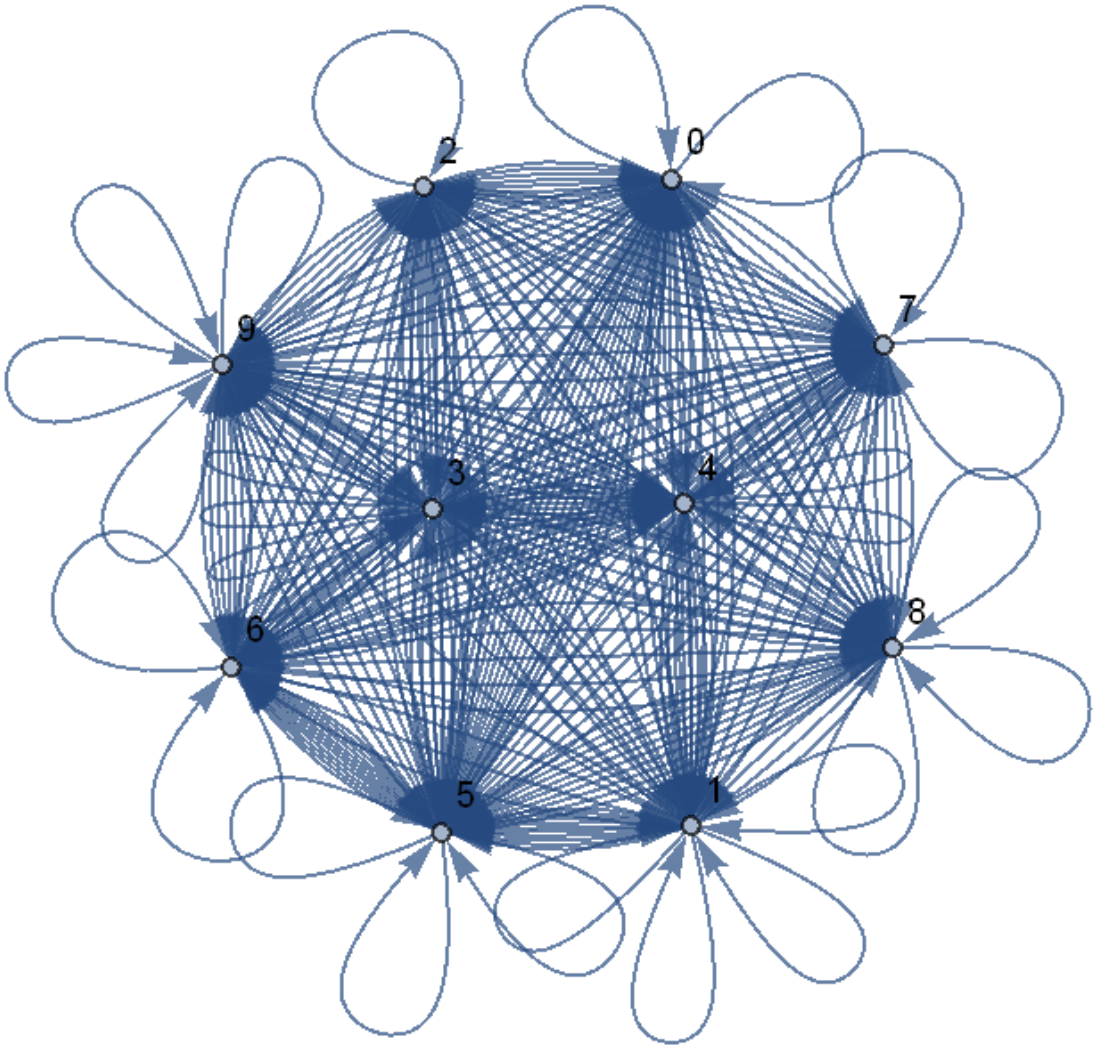
GIFT – **G**raphical **I**nteger **F**actorization **T**echnique



1

João Carlos Leandro da Silva

joaodasilva@rainbowofprimes.com

Giuseppe de Nicolellis

denic@mclink.it

The first name is that of the main author because he has invented this method entirely on his own. He has also designed and coded all related algorithms using *Mathematica* from Wolfram Research Inc. The second author has efficiently implemented some of these algorithms and has developed additional libraries thus creating a custom program that runs in polynomial time with respect to the given number. Please note that such program only concerns the pre-computation phase of this technique. The final phase consists of setting up a search experiment to find a multiple of any prime factor of the given modulus [1]. For obvious reasons, this paper will not contain any technical detail regarding the custom program. At this point in our ongoing research, we are currently interested in finding a partner (individual, company, institution or government) that has the required computational resources in order to test if the present technique can factor an integer like RSA-1024 bits in a reasonable amount of time. The aim of this paper is to present the subject at hand as clear and concise as possible in order to reach a wide audience.

Cover illustration: Graphical representation of the famous 309 decimal digit composite number (RSA-1024 bits) with no known prime factor to the best of public knowledge.

Both authors make no warranty of any kind, expressed or implied, with regard to programs or documentations contained in this paper. The authors shall not be liable in any event for incidental or consequential damages in connection with, or arising out of, the furnishing, performance, or use of these programs or documentations.

Furthermore, if anyone ever discovers an integer factorization method that is polynomial with respect to the input then such is a cyberweapon [2]. Hence, if someone reads this paper and verifies on their own that the given technique or a variation of it does indeed work then the present authors are not responsible in anyway or in any circumstance for whatever results from such actions since we strongly support the ban of all cyberweapons.

1. INTRODUCTION

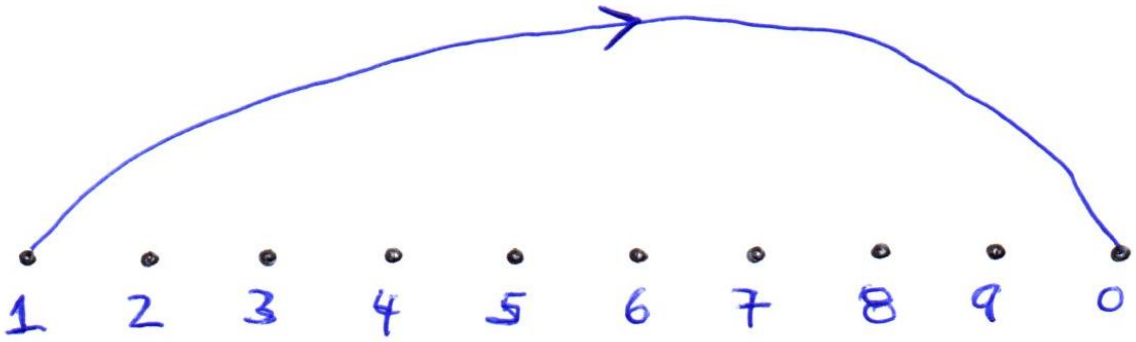
Prime numbers are the building blocks of arithmetic. The reason is because every integer greater than 1 is either a prime or a finite product of prime numbers and such product is unique. Thus, being able to decompose or factor an integer is of fundamental importance to number theory. Anyhow, it seems that efficient integer factorization is a truly difficult task. Several methods (ECM, QS and NFS) are in use today but they all fail to scale-up to numbers such as RSA-1024 bits and beyond. 3D integer factorization is an alternative since it attempts to generate a multiple of any prime factor and uses a search experiment to find such prime number. Unfortunately, this new scheme requires further research. In the present work, GIFT is the natural aftermath in our line of reasoning because it creates a bridge between number theory and graph theory [3]. In this new approach, every integer is a finite set of decimal digits and such set corresponds to a graph. We hope that such link reveals the intrinsic relationship that may exist between the digits of an integer and its prime factors.

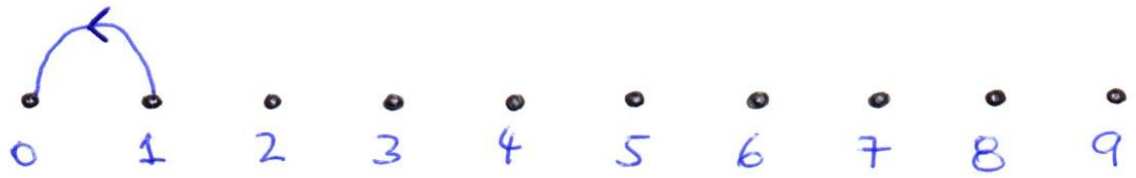
2. MAIN CONCEPTS AND TERMINOLOGY

GIFT stands for graphical integer factorization technique. In essence, we transform any integer into its respective two-dimensional graph. Any such graph has only four types of components: NODES, STICKS, SELF-LOOPS and LOOPS. Each of the decimal digits (1, 2, 3, 4, 5, 6, 7, 8, 9, 0) are represented by a black dot or **NODE** as shown below:

In fact, the position of these NODES in the above space is completely irrelevant since there is no distinction between (1, 2, 3, 4, 5, 6, 7, 8, 9, 0) and (0, 1, 2, 3, 4, 5, 6, 7, 8, 9) or any other possible combination among the ten different decimal digits. Such seems trivial because all we have are NODES but the same applies when our graph is more intricate, that is, it

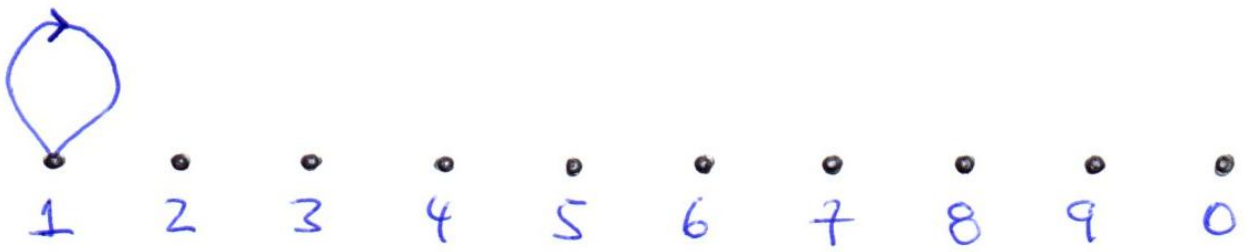
contains other components. For example, consider the integer 10 and its equivalent graphs:



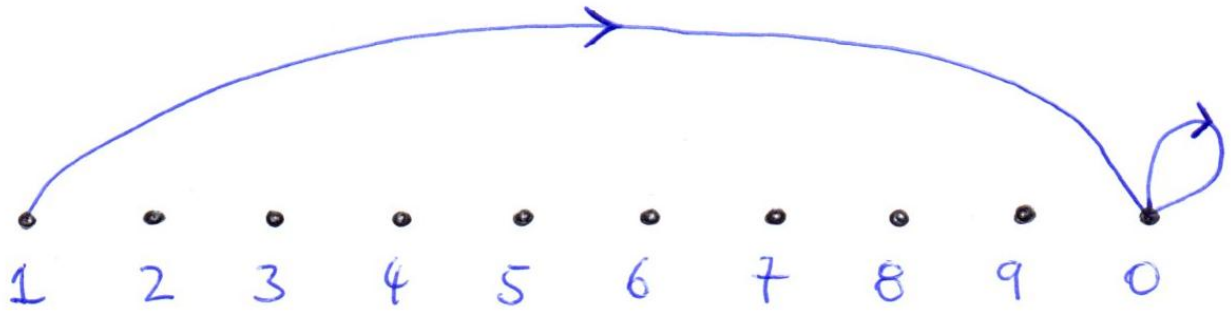


The top graph is a large blue arc or **STICK** that starts at NODE 1 and terminates at NODE 0. Note that we are dealing with a directed graph [4] since in the above STICK there is an arrow that points from the origin (NODE 1) to the destination (NODE 0). The bottom graph is a small blue arc or STICK that starts at NODE 1 and ends at NODE 0. It is important to realize that both the lengths of the arcs and the size of the spaces between the nodes has no significance. All that matters is the graph itself.

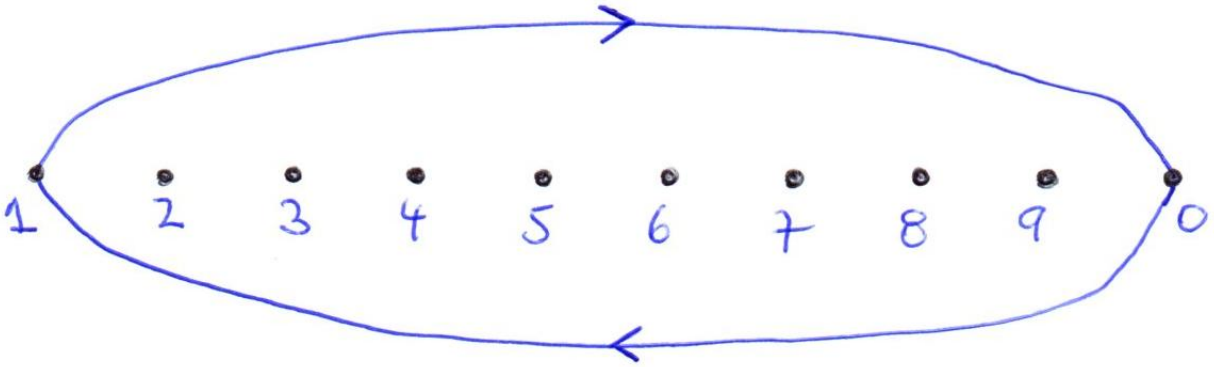
Now, take into account the integer 11. It is represented by a blue circle or **SELF-LOOP** because it starts at NODE 1 and ends at NODE 1 as shown:



Since the origin and destination are the same, the user choice with respect to the direction of the arrow is irrelevant. It is straightforward to notice that all integers from 12 until 99 will result in a STICK (for different digits such as 23, 34, 45 and so on) or in a SELF-LOOP (for identical digits like 22, 33, 44 and so on). In vivid contrast, the integer 100 is equivalent to a STICK and a SELF-LOOP as shown below:



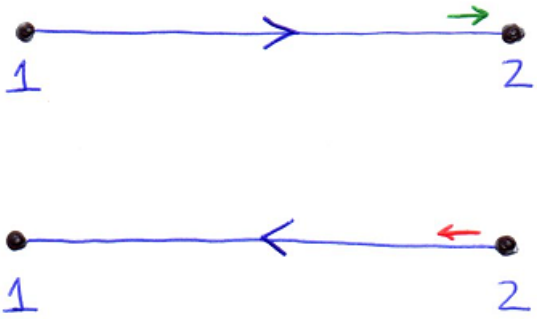
Next, the prime 101 is comparable to a large blue circle or **LOOP** because it starts at NODE 1 goes to NODE 0 and returns to NODE 1 as shown:



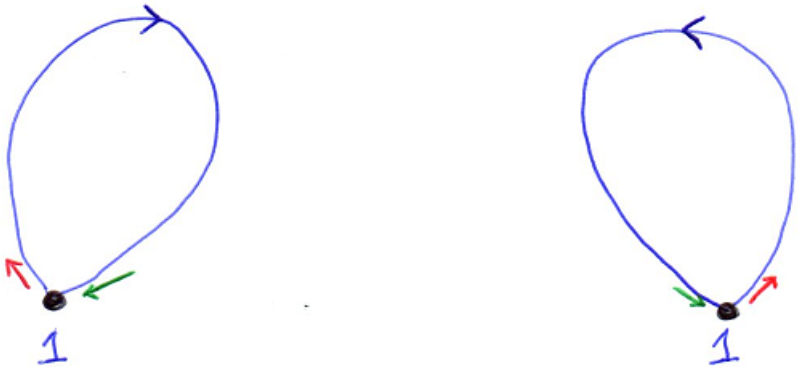
If instead of (1, 2, 3, 4, 5, 6, 7, 8, 9, 0) we had (0, 1, 2, 3, 4, 5, 6, 7, 8, 9) the resulting graph would still be the same, that is, a LOOP. Once again, the size of the circle has no significance. All that matters is the graph itself.

According to GIFT, any integer is equivalent to a graph with at most four different components. The word “topology” and all of its derivatives are absent here because we feel that it is not appropriate in this new context. Instead, two graphs are equal only if their components match in type and number. For example, the graphs of 12 and 23 are equal because they both

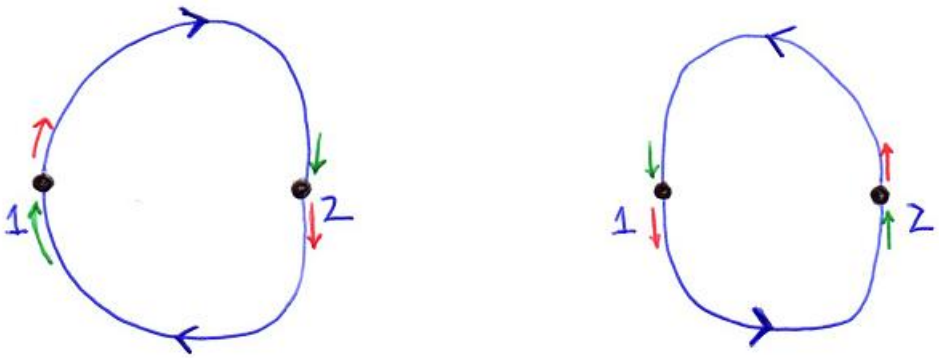
consist of a single STICK. We analyze in detail each of the components of GIFT. There are two possibilities for the **STICK**. In the first case, the arrow is entering NODE 2, here written in green as IN and in the second case, the arrow is exiting NODE 2 here written in red as OUT as shown:



There is only one possibility for the **SELF-LOOP**. Independent of the direction of the circle, there is just one arrow entering (IN) the respective node and another arrow exiting (OUT) that same node, as shown below:



There is only one possibility for the **LOOP**. Independent of the direction of the circle, there is just one arrow entering (IN) each of the two nodes and one arrow exiting (OUT) each of the two nodes, as shown below:

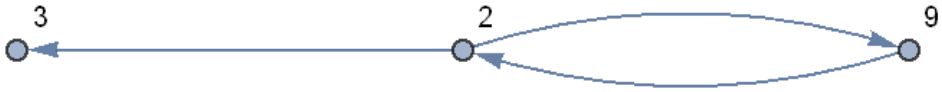


3. A FIRST EXAMPLE

In 3D integer factorization, we introduced the new idea of factoring a semiprime by generating a multiple of any prime factor and doing the search experiment. On the other hand, GIFT generates a directed graph and performs all the necessary mathematical operations. What results from all this are potential multiples of any prime factor but here such multiples are the outcome of an intrinsic mathematical process and not some random choice by a human user. An example will clarify the matter. 2923 is the modulus and after running the following five lines of *Mathematica* code:

```
ClearAll["Global`*"]
modulus=2923
list=IntegerDigits[modulus]
sequence=Drop[Thread[list-> RotateLeft[list]],-1]
directedgraph=Graph[sequence,DirectedEdges->True,VertexLabels->"Name"]
```

We obtain the directed graph shown below:

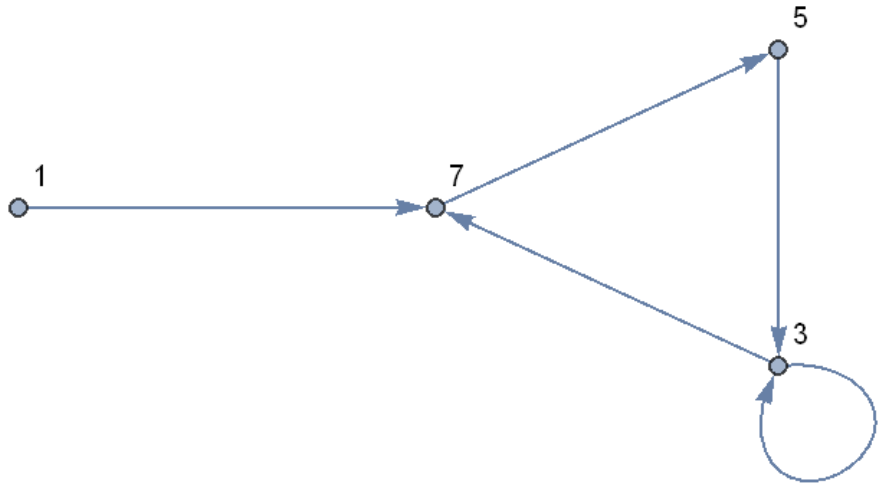


It is obvious that the above three blue colored dots are the only NODES. There is a single STICK with the numerical value of 23 and a single LOOP with the corresponding value of 292. How do we factor 2923? With a single step, that is, performing GCD[2923, 292 × 23 – 1] = 79 where GCD stands for the greatest common divisor. Once the graph is generated, GIFT provides us with a numerical value for every component. In our case, 6715 is the correct multiple because 6715 = 85 × 79. The general mathematical relation is $LOOP \times STICK \pm k$ where k represents a constant that must be much smaller than either 292 or 23. In fact, $k = 1$ and since the computer also finds the mathematical relation, the human user does not need to intervene in any manner. Such is the main advantage of GIFT.

4. FORMULAS, EQUATION AND UNKNOWNS

The previous modulus has only four decimal digits but as the moduli increase their mathematical relations become more intricate and therefore additional terminology must be introduced. We will define the meaning of each new term. Accordingly, **FORMULA** represents any mathematical operation between two or more components of GIFT. For example, in our first example where 292 × 23 or LOOP × STICK such is a FORMULA. It is clear that, in this specific case, multiplication is the operation. Note that in this new context FORMULA is the building block of the mathematical relation. So, if there is no FORMULA, we cannot factor the modulus. Yet, it is easy to show that all integers equal or greater to one hundred must have a FORMULA but that does not imply the factorization of every such integer.

Assume 175337 as our modulus. Using the previous lines of *Mathematica* code, we obtain the directed graph below:



Observe the type and number of components present above. First, there are four NODES. Second, one STICK. Third, two LOOPS. Fourth, one SELF-LOOP. There are two FORMULAS. One is LOOP + SELF-LOOP and the other is LOOP + STICK. Therefore, EQUATION consists in their product. Note that **EQUATION** simply represents the final mathematical relation.

GIFT provides us with all the necessary numerical values. The STICK is 17 while the LOOPS are 7537 and 3753. The SELF-LOOP is $3^2 = 9$. Thus, EQUATION is $(3753 + 3^2) \times (7537 + 17) \pm k$ so our custom program runs $GCD[175337, (3753 + 3^2) \times (7537 + 17) \pm k]$ and gets $GCD[175337, (3753 + 3^2) \times (7537 + 17) - 4] = 271$ for $k = 4$. At this point, we explain why the SELF-LOOP associated with NODE 3 is three raised to the power of two. The crucial thing to realize is that GIFT allows anyone to do two tasks. First, transform any integer into a directed graph. Second, this method attributes to every component of the given graph a specific numerical value. Please note that these two processes are not always reversible. For example, the integer 11 translates into a SELF-LOOP but if we use GIFT, the equivalent numerical value is not 11 but $1^2 = 1$. Why? There are ten nodes and any node may contain a SELF-LOOP. If NODE 0 has a SELF-LOOP then the numerical value cannot be $0^2 = 0$ for that is zero. The logical alternative for NODE 0 is $10^2 = 100$ and such explains why any SELF-LOOP associated with some NODE x must be x^2 where x stands for the number of the relevant node.

A remark concerning UNKNOWNS. The previous mathematical relations seen until now only involve one unknown, that is, the constant k because without it, we are unable to generate a multiple of any of the prime factors. In spite of that, depending on the modulus and the respective graph, the number and types of UNKNOWNS can be greater than one. Large integers like RSA-1024 bits (look at the graph on the first page of this paper) have many decimal digits and the probability that these digits repeat and consecutively duplicate is high. In fact, the directed graphs of such integers do not possess any STICK. They only contain many LOOPS and multiple SELF-LOOPS. Semiprimes like RSA-1024 bits and beyond will have two UNKNOWNS. One of these will be associated with NODE 0 and the other with a parameter of the basic algorithm of a given computer within the search experiment which is related to the constant k since the final mathematical relation has always the structure: $EQUATION \pm k$.

5. ALGORITHM AND RULES

We will adopt “graphical configuration” to define the form of the graph. Since GIFT has four types of components, the total number of different graphical configurations is eight assuming that we consider both singular and plural forms as one. The first is NODE or any single decimal digit such as 0 or 9. The second is a combination of NODES and STICKS or any integer bigger than 9 without any repeating decimal digit. Examples of such are 10 or 1234567890. The third is a merger of NODES and LOOPS and the general format is Xabc...ghiX where X represents any single decimal digit except zero and the digits in between (abc…ghi) are all different. A straightforward example is 12345678901 for a single loop and adding another digit 1 somewhere in the middle, we get two loops and so on. The fourth is an aggregate of NODES and SELF-LOOPS and the general format is X…X where X represents any single decimal digit except zero. Two possible examples are 11 and 222. The fifth is a mix of NODES, STICKS and LOOPS. The general format is Xabc...fghXi where X represents any single decimal digit except zero and the digits in between (abc…fgh) are all different. An example is 12345678910 for a single loop and adding another digit 1 somewhere in the middle, we get two loops and so on. The sixth is a union of NODES, STICKS and SELF-LOOPS and the general format is X…Xbc where X represents any single decimal digit except zero. Two possible examples are 1123 and 44456. The seventh is a fusion of NODES, LOOPS and SELF-LOOPS. Such form has no general format and RSA-1024 bits is an example. The eighth is a unification of all four types and no general format exists. An example is 1476602513 which consists of eight NODES, one STICK, one LOOP and one SELF-LOOP.

GIFT has rules that identify the numerical values for the four types of components. Other rules are necessary to build both the FORMULAS and the EQUATION. Unfortunately, there are many exceptions to these rules since all graph configurations involving STICKS (except NODES and STICKS) have a tricky form. Thus, we focus our attention on those graphs such as RSA-1024 bits that deal with NODES, LOOPS and SELF-LOOPS.

Algorithm

STEP I: input **M**

STEP II: transform **M** into a directed graph

STEP III: count the total of IN's and OUT's for each **NODE**

STEP IV: if the total ≤ 2, that **NODE** has no **FORMULA**. For all **NODES** that total ≥ 3, create respective **FORMULA**

STEP V: set up all **FORMULAS**

STEP VI: create **EQUATION**

STEP VII: set up **UNKNOWNS**

STEP VIII: calculate **EQUATION**

STEP IX: **SEARCH EXPERIMENT**

STEP X: output any non-trivial factor

STEP I. The variable **M** stands for modulus where **M** $\geq$ 100.

STEP II. The modulus becomes a list of decimal digits. The first element of such list is the leftmost digit while the last element is the rightmost digit of the given modulus. Generate directed graph.

STEP III. Make a table for all existing **NODES** of the directed graph and, for each, count the number of both incoming and outgoing arrows.

STEP IV. From STEP III, check the number of FORMULAS. Every **FORMULA** starts with a "(" and ends with a ")" and the minimum number of components is two.

STEP V. Check that all **NODES**, S**TICKS**, **LOOPS** and **SELF-LOOPS** are found among the **FORMULAS** for the given graph.

STEP VI. From STEP V, create the respective EQUATION. **EQUATION** must have at least one **FORMULA** and one **UNKNOWN**.

STEP VII. Depending on the final mathematical relation, the type and number of **UNKNOWNS** is determined.

STEP VIII. All **FORMULAS** become numerical values and the only missing variable(s) is the **UNKNOWN** or **UNKNOWNS**.

STEP IX. The **SEARCH EXPERIMENT** consists of running simultaneously many computers in order to find a non-trivial factor.

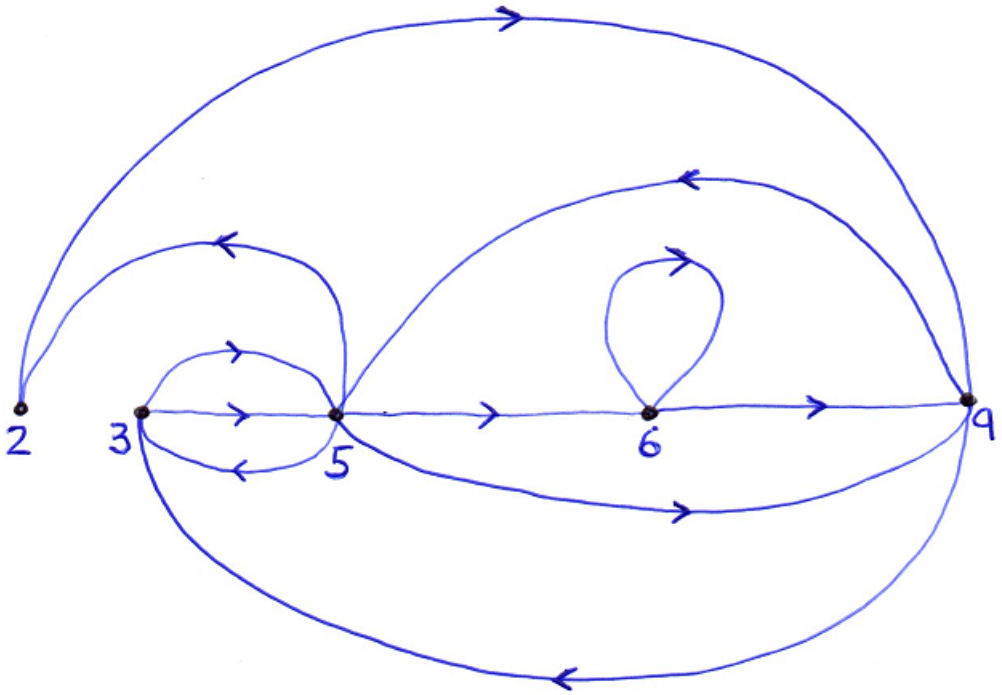
STEP X. If the search experiment is successful, we find a prime factor and the respective factorization time in seconds.

We apply our algorithm to the previous modulus (**M**) or 175337. There is no doubt that **M** > 100. Next, **M** becomes the list {1, 7, 5, 3, 3, 7} and we consider the table below:

NODE	IN	OUT	total
one	---	1	1
two	---	------	-----
three	2	2	4
four	---	------	-----
five	1	1	2
six	---	------	-----
seven	2	1	3
eight	---	------	-----
nine	---	------	-----
zero	---	------	-----

How can we know if the graphical configuration of **M** has any STICK? All you need to do is verify if there is at least one NODE that has a total value of 1. Looking above we easily identify NODE 1 as such. If you look at the graph on page 8, there is only one STICK and its numerical value is 17 as expected. Next, consider NODE 3 since the total value is 4. Looking at the graph and following the direction of the arrows starting at the given node, we get a **LOOP** with a value of 3753. Besides, it is obvious that NODE 3 has a **SELF-LOOP** with a value of $3^2 = 9$. The respective **FORMULA** is the addition of LOOP and SELF-LOOP, that is, (3753 + 9) as defined by our algorithm. NODE 5 is not taken into account because its total value (as shown above) is 2. On the other hand, we must consider NODE 7 since the total value is 3. Here, the **STICK** is 17 and the **LOOP** has a value of 7537. Thus, the corresponding **FORMULA** is (7537 + 17) and the resulting **EQUATION** is $(3753 + 3^2) \times (7537 + 17) \pm k$ where k is the only **UNKNOWN**. In this case, the **SEARCH EXPERIMENT** consists of two computers, the first runs $GCD[175337, (3753 + 3^2) \times (7537 + 17) + k]$ and the second runs $GCD[175337, (3753 + 3^2) \times (7537 + 17) - k]$. For $k = 4$, the second machine outputs the non-trivial factor of 271.

Assume 529356695359 and look at its graph below:



There are five nodes but NODE 2 has a total value of 2 so only four nodes are considered. Starting at NODE 3, the numerical values of the LOOPS are 353, 353, 35293, 35293, 3593, 3593, 35693 and 35693. There are four different loops (each repeated twice) since between NODE 3 and NODE 5 there are three direct connections (top, middle and bottom) where both top and middle have the same direction. Next, the values of the LOOPS of NODE 5 are 595, 5295, 5695, 535, 535, 5935, 5935, 52935 and 52935. At NODE 6, we find three LOOPS (6956, 69356 and 69356) and one SELF-LOOP with a numerical value of $6^2 = 36$. The LOOPS of NODE 9 are 959, 9569, 9529, 93569, 93569, 9359, 9359, 93529 and 93529. GIFT has strict rules regarding the construction of LOOPS. First, the value of any given LOOP must have at least three decimal digits like 353. Second, any LOOP associated with NODE X must start with digit X and finish with digit X such as NODE 9 and 93529. Third, in any given LOOP, the digits in between (X…X) cannot have duplicates and must be different from X. Concerning SELF-LOOPS the rule is simple. If NODE x has n SELF-LOOP(S) then its value is x^{n+1} where x stands for the relevant node and n for the number of self-loops. Yet, NODE 0 is an exception to this rule.

6. TWO BASIC ALGORITHMS

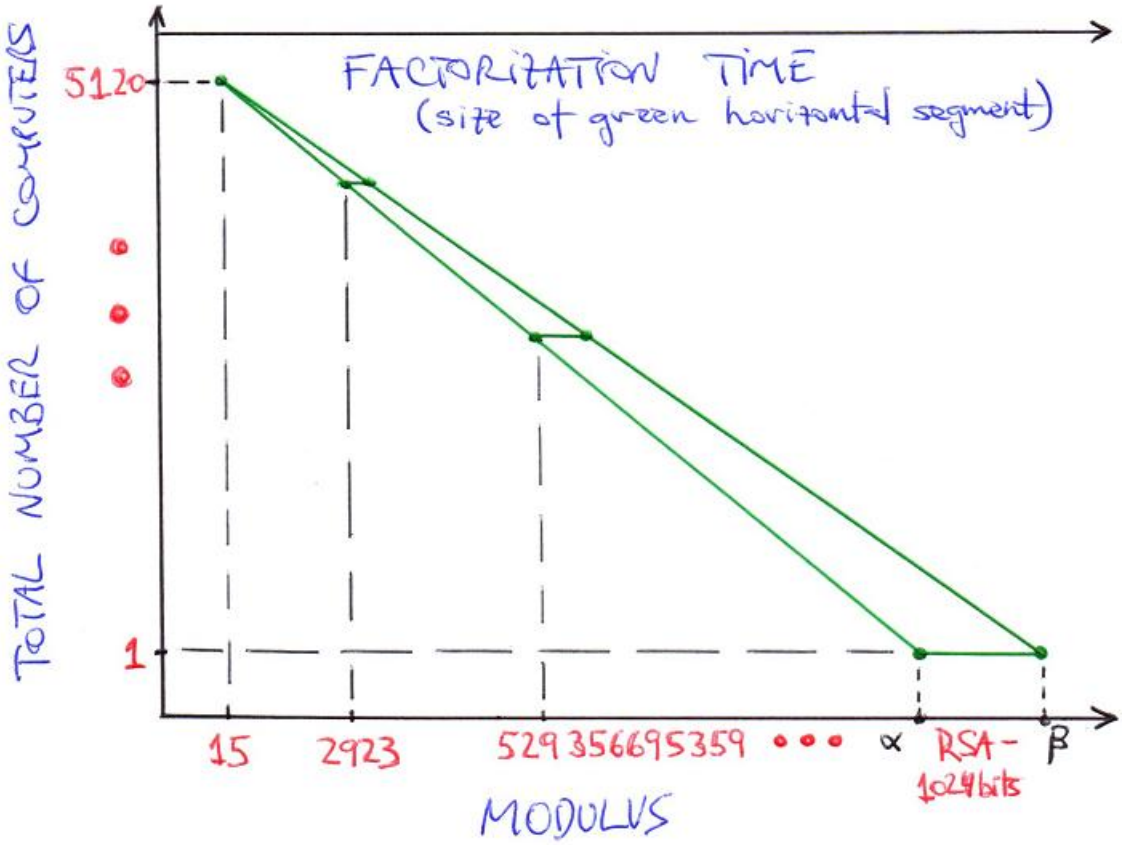
Recall that in 3D integer factorization, we introduced the concept of basic algorithm as a *Mathematica* notebook where its code attempts to generate a multiple of any prime factor with respect to the given modulus. Hence, the mathematical theory corresponds to a finite set of basic algorithms distributed among the hardware and all of these constitute the computer experiment itself. In our case, GIFT requires two *Mathematica* notebooks, consequently, we have two basic algorithms which are different in only one aspect. Remember that in 3D integer factorization, the interval goes from 10^x until $1000 + 10^x$ where $x \geq 4$. Note that in such interval, the range changes from $[10000, 11000]$ to $[100000, 101000]$ and so on. The first basic algorithm of GIFT has the same interval while in the second algorithm, the interval initiates at $y \times delta$ and ends at $(y + 1) \times delta$ where $delta = 1000$ and $y \geq 1$. Thus, in this interval, the range evolves from $[1000, 2000]$ to $[2000, 3000]$ and so on. It is clear that the range in the first algorithm grows much faster than that of the second algorithm. Yet, this algorithm checks ranges not covered by the first algorithm. Since the variable y is the constant k in $EQUATION \pm k$ and we do not know (beforehand) what numerical value will transform the final mathematical relation into a multiple of any prime factor, it is best to search for such k in two distinct ways. In fact, the first basic algorithm of GIFT looks for a possible large k while the second algorithm considers smaller numerical values. In other words, the first algorithm performs a fast search (by increments of powers of ten) while the second algorithm carries out the search sequentially, that is, it does not jump any integer. For example, if you look at the first two iterations of these algorithms, it is obvious that the first algorithm jumps (89,000 integers) from 11000 to 100000 while the second algorithm checks every integer between 1000 and 3000 inclusive. On the other hand, after only nine iterations, the first algorithm is looking for a k between 1000000000000 and 1000000001000 while the second algorithm has checked every integer from 1000 to 10000 inclusive. There is no doubt that these two basic algorithms complement each other.

7. VARIANTS AND MATRIX SIZE

Another interesting property of our new technique is that we are no longer restricted to the modulus itself. Since GIFT transforms any integer into a graph, there is the possibility to use variants or numbers that are related to the given semiprime. For example, reversing the decimal digits of the modulus (**M**) reveals some intriguing traits [5]. We define such variant by a bold capital **R** that stands for reverse. Likewise, the variant (**M** + **R**) implies the addition of the modulus and its reverse. In the same way, the variant (**M** - **R**) means the difference between the modulus and its reverse or vice-versa since the resulting integer must be positive. Finally, the last variant is (**M** × **R**) or the product of the modulus and its reverse. We must consider these five variants because it is not possible to know beforehand which respective graph will outcome into a multiple of any prime factor in the shortest factorization time.

In 3D integer factorization, we presented a method based on the generation of a multiple of any prime factor. Therefore, factoring reduces to searching efficiently for one such multiple. GIFT follows such principles with just one exception. Given any modulus, the number of computers required to carry out the respective search experiment is determined in a completely different manner. In order to calculate the exact number of machines or matrix size, we must first count how many NODES are in the previously mentioned EQUATION. In general, the matrix size is the product of three parameters: two basic algorithms, five variants and two to the power of the number of nodes minus one. For example, the graph of RSA-1024 bits includes all ten NODES. So, the $matrix\ size = 2 \times 5 \times 2^{(10-1)} = 5120$. This means that 5,120 computers are required to do the respective search experiment on the given RSA semiprime using GIFT. All of these physical machines or cores must run simultaneously since we cannot know in advance which specific computer will be the first to find a non-trivial factor. It is evident that 5120 is the maximum value for the matrix size while its minimum value is 20, that is, when the EQUATION has only two nodes associated with it.

8. SEARCH EXPERIMENT AND PLOT



We explain in detail the notion of search experiment by examining the plot above. The black vertical axes represents the total number of computers (from 1 to 5,120) that have successfully factored the given modulus. The bottom horizontal axes indicates the modulus in decimal digits. For example, 15 is the first semiprime and RSA-1024 bits is the last. The top horizontal axes illustrates the factorization time symbolized by the size of the green horizontal segment. For instance, if our search experiment uses all the machines available to break 15, most likely, five thousand one hundred and twenty computers will factor it instantly (depicted as a green dot) as shown. Due to paper size limitations, we did not draw the above plot according to scale. The sizes of the green segments corresponding to 2923, 529356695359 and RSA-1024 bits are not proportionate to their factorization times. It implies that even though the factorization time for the RSA semiprime is unknown, the green segment matching the points α and β should be much bigger. Thus, if GIFT is indeed an efficient integer factorization method then as the moduli increase so do the respective green

segments and the number of machines that successfully factor the given modulus steadily decrease but should be at least one. As shown in the previous plot, it is our aspiration that for the famous RSA-1024 bits, one computer of the search experiment is able to factor such semiprime in a reasonable amount of time. In fact, the most logical way to verify the validity of GIFT is to use 5,120 cores and test it with different moduli (recording in every run, the shortest factorization time or the first machine that outputs a prime factor) in order to see if the results from the respective search experiments coincide with the plot in the preceding page.

9. CONCLUSIONS

In this paper, we proposed a new integer factorization method that joins number theory with graph theory. GIFT changes any positive integer into a directed graph. From such graph we can easily identify NODES, STICKS, LOOPS and SELF-LOOPS. Each component has a specific numerical value and any two or more constitute a single FORMULA. Joining all formulas creates the EQUATION or the final mathematical relation. In order to solve it for any UNKNOWN, we do a search experiment or run simultaneously many computers to generate a multiple of any prime factor. If successful, we factored (possibly in a short time) the given modulus.

GIFT has several advantages over 3D integer factorization. First, it needs fewer machines to carry out the search experiment. Second, the user does not have to ponder how to generate a multiple of any prime factor because GIFT creates the final mathematical relation using its own rules. Third, to the best of our knowledge this is the first-ever integer factorization method based on visual computing [6].

We hope that this work will make other researchers think about this old but fascinating problem under a new light. Furthermore, we would be very grateful if someone could try to formalize this method so that one may understand why GIFT works at all. Last but not least, any sort of feedback (comments and/or questions) are also welcome.

REFERENCES

[1] J. C. L. da Silva, 3D integer factorization, Available from https://www.rainbowofprimes.com

[2] J. C. L. da Silva, The definition of a cyberweapon, Available from https://www.rainbowofprimes.com

[3] G. Chartrand and P. Zhang, *A first course in graph theory*, Dover Publications, New York, 2012.

[4] N. Deo, *Graph theory with applications to engineering and computer science*, PHI Learning Private, New Delhi, 2017.

[5] J. C. L. da Silva, *Factoring Semiprimes and Possible Implications for RSA*, In Proceedings of the 26th IEEE Convention in Israel, Eliat, Israel, 17-20 November 2010; pp. 182-183.

[6] S. S. Wagstaff, Jr., *The Joy of Factoring*, Student Mathematical Library, volume 68, American Mathematical Society, Providence, 2013.

