Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 62 (2015) 603 – 610

Procedia Computer Science

The 2015 International Conference on Soft Computing and Software Engineering (SCSE 2015)

A Study on the Limitations of Evolutionary Computation and other Bio-inspired Approaches for Integer Factorization

Mohit Mishra[a], Vaibhav Gupta[a], Utkarsh Chaturvedi[a], K. K. Shukla[a], R. V. Yampolskiy[b]

[a]*Dept. of Computer Science and Engineering, Indian Institute of Technology (BHU) Varanasi, Varanasi 221005, India.*
[b]*Dept. of Computer Engineering and Computer Science, University of Louisville, KY USA.*

Abstract

Integer Factorization is a vital number theoretic problem frequently finding application in public-key cryptography like RSA encryption systems, and other areas like Fourier transform algorithm. The problem is computationally intractable because it is a one-way mathematical function. Due to its computational infeasibility, it is extremely hard to find the prime factors of a semi prime number generated from two randomly chosen similar sized prime numbers. There has been a recently growing interest in the community with regards to evolutionary computation and other alternative approaches to solving this problem as an optimization task. However, the results still seem to be very rudimentary in nature and there's much work to be done. This paper emphasizes on such approaches and presents a critic study in details. The paper puts forth criticism and ideas in this aspect.

Peer-review under responsibility of organizing committee of The 2015 International Conference on Soft Computing and Software Engineering (SCSE 2015)

Keywords: Evolutionary Computation; Swarm Intelligence; Integer Factorization.

1. Introduction

The task of integer factorization has preoccupied the research community for years. It is assumed to be a difficult task owing to its computational infeasibility in factoring large semi-prime numbers. It has suitably been applied to design cryptosystems like RSA algorithm [1], Rabin cryptosystem [2] and fast Fourier transform [3]. Of these, RSA encryption systems is most well-known and frequently used in public key cryptography. It finds application in e-commerce and digital communications.

Integer factorization can be defined as the decomposition of a composite number into its non-trivial divisors, specifically prime factors in which case we can call the problem as the problem of prime factorization. The most difficult task in this regards is to factor out a semi-prime number. A semi-prime number is a composite number formed

Peer-review under responsibility of organizing committee of The 2015 International Conference on Soft Computing and Software Engineering (SCSE 2015)
doi:10.1016/j.procs.2015.08.553

by the product of two prime numbers. The task of factoring a semi-prime formed by the product of two large randomly chosen and similar sized prime numbers is extremely difficult. For such problems, no efficient polynomial-time algorithm exists for non-quantum computers. However, it must be noted that Shor [4] introduced a quantum-computer based algorithm that solves the problem in polynomial time.

The problem of integer factorization is a one-way mathematical function [5]. A one-way mathematical function is essentially a mathematical function which is easy to compute in one direction, but difficult to reverse. For example, let us suppose we are given two randomly chosen prime numbers, say p and q, we can easily compute their product, say N (=pq). However, given only N, it is very difficult to compute p and q. This becomes an arduous task because it is computationally infeasible to do so in polynomial time on a non-quantum computer.

A lot of attempts have been made in factoring very large semi-prime numbers, e.g. RSA Factoring Challenges [6]. Most recent record holds factorization of RSA-768 by Lenstra et al. [7]. It involved usage of hundreds of machines over a span of two years in research. This was accomplished using a highly optimized General Number Field Sieve (GNFS) [8] algorithm. A GNFS algorithm has typically the best known asymptotic running time complexity:

$$O\left(exp\left(\left(\frac{64}{9}\right)^{\frac{1}{3}}(\log b)^{\frac{2}{3}}\right)\right) \tag{1}$$

The approaches taken by the research community in solving the problem can be classified into one of the following categories:

1. First Category or Special Form: Special form algorithms exploit the special structure or properties of either the semi-prime to be factored or one of its unknown prime factors.

2. Second Category or General Form: The running time of a general-purpose factoring algorithm depends entirely on the size of the integer number required to be factored.

3. Third Category or Alternative Form: This category is immensely diverse and involve approaches like genetic algorithms [9], neural networks [10], Oracles [11], Meta-heuristics [12].

This paper focuses on the Third category, and emphasizes on evolutionary computing algorithms like genetic algorithms and swarm intelligence algorithms. There have been such alternative approaches to solving integer factorization found in [9], [10], [11] and [12] which suggest the growing interest of researchers across the academia towards nature-inspired computing for integer factorization. However, these approaches still face the problem of correctly designing an objective function that can provide a selection pressure for converging towards the solution, and hence become difficult to scale. The objective of this paper thus is to explore these flaws and difficulties, and also present some observations, and what we require to design a good objective function.

Section 2 presents research work done on genetic algorithm for integer factorization and discusses the flaws taken in a seemingly correct objective function. Section 3 is a study on swarm intelligence algorithms and further discusses the challenges involved. In Section 4, we present a discussion on the formulation of objective functions for integer factorization as discrete optimization tasks and their issues respectively, followed by a brief mathematical discussion on error objective function in Section 5. We then present some new ideas to reduce space search by introducing more equations in Section 6. Section 7 presents a criticism of the evolutionary and other bio-inspired computing approaches used for integer factorization problem based on Information Search Theory due to [15]. We finally conclude the paper in Section 8.

2. Genetic Algorithm

Yampolskiy [9] presented a genetic algorithm based approach for integer factorization. For experiments, we also adopted a similar strategy for solving the problem, however, with subtle differences in the objective function, with different crossover and mutation strategy.

We assume that the number of digits in the prime factors (say p and q) of semi-prime N is same since we are

interested in solving the most difficult cases of integer factorization. The earlier approach involved an objective function based on the similarity of digits of the target semi-prime number and the semi-prime number computed from the prospective evolved prime factors. Our approach involved addition to the original fitness function in the way that we assign weights to the respective decimal places, with the less significant places being given the most weights.

In our implementation we have assumed a chromosome to be a string consisting of the factors (say p and q) of semi-prime N. We assume that the number of digits in the factors are same since we are interested in solving the most difficult cases of integer factorization. The earlier approach involved an objective function based on the similarity of digits of the target semi-prime number and the computed semi-prime number from prospective prime factors evolved. We have changed the objective function such that not only the similarity between the digits but also the position at which there is a difference in digits is taken into consideration. For example if we need to factorize 437 then the number 436 will be considered closer to 437 than 337 in terms of fitness because in 436 there is a difference of 1 at one's position and in 337 there is a difference of 1 at hundredth position. Fitness value of a chromosome is calculated by summing the product of modulus of difference of digits of the target semi-prime number and calculated semi-prime number from rightmost digit to left most with a multiplying factor, which itself is increased by a constant factor as we go from right to left.

In crossover the parents are selected such that we have one fittest and one weakest member of the remaining population. The chromosomes in our population are a string consisting of the factors (say p and q) of the semi-prime number N, therefore during the crossover we ensured that only some genes of p or q are exchanged with the parents and not the whole parts of chromosomes as done in earlier approaches.

We have performed mutation on the fittest members of our population such that rate of mutation is always greater than rate of crossover so that we can overcome the problem of local minima. In mutation we have randomly selected a gene of the chromosomes and mutated it such that it is different from its current value taking care that the last digit of factors are 1,3,7 or 9 only.

There is an implicit but major flaw in both the approaches. Suppose we need to factor 437 which is basically the product of 23 and 19. According to this approach, 0673 is a better solution than 1822 since 06*73 = 438 evaluates to a better fitness value than 18*22 = 396. However, we know that 1822 is a much closer solution than 0673. Therefore, this approach may provide solutions p and q whose product provides an integer having most of its digits same as the target integer to be factored, but these solutions may be far away from the actual solutions. This approach doesn't take this into account, and therefore fails to converge in many cases. In this view, mutation plays a very important role. However increasing the mutation rate has no significant effect since then it is merely a random phenomenon that a solution is obtained. Hence, owing to missing a selection pressure, this method does not converge for every instances of integer factorization, and may lead to erroneous results.

However, it must be noted that this method does provide an insight and a plausible direction in solving the problem, but definitely requires additionally fitness functions to be incorporated.

3. Other Approaches

There has been a growing interest among researchers in adopting and exploring the potential of swarm intelligence algorithms and neural networks for solving integer factorization. Laskari [10] used neural networks for solving small instances of integer factorization problems (up to 10^7). Dass et al [14] explored binary versions of Differential Evolution and Particle Swarm Optimization for factoring semi-prime numbers. Mishra et al [12] adopted a multithreaded chaotic firefly algorithm to factor integers as large as of the order 10^{14} within given resource limits. Mishra [13] also devised a molecular geometry optimization based heuristic algorithm for factoring semi-prime numbers, the results being more efficient than previously reported results using firefly algorithm, GA and PSO.

It must be noted the above approaches used certain exact mathematical functions in number theory as their objective functions. We shall be exploring these functions, and discuss why they do not lead to convergence of these algorithms in many cases and preclude them from scalability. Later we will discuss more such objective functions and try to understand the criteria of design of objective functions that can actually help us to factor a semi-prime.

4. Formulation of Objective Function

Most approaches taken so far are based on the congruence of squares method. Given a positive integer N, the method relies on figuring out x and y such that

$$x^2 - y^2 = N \tag{2}$$

We can then factor N as (x+y)(x-y) and thus obtain the required prime factors. However, this method is quite slow in nature, and it suffices to a weaker version of the method:

$$x^2 \equiv y^2 (mod\ N),\ \ x \neq y (mod\ N) \tag{3}$$

This is equivalent to

$$x^2 - y^2 \equiv 0 (mod\ N) \tag{4}$$

And therefore N | (x+y)(x-y) but x ≠ y(mod N). N divides neither (x+y) nor (x-y) alone, which means that both (x+y) and (x-y) contain proper factors of N, wich can be obtained using their respective GCD with N.

Eq. (4) forms the objective function for alternative approaches since this can be modelled as a discrete optimization task as follows:

$$minimize\ f(x) = (x^2 - y^2)|(mod\ N) \tag{5}$$

$$constraint\ (x \pm y)(mod\ N) \neq 0 \tag{6}$$

$$x, y\ \in [2, N-1] \tag{7}$$

The constraint can further be reduced to just $(x - y)(mod\ N) \neq 0$ by reducing the domain of x and y to $[2, \frac{N-1}{2}]$

Below is the surface plots and contour plots for the above objective function for N = 35.

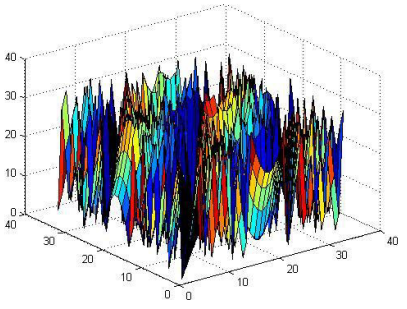

Fig. 1. Surface lot of $(x^2 - y^2)|(mod\ N)$ for N = 35.

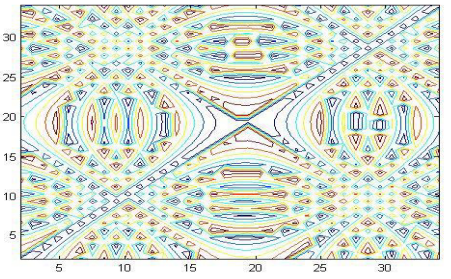

Fig. 2. Contour plot of $(x^2 - y^2)|(mod\ N)$ for N = 35.

There are multiple solutions possible for pairs of x and y. Therefore, it simply becomes a task to figure out just one of such pairs, and that solves our problem. From the contour plot, we can observe somewhat symmetry in the contour of the optimization function, which can be exploited to one's advantage. Mishra et al [13] successfully adopted this objective function with MGO algorithm, but we would observe that the surface plot of the function is very chaotic in nature involving sharp drops and steeps, which then leads to getting trapped in some local minima. It then becomes a matter of chance to click through a solution that ends the algorithm with the correct solution. Observe that the mean standard deviation is also around the same order as the mean function evaluation which authenticates our statement. Metaheuristics and evolutionary computing techniques require direction to move, but in this chaotic with sharp edges scenario, there is hardly any direction or selection pressure that can be provided except in an extremely confined local domain.

Let us consider another objective function which is a single variable mathematical function:

$$f(x) = N(mod\ x) \tag{8}$$

This function has an extremum (minimum) at 0 when x divides N. Therefore, when N is a semi-prime number, this function evaluates to zero only when x is one of the factors of N. The question now is what is the domain of x? Obviously x lies somewhere between 2 and (N-1). The search space can be further reduced to $[2, \sqrt{N}]$ since one of the factors is definitely less than equal to $\sqrt{N}$, which drastically reduces the search space by $\sqrt{N}$.

Thus the objective function in this case can be designed as:

$$minimize\ f(x) = N\ (mod\ x), x \in [2, \sqrt{N}] \quad (9)$$

Since we are interested in most difficult instances of the integer factorization problem where the two factors are of equal size, we can further reduce the search space for x. Now, the domain of x will contain all integers of the order of the same number of digits as figured out from N, but always less than equal to $\sqrt{N}$. That is, if N comprises n digits, its factors p and q would consist of atmost (n/2) digits. Therefore x here will be of the order of $10^{n/2}$.

Unlike the previous case of congruence of squares where multiple solutions were possible, in this case, there is exactly one and only solution to this function.

Below are the function plot and stem plot for the above single variable objective function for N = 4757. As we can see here again that the objective function suffers from sharp edges and steep slopes, with a lot of local minima. This might again lead to trapping and prolonged running of the programming without converging to the actual solution. The local minimum does imply that the solutions obtained are closer to the actual solution as can be observed from the plots below. Also, as used in some of the research work including those by Das et al [14] and Mishra et al [12, 13], the mean standard deviations are again of the same order as that of the mean function evaluations. This clearly shows the unknown dynamics of the problem, which prevents us from forming an objective function that can reasonably provide the direction to obtaining the correct solution. Scalability yet becomes questionable in this regard.

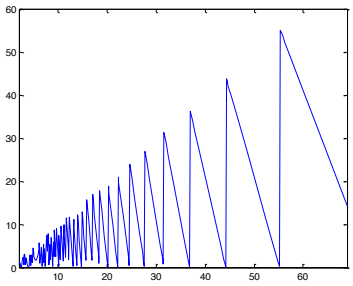

Fig. 3. Function plot of $N(mod\ x)$ for N = 4757

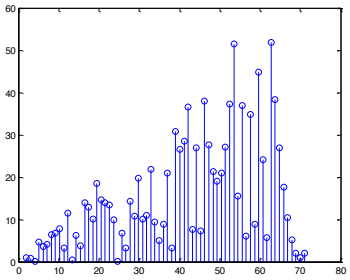

Fig. 4. Stem plot of $N(mod\ x)$ for N = 4757

An interesting function (not used yet in literature) can be noted as follows:

$$minimize\ f(x) = |\log(N) - \log(x) - \log(y)|,\ \ x, y\ \in [2, N-1] \quad (10)$$

There is just one minimum of this function which is zero, and this seem to be a promising objective function because it presents an almost smooth continuous function plot with not many sharp edges and steep slopes. Selection pressure can be possible in this case.

The search space can further be reduced on similar reasons as mentioned earlier. The domain of x can be reduced to $[2,\ \sqrt{N}]$, and that of y as $[\sqrt{N}, N]$. Further incorporating our assumptions, we can further reduce the search space and thus have domain consisting of numbers within the same order of digits.

Let us consider an example, say N = 4757. The factors of 4757 are 67 * 71. The floor value of the square root of 4757 is 68. Since 4757 is 4 digit long, therefore we assume our solution to be 2 digit long. Thus the domain for x becomes [10, 68], and that of y becomes [69, 99].

Below are contour plots for different values of N (41*19 and 131*263):

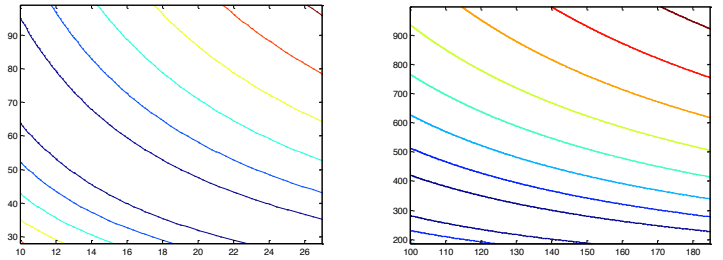

Fig. 5. (a) N = 41*19 (b) N = 131 * 263

As we see that the contours for this objective function is smoother than the previous objective functions discussed. This might lead to an interesting direction since we can see selection pressure being present in this case.

5. Error Functions for approximate factors

Suppose using some approximation algorithm, we approached an approximate solution for factors of N, say p and q. The exact factors were P and Q. Thus, we now need to figure out the error in p and q such that we can get to the final solution. Any recursive or iterative approach should be convergent in nature. Thus the error function which can act as an objective function in figuring out the final factors should be convergent in nature.

If x and y are errors in p and q respectively, then

$$P - x = p \tag{11}$$

$$Q - y = q \tag{12}$$

$$N = PQ = (p + x)(q + y) \tag{13}$$

This equation yields $qx + py + xy = N - pq$. Thus we have one equation with two unknowns. The question now becomes how big can x and y be, what are their domains. It totally depends on what our algorithm computes for p and q. Having any hint or knowledge on the boundaries of x and y can greatly simplify our problem. An unconstrained optimization task can be formulated to act as an error function:

$$minimize\ |qx + py + xy + pq - N| \tag{14}$$

Additional constraints can be added if we look further deep into the problem. We know that one of the factors is less than or/and equal to $\sqrt{N}$ and the other factor is greater than or/and equal to $\sqrt{N}$. Thus additional constraints become:

$$p + x \leq \sqrt{N} \tag{15}$$

(16)

Thus a linear programming problem can be devised for the error function:

$$qx + py + xy + pq - N = 0 \tag{17}$$

$$a \leq p + x \leq \sqrt{N} \tag{18}$$

$$\sqrt{N} \leq q + y \leq b \tag{19}$$

Where *a* represents the minimum order of the solution, and *b* represents the maximum order of the solution. If the approximate solutions obtained are assumed to be reasonably close to the actual solutions, the search domain for both *x* and *y* can further be reduced. This requires a measure of how close we are to the actual solution. This boils down to the same problem of effective design of the objective function for the approximation algorithm.

Another way to look at the problem is to represent the factors as $P = (\sqrt{N} - x)$ and $Q = (\sqrt{N} + y)$. Note that $\sqrt{N}$ is floored for P and ceiled for Q. Their product (floored value and ceiled value) is n. Then,

$N = PQ = n - xy + \sqrt{N}(y - x)$ is an approximate function that can represent another objective function in terms of errors.

6. Reducing search space by creating more solutions

We can reduce the search space by introducing more equations. We have $N = PQ$. Now we multiply both sides of the equation with a small prime number say S. Thus we have $NS = PQS$. Evolving P in both the equations together can bring us closer to the solution. Thus we can create multiple solutions equations to generate our actual solution. Or we can multiply both sides of the equation with a set of small prime numbers, thus creating multiple extrema for our solutions, and thus we can figure out the factors of N. However, it must be noted here again, that the GA approach adopted earlier will still face the same flaw in this case also, but incorporation of a better fitness function will definitely yield solutions much efficiently.

7. Critiquing Evolutionary-Search and other Bio-Inspired Algorithms based on Conservation in Information in Search

Now we know that exactly two integers will exist, those are the prime factors of the semi-prime under consideration. Therefore for a semi-prime number, N, the probability of finding the two factors using a random search is

$$p = \frac{1}{N} x \frac{1}{N} = \frac{1}{N^2} \tag{20}$$

Therefore the endogenous information [15] measure:

$$I_\Omega = -\log p = 2 \log N \tag{21}$$

Now to measure the exogenous information [15], we need to know the problem-specific structure that the search algorithm takes into account. For example, if we just evolve one single factor using the objective function as defined in Eq. (8), then the probability of finding that factor is

$$q = \frac{1}{N} \tag{22}$$

Hence, the exogenous information measure [15] will be

$$I_s = -\log q = \log N \tag{23}$$

Therefore, the active information measure [15] for this will be

$$I_+ = \log N \tag{24}$$

Now, if we take Eq. (2) into account, let us say that there are *m* solutions that satisfy the equation. The number of ways in which we can select two integers in [2, (N-1)/2] is

$${}^{\frac{N-1}{2}}C_2 = (N-1)(N-3)/8 \tag{25}$$

Therefore $q = \frac{8m}{(N-1)(N-3)}$

Hence,

$$I_s = -\log q = \log(N-1) + \log(N-3) - \log(8m) \tag{26}$$

Since N>>1, we can safely approximate (N-1) and (N-3) to be N, for ease of understanding. Therefore,

$$I_s = 2\log(N) - \log(8m) \tag{27}$$

And thus we get

$$I_+ = \log(8m) \tag{28}$$

However, m is unknown, and to predict the active information available to us thus becomes unknown. The greater the value of m the more active information is available to us. Thus it becomes of problem of determining the distribution of the solutions in the domain.

Let's us assume the objective function now to be Eq. (10). We know that one factor lies in [2, $\sqrt{N}$] and the other lies in [$\sqrt{N}$, N]. Therefore, q in this case can be approximated as

$$q = \frac{1}{N\sqrt{N}} \tag{29}$$

Which gives

$$I_s = -\log q = 1.5 \log N \tag{30}$$

Therefore,

$$I_+ = 0.5 \log N \tag{31}$$

One must note that this value is the same if we adopted simple N = xy equation with the same constraints. However, using logarithmic objective function can reduce search space substantially. For example for N of the order of 10^{100} the search space if log x is considered to be a variable is of the order 100. This can help us to closely approximate solutions with an available gradient. However, the accuracy of the solutions might again make us fall into the trap. Therefore, the limitation thus becomes recurrent in this problem, where design of objective function and selection pressure and the sole factors that can decide the fate of the algorithm.

8. Conclusions

In this paper we have surveyed a number of bio-inspired algorithms applied to the problem of Integer Factorization. Our analysis indicates that approaches based on seeking gradual improvement in the solution space such as Genetic Algorithm tend to perform poorly on non-trivial instances of the integer factorization problem. This is due to the all-or-nothing nature of the factorization problem, either a number is a factor or it is not. In this domain it appears no heuristic information is available to discard incorrect solutions or to navigate to more promising areas of the search space. Fitness functions based on similarity of digits between product of potential solution factors and number being factored, used to attempt to narrow down promising areas in the search space, produces almost no reduction due to the large number of local maxima points. It may be possible that in the future breakthroughs in the number theory will lead to design of successful heuristics for the integer factorization problem but currently available methods are unlikely to scale to the size of semiprimes used in real world cryptographic applications.

References

1. Rivest, R.; Shamir, A.; Adleman, L. "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems". Communications of the ACM 21 (2): February120–126, 1978.
2. Rabin,Michael. Digitalized Signatures and Public-Key Functions as Intractable as Factorization. MIT Laboratory for Computer Science, January 1979.
3. Duhamel, P.; Vetterli, M. "Fast Fourier transforms: a tutorial review and a state of the art"/ Signal Processing **19** (4): 259–299, 1990.
4. Shor, P.W. (1997) 'Polynomial time algorithms for prime factorization and discrete logarithms on a quantum computer', SIAM Journal Sci. Statist. Computing, Vol. 26, pp.1484–1509.
5. S. Goldwasser and M. Bellare, Lecture Notes on Cryptography, July 2008, retrieved from http://cseweb.ucsd.edu/~mihir/papers/gb.pdf in Jan-Feb 2013.
6. RSA Challenge Numbers. Retrieved from http://www.emc.com/emc-plus/rsa-labs/historical/the-rsa-challenge-numbers.htm on Nov. 12, 2013
7. T. Kleinjung, et al, "Factorization of a 768-Bit RSA Modulus", Advances in Cryptology – CRYPTO 2010, Lecture Notes in Computer Science, vol. 62223, pp. 333-350.
8. Pomerance, C. 'A tale of two sieves', Notices of the AMS, Vol.43, pp.1473–1485.
9. Yampolskiy, R. V. 'Application of bio-inspired algorithm to the problem of integer factorization', International Journal of Bio-inspired Computation. Vol. 2, No. 2, pp. 115-123, 2010.
10. Laskari, E. C.; Meletiou, G. C.; Tasoulis, D. K.; Vrahatis, M. N. "Studying the performance of artificial neural networks on problems related to cryptography," Nonlinear Analysis: Real World Applications, Vol. 7, pp.937–942, 2006.
11. Maurer, U. and Rueppel, R. (Eds.). "Factoring with an Oracle". Springer-Verlag, 1992.
12. Mishra, M., Chaturvedi, U., Pal, Saibal K.. 'A Multithreaded Bound Varying Chaotic Firefly Algorithm for Prime Factorization', IEEE International Advance Computing Conference (IACC), 21-22 Feb, Gurgaon, pp. 1321-1324, Feb. 22-232014.
13. Mishra, M; Chaturvedi, U; Shukla, K. K.; "A New Heuristic Algorithm based on Molecular Geometry Optimization and Its Application to the Integer Factorization Problem". Published in the Proceedings of the 2014 International Conference on Soft Computing and Machine Intelligence, pp. 115-120, New Delhi, India, Sept. 26-27, 2014.
14. Dass, P.; Sharma, H.; Bansal, J.C.; Nygard, K.E., "Meta heuristics for prime factorization problem," Nature and Biologically Inspired computing (NaBIC), 2013 World Congress on , vol., no., pp.126,131, 12-14 Aug. 2013. doi: 10.1109/NaBIC.2013.6617850.
15. Dembski, W.A.; Marks, R.J., "Conservation of Information in Search: Measuring the Cost of Success," Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on , vol.39, no.5, pp.1051,1061, Sept. 2009.