Particle Swarm Optimization Algorithm for Integer Factorization Problem (IFP)

Bhargab Choudhury
Department of Information Technology
North Eastern Hill University
Shillong 793022 India

Sangita Neog
Department of Information Technology
North Eastern Hill University
Shillong 793022 India

ABSTRACT
This paper presents particle swarm optimization (PSO) method to find the prime factors of a composite number. Integer factorization is a well known NP hard problem and security of many cryptosystem is based on difficulty of integer factorization. A particle swarm optimization algorithm for integer factorization has been devised and tested on different 100 numbers. It has been found that the PSO method performs with little variability over swarm size.

General Terms
Swarm Intelligence, Cryptanalysis

Keywords
Legendre Congruence, PSO

1. INTRODUCTION
Number theory is one of the purest and oldest branches of mathematics, but it has turned out to be one of the most useful when it comes to computer security. In 1801 Gauss identified primality testing and integer factorizing as the two most fundamental problems. Integer factorization or prime factorization is the decomposition of a composite number into non-trivial divisors, which when multiplied together equals to the original integer. Many cryptographic algorithms are based on the difficulty of factoring large composite integer or a related problem, e.g. RSA. The RSA [9] cryptosystem was proposed in 1978 by R.L. Rivest, A. Shamir and L. Adlenman, is the most well known public key cryptosystem. It is widely used to secure the information in the insecure channel [12]. It is also implemented in most web servers and browsers, and present in most commercially available security products. RSA algorithm uses a pair of keys to encrypt and decrypt a message. One key is used to encrypt a message, called public key and the other key is used to decrypt a message, called private key. The security of RSA is based on the difficulty of integer factorization. The integer factorization problem (IFP) is a well-known topic of research within both academics and industry. The mathematical definition of IFP is given below:

Given N, find primes p_j for i = 1, 2....., r with $p_1 < p_2 < .... < p_r$ and ej ε N for j = 1, 2,….,r such that $n = p_1^{e1}. p_2^{e2} ….. p_r^{er}$

2. LEGENDRE'S CONGRUENCE
If we have two integers x and y such that

$x^2 \equiv y^2 \bmod N$, $0<x<y<N$, $x \neq y$, $x+y \neq N$

then gcd(x-y, N) and gcd(x+y, N) are possibly nontrivial factors of N, because N divides (x+y)(x-y), but N does not divide (x-y) and (x+y). The congruence is often called Legendre's Congruence [13].

3. THE PSO APPROACH
Kennedy and Eberhart proposed particle swarm optimization (PSO) [7] method based on social behavior of a flock of migrating birds. Particles are abstract entities. The set of particles is called a swarm. Each particle has position and velocity and also equipped with a small memory comprising its previous best position (pbest) and a global best position (gbest). Particles are evaluated by a fitness function. The main strength of PSO over other nature inspired computing is its fast convergence. Information sharing mechanism in PSO is one way. The gbest particle gives out the information to others particles. Pseudo code of PSO [4] is given below.

For each particle

    Initialize particles position and velocity

End

Do

    For each particle

        Calculate fitness value

        If fitness value is greater than the best fitness value then set current value as the new pbest

    End

    Choose the particle with the best fitness value of all the particles as the gbest

    Calculate particle velocity

    Calculate particle position

End

While maximum iteration or goal is not attained

Flow diagram illustrating the particle swarm optimization algorithm is shown in Figure 1.

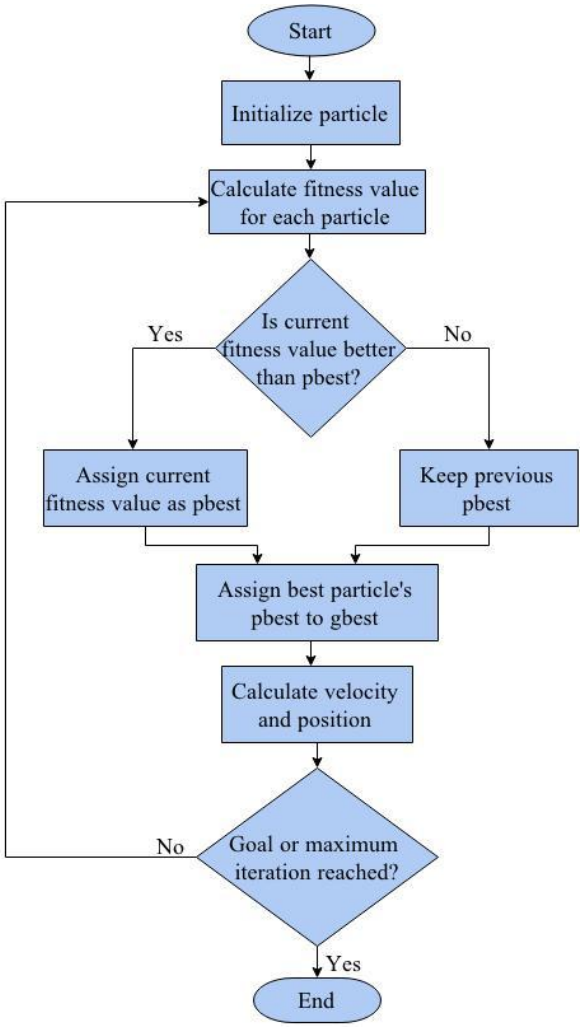

Fig 1: Flow chart of PSO

3.1 Initialization

Particles are initialized as a two dimensional vector $X=(x_1, x_2)$, where X takes a random integer value between 1 and $N^{1/2}$ (N is the input number) such that $x_1<x_2$. Particle's velocity also initialized as a two dimensional vector $V=(v_x, v_y)$, where V takes a random integer value between 1 and (N-3)/10.

3.2 Fitness Function

Fitness function value gives the effectiveness of the particle in the search space. The value of the fitness function indicates the distance between the current position of the particle and its solution. The fitness function is defined as:

$f(x_1, x_2)=abs(x_1^2-x_2^2) \bmod N,\ 0<x<y<N,\ x\neq y,\ x+y\neq N$

If f=0 for a particle, then that position of the particle is taken as a solution.

3.3 Update Velocity and Position

The velocity, V_t and position, X_{t+1} of the particle P at time t is updated using the following formula [4]:

$V_{t+1}=\omega*V_t + c_1*\varphi_1*(pbest_t-X_t) + c_2*\varphi_2*(gbest_t-X_t)$

$X_{t+1}=X_t+v_{t+1}$

Where:

ω: The inertia weight ω controls the momentum of the particle.

c_1, c_2: Acceleration coefficients

φ_1, φ_2: Two random variables in the interval [0, 1] which inject the unpredictability of the particles movement

The first part in the velocity updating formula represents the inertial velocity of the particle. A higher value of ω facilitates global search while a lower value of ω facilitates local search [10]. ω determines rate of contribution of previous velocity. There are many inertia weight strategies in PSO. Banasal et. al [3] stated that Chaotic Inertia Weight is the best strategy for better accuracy and Random Inertia Weight strategy is best for better efficiency. The second part in the velocity updating formula is called cognition part and it represents personal experience of the particle [11]. The third part is called social part and it represents global experience of the particle [11].

3.4 Boundary Condition

Boundary condition is an important part of PSO which restricts the particle to fly within the feasible space. There are many boundary conditions: absorbing, reflecting, invisible, damping [6]. We have developed a boundary condition utilizing the gbest at time t and the current position of the particle and set velocity equal to zero in that direction. It is represented as follows.

$X_{t+1}=abs(R\times(gbest_t-X_{t+1}))+X_{min}$

$V_{t+1}=0$

Where:

R: A random number in the interval [0, 1]

X_{min}: Lower bound of X

3.5 Neighborhood Policy

Inside the swarm a topology is defined. It is a set of links between particles, saying "who informs whom". The set of particles that informs a particle is called its neighbourhood. There are many neighbourhood topologies [8]: ring, fully connected, mesh, star, and tree as shown in the Figure 2. We have used fully connected topology, because the PSO provide a faster convergence by focusing on the best position encountered by all particles by taking the whole population as its neighbors [1]. All nodes in this topology are directly connected among each other.

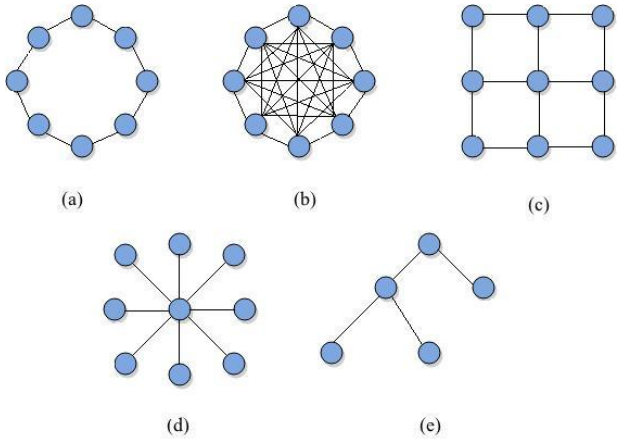

Fig 2: (a) Ring topology (b) Fully connected topology (c) Mesh topology (d) Star topology (e) Tree topology

4. COMPUTATIONAL EXPERIENCE

4.1 Experimental Setup

The swarm size is taken to be 20, 30, 40, 50, and 60. In our experiment three inertia weight strategies are considered- constant inertia weight, random inertia weight [2], and chaotic inertia weight [5]. The value of ω for constant inertia weight strategies is set to 0.76. Random Inertia Weight and Chaotic Inertia Weight are computed according to the equation (1) and (2) respectively.

$$\omega=0.5+Rand()/2 \quad (1)$$

Where:

Rand(): Rand() is a random number within the range [0, 1]

$$\omega=(\omega_1-\omega_2)\times(MAXiter-iter)/MAXiter+ \omega_2\times z \quad (2)$$

Where:

ω_1: Initial value of inertia weight

ω_2: Final value of inertia weight

MAXiter: Maximum iterative time

iter: Current iterative time

$z=4\times z\times(1-z)$, provided that initial value of z in the range (0, 1)

Values for ω_1 and ω_2 are set to 0.9 and 0.4 respectively. The value of acceleration coefficient c_1 and c_2 are taken equal to 2.25. Maximum number of iteration is set to 50. For each value of N, 50 independent runs are performed. We have tested over 100 random numbers. The PSO is coded in C, and run using an Intel Core 2 Duo 2.20 GHz PC with 3GB RAM.

4.2 Result and Analysis

The performance criteria [2]: Success Rate (SR) and Number of Function Evaluation (NFE) are shown in the Table 1 and Figure 4. The success rate is computed according to the equation (3).

$$SR= G_{times}/T_{runs} \quad (3)$$

Where G_{times} is the total number of times the set goal was reached over 50 independent runs and T_{runs} is the total number of independent runs.

The number of function evaluation is computed according to the equation (4).

$$NEF= (S_{size}\times t_{avg})/SR \quad (4)$$

Where S_{size} is the swarm size and t_{avg} is the average number of iterations the set goal was reached over 50 independent runs. The experiment result is shown at Table 1. Variation of SR, average iterations and time with swarm size for Chaotic Inertia Weight, Random Inertia Weight and Constant Inertia Weight are shown in Figure 3-5. In Figure 3 it is observed that SR increases with increase in swarm size and maximum SR is observed in Random Inertia Weight Strategies. Figure 4 represent the variation of average iterations with respect to swarm size. It is found that average iterations decrease with increase in swarm size and minimum average iterations is observed for Random Inertia Weight. Figure 5 represent the variation of execution time with swarm size. It is found that execution time increases with increase in swarm size and minimum time is observed for Random Inertia Weight. As an overall it is observed that PSO with Random Inertia Weight perform outstanding comparing with Chaotic Inertia Weight and Constant Inertia Weight.

Table 1: Results Achieved from Experiments

Swarm Size	Success Rate (SR)	Average Iterations	NFE	Time (S)
Chaotic Inertia Weight				
20	0.347	17.839	1028.184	0.0502
30	0.428	15.602	1093.598	0.0533
40	0.544	14.287	1050.515	0.0551
50	0.552	14.728	1334.058	0.0583
60	0.613	14.788	1447.439	0.0598
Random Inertia Weight				
20	0.405	18.016	888.679	0.0501
30	0.486	16.724	1032.346	0.0538
40	0.565	16.397	1160.850	0.0551
50	0.575	14.187	1233.652	0.0580
60	0.657	13.064	1193.059	0.0587
Constant Inertia Weight				
20	0.394	17.084	867.208	0.0505
30	0.489	14.089	864.356	0.0531
40	0.548	15.844	1156.496	0.0560
50	0.616	13.955	1132.711	0.0575
60	0.632	15.213	1444.272	0.0609

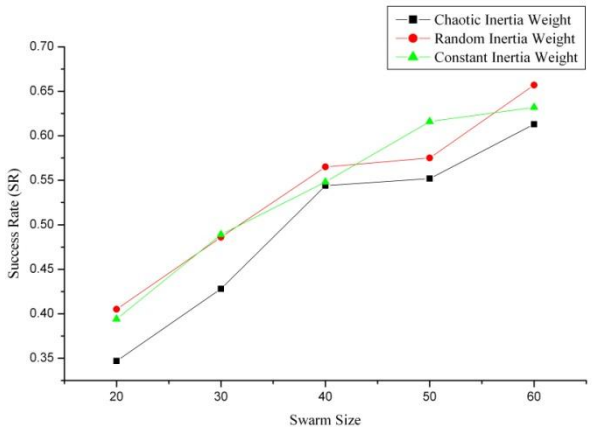

Fig 3: Variation of SR with swarm size

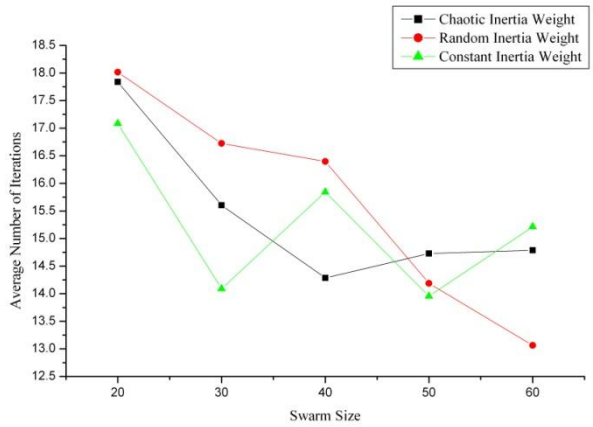

Fig 4: Variation of average iterations with swarm size

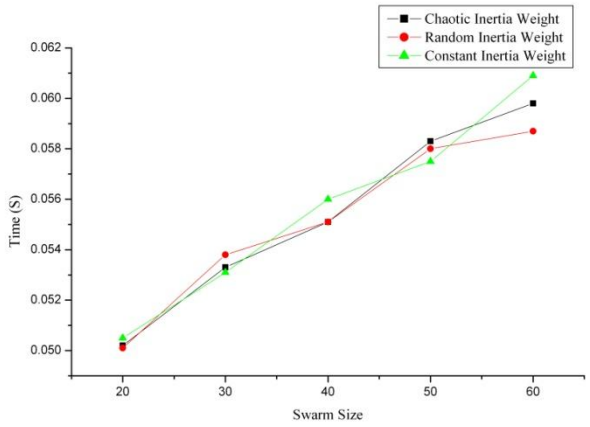

Fig 5: Variation of time with swarm size

5. CONCLUSION

This paper presents a methodology involving particle swarm optimization to find the factors of an integer number. The particles are represented as integer particles and are guided using fitness function. The particle's previous best (pbest) and global best (gbest) positions of the particles help the procedure to move towards the solution. The procedure follow fully connected neighbourhood topology. We have used a new boundary condition for finding the factors of an integer number. The procedure has been validated with 100 numbers of different sizes. From our experiment we can infer that our approach can find the factors of an integer number and good performance is observed for swarm size 50-60. The further works involve in scaling the procedure in tackling very large number and increasing success rate (SR).

6. REFERENCES

[1] Abraham, S., Sanyal, S., and Sanglikar, M. 2010. Particle Swarm Optimization Based Diophantine Equation Solver, International Journal of Bio-Inspired Computation 2(2), 100-114

[2] Arasomwan, M. A. and Adewumi, O. A. 2014. An Investigation into the Performance of Particle Swarm Optimization with Various Chaotic Maps, 2014. Hindawi Publishing Corporation Mathematical Problems in Engineering Volume 2014, Article ID 178959, 17 pages.

[3] Bansal, J. C., Singh, P. K., Saraswat, M., Verma, A., Jadon, S. S., & Abraham, A. (2011, October). Inertia weight strategies in particle swarm optimization. In Nature and Biologically Inspired Computing (NaBIC), 2011 Third World Congress on (pp. 633-640). IEEE.

[4] Das, S., Abraham, A., and Konar, A. 2008. Particle swarm optimization and differential evolution algorithms: technical analysis, applications and hybridization perspectives. In Advances of Computational Intelligence in Industrial Systems (pp. 1-38). Springer Berlin Heidelberg.

[5] Feng, Y., Teng, G. F., Wang, A. X., & Yao, Y. M. (2007, September). Chaotic inertia weight in particle swarm optimization. In Innovative Computing, Information and Control, 2007. ICICIC'07. Second International Conference on (pp. 475-475). IEEE.

[6] Huang, T., and Mohan, A. S. 2005. A hybrid boundary condition for robust particle swarm optimization. Antennas and Wireless Propagation Letters, IEEE, 4, 112-117.

[7] Kennedy, J. 2010. Particle swarm optimization. In Encyclopedia of Machine Learning (pp. 760-766). Springer US.

[8] Medina, A. J. R., Pulido, G. T., and Ramírez-Torres, J. G. 2009. A Comparative Study of Neighborhood Topologies for Particle Swarm Optimizers. In IJCCI (pp. 152-159).

[9] Rivest, R. L., Shamir, A., & Adleman, L. (1978). A method for obtaining digital signatures and public-key cryptosystems. Communications of the ACM, 21(2), 120-126.

[10] Shi, Y., & Eberhart, R. C. (1999). Empirical study of particle swarm optimization. In Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on (Vol. 3). IEEE.

[11] Sun, J., Lai, C. H., & Wu, X. J. (2011). Particle Swarm Optimisation: Classical and Quantum Perspectives. CRC Press.

[12] Vignesh, R. S., Sudharssun, S., & Kumar, K. J. (2009, December). Limitations of quantum & the versatility of classical cryptography: a comparative study. In Environmental and Computer Science, 2009. ICECS'09. Second International Conference on (pp. 333-337). IEEE.

[13] Yan, S. Y. 2002. Number theory for computing. Springer Science & Business Media.