039i$	1015638.765	14542615005	10day/11hour
$\sqrt{32}$	$2106442+1879505i$	2823054.542	103711268594	— /80hour
$\sqrt{34}$	—	< 24289452	Finite	— /130hour
$\sqrt{36}$	—	< 80015782	Finite	— /26hour

$*$ the number of points distinct from the origin

The following computational techniques turned out to be useful.

- As is described in Section 3.2 we create an arc from r_j to x_{n+1}. In addition, for all y lying on the path from x_j to r_j, we also create arcs from y to x_{n+1} and delete arc y to its successor on the path.
- We retain Gaussian primes in B_n with pointers in a hash table. We adopted the imaginary part b as the key for a Gaussian prime $a+bi$. As is easily seen from Fig.5, this is most advantageous among three natural candidates for the key,
  (a) real part a
  (b) imaginary part b
  (c) argument $\arctan(b/a)$.

If the real part is adopted, we need to check many extra points lying outside the circle centered at x_{n+1} with radius k. If the argument is used, the circle is enclosed more tightly, but the computational cost for arctan is high.

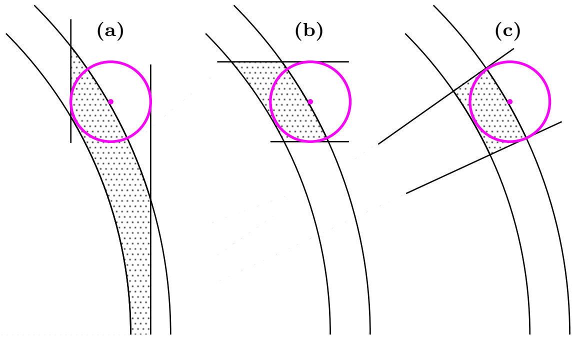


Figure 5: Choice of the key for hash table

- The computational process is divided into two: one is to generate Gaussian primes in the order of their absolute value, and the other is to make subgraphs G_n to check for the connectivity among them. The computational cost of the former is prohibitively higher than that of the latter.

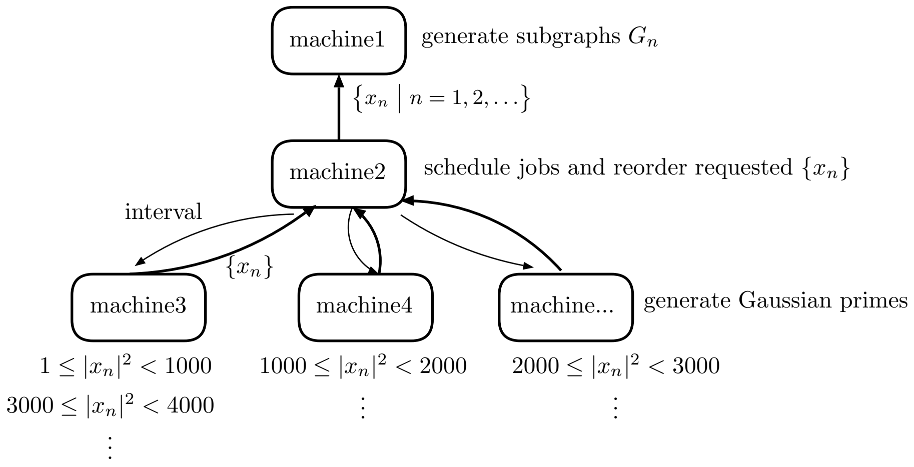

machine1
generate subgraphs G_n
$\{x_n \mid n = 1, 2, \ldots\}$
machine2
schedule jobs and reorder requested $\{x_n\}$
interval
$\{x_n\}$
machine3
machine4
machine...
generate Gaussian primes
$1 \le |x_n|^2 < 1000$
$3000 \le |x_n|^2 < 4000$
$1000 \le |x_n|^2 < 2000$
$2000 \le |x_n|^2 < 3000$

Figure 6: Task assignment for parallel computing

We can parallelize the generation of Gaussian primes easily because there is no interaction from the connectivity testing. See Fig.6 in which machine1 tests for the connectivity, whereas machine2 assigns the task of generating Gaussian primes to the remaining machines. Generating Gaussian primes is assigned to many different CPUs according to their absolute values. We could generate Gaussian primes approximately 20 times faster. The bottleneck of generating Gaussian primes is thus resolved. It seems difficult to parallelize the connectivity testing, which is now the bottleneck of the process.

5 Estimating the farthest distance reachable from the origin

On the basis of percolation theory Vardi [4] constructed a model of Gaussian primes and conjectured that

$$k \sim \sqrt{2\pi\lambda_c \log|\xi(k)|},$$

where $\lambda_c \approx 0.35$ is a constant in continuum percolation and $\xi(k)$ is the farthest point reachable from the origin with step size k. This relation can be rewritten as

$$\log|\xi(k)| \sim \frac{k^2}{2\pi\lambda_c},$$

which is shown in Fig.7. Although the observed data are approximated fairly well by this estimate, they lie consistently above the estimate and form a zigzag line. An improved estimate is proposed below.

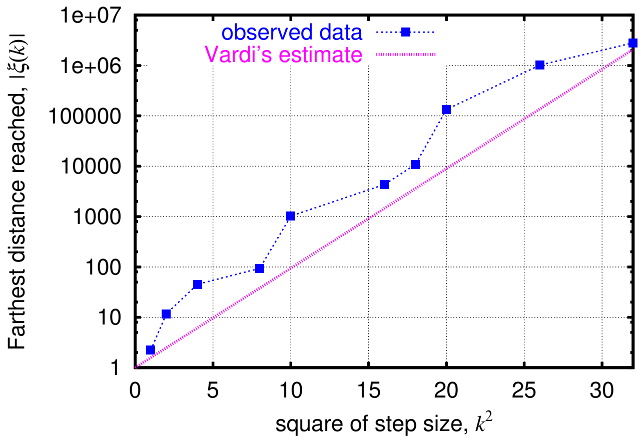
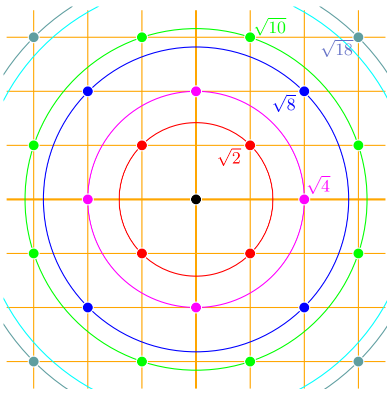


Figure 7: Vardi's estimate

When a Gaussian integer $a+bi$ is a Gaussian prime excepting $\pm1\pm i$, then (a,b) is either (odd, even) or (even, odd). The difference (x,y) between two Gaussian primes is either (even, even) or (odd, odd). For a given step size

k, there are only finitely many pairs (x, y) such that $x^2 + y^2 \leq k^2$ and $x \equiv y \pmod 2$. They are possible edges in graph G. We denote the number of such pairs (x, y) with $x \geq 0$ and $y > 0$ for step size k by $p(k)$.



k	$p(k)$	possible edges
$\sqrt{1}$	1	(0,1)
$\sqrt{2}$	2	(1,1)
$\sqrt{4}$	3	(0,2)
$\sqrt{8}$	4	(2,2)
$\sqrt{10}$	6	(1,3) (3,1)
$\sqrt{16}$	7	(0,4)
$\sqrt{18}$	8	(3,3)
$\vdots$	$\vdots$	$\vdots$
$\sqrt{40}$	18	(2,6) (6,2)
$\sqrt{50}$	21	(1,7) (5,5) (7,1)

Up to the fourfold symmetry

Figure 8: Possible edges corresponding to step size k

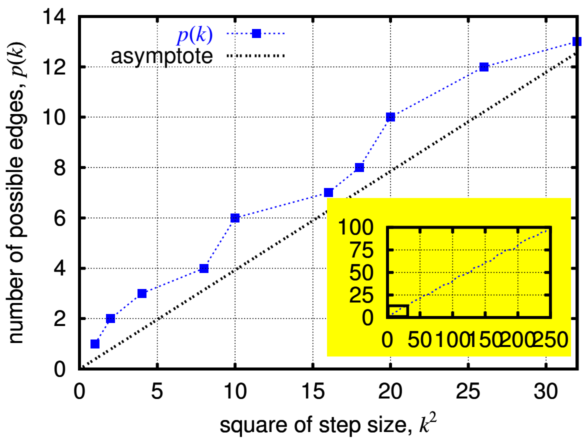


Figure 9: Relation between the number of possible edges $p(k)$ and square of step size k

Figure 9 shows the relation between $p(k)$ and k^2. Globally

$$p(k) \sim \pi k^2/8,$$

but we can observe local zigzags.

It is remarkable that the zig-zag lines in Fig.7 and Fig.9 look similar. Combination of Fig.7 and Fig.9 results in Fig.10, which shows the relationship between $|\xi(k)|$ and $p(k)$. As is evident in Fig.10, $|\xi(k)|$ and $p(k)$ are almost proportional to each other. This demonstrates an improvement upon Vardi's original estimate.

Least square method using the 10 data with $1 \leq k \leq 32$ yields

$$\log|\xi(k)| \sim 1.160p(k).$$

On the other hand, Vardi's estimate, together with $p(k) \sim \pi k^2/8$, yields

$$\log|\xi(k)| \sim \frac{k^2}{2\pi\lambda_c} \sim \frac{1}{2\pi\lambda_c}\frac{8p(k)}{\pi} = \frac{4p(k)}{\pi^2\lambda_c} = 1.158p(k).$$

These two estimates consider very well with each other, as is demonstrated also in Fig.11.

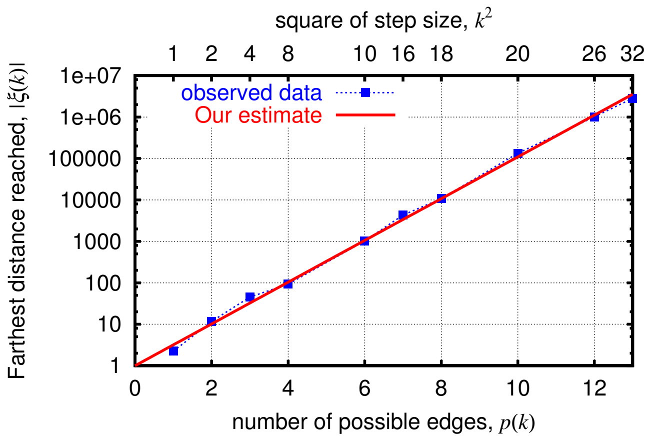


Figure 10: Our estimate

Whereas Vardi's estimate captured the essential nature of the Gaussian moat problem on the basis of percolation theory that puts emphasis on randomness and continuity. Our contribution here is a modification, or a calibration, of Vardi's estimate with considerations of discreteness. The modified estimate fits the observed data pretty well, and is consistent with Vardi's in the global scale.

Acknowledgements

I would like to thank Kazuo Murota and Shiro Matuura for their helpful advice. I would like to thank Kosuke Yabuki who brought this interesting problem to me. I would like to thank Shuji Kijima for his useful idea. I also would like to acknowledge the computational environment supported by the Special Coordination Fund for Promoting Science and Technology, Ministry of Education, Culture, Sport, Science and Technology.

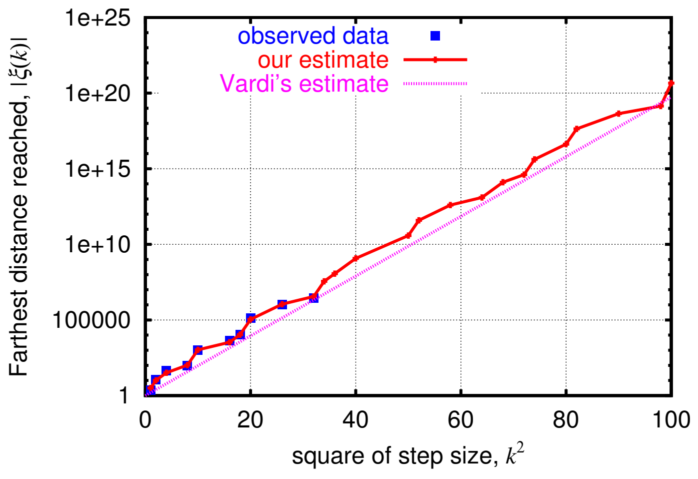


Figure 11: Two estimates

References

[1] E. Gethner, S. Wagon, and B. Wick, "A Stroll Through the Gaussian Primes," *American Mathematical Monthly* **105**:4 (1998), 327–337.

[2] E. Gethner and H. M. Stark, "Periodic Gaussian Moats," *Experimental Mathematics* **6**:4 (1997), 289–292.

[3] S. Wagon, *Mathematica in Action*, Springer-Verlag New York, 1999.

[4] I. Vardi, "Prime Percolation," *Experimental Mathematics* **7**:3 (1998), 275–288.

[5] J. H. Jordan and J. R. Rabung, "A Conjecture of Paul Erdös Concerning Gaussian Primes," *Mathematics of Computation* **24** (1970), 221–223.

[6] P. Ribenboim, *The Little Book of Big Primes*, Springer-Verlag, New York, 1991.

[7] G.H. Hardy, E.M. Wright, *An Introduction to the Theory of Numbers 5th Edition*, Oxford University Press, 1979.

[8] M. Agrawal, N. Kayal and N. Saxena, "PRIMES is in P," Preprint (2002), `http://www.cse.iitk.ac.in/news/primality.html`

[9] M. O. Rabin, "Probabilistic algorithm for testing primality," *Journal of Number Theory* **12** (1980), 128–138.

[10] H. Okumura et al., *Dictionary of Algorithms in Java*, Gijutsu-Hyohron Co., Ltd., 2003.