Wasit Journal for Science & Medicine **2009 2 (1) : (1 - 10)**

Integer Factorization Problem Solving Using Tabu Search

Ahmed T. Sadiq Tayseer S. Atia
Abd-alsattar S.Awad
Computer Sciences Department University of Technology

حل مشكلة تحليل الاعداد الصحيحة باستخدام البحث المحرّم

أحمد طارق صادق تيسير سلمان عطية
عبد الستار سالم عوض
قسم علوم الحاسبات الجامعة التكنولوجية

الخلاصة

تقنية البحث المحرّم هي احدى تقنيات البحث عن الهدف في الذكاء الاصطناعي. البحث المحرّم له عدة تطبيقات في مجال حل مشاكل الافضلية. في هذا البحث تم استخدام تقنية البحث المحرّم لحل مشكلة تحليل الاعداد الصحيحة والتي تلعب دور كبير في تحليل الشفرات لانظمة تشفير المفتاح العام. اظهرت النتائج من خلال التجارب التي اجريت في هذا البحث ان الخوارزمية المقترحة نجحت في عملية تحليل الاعداد الصحيحة ونتائجها كانت افضل من الخوارزمية الجينية كمقارنة.

Abstract

Tabu Search (TS) technique is one way of search for goal algorithm in the Artificial Intelligent (AI). TS is used to solve several optimization problems. In this paper a proposed TS algorithm will be presented to solve the integer factorization problem that plays very big role in the cryptanalysis of public-key cryptography. The experimental result of this work shows that the proposed algorithm can factorize the numbers in good time compared with genetic algorithm technique.

1. Introduction

The integer factorization problem (IFP) is the following: given a composite number n that is the product of two large prime numbers p and q, find p and q. While finding large prime numbers is a relatively easy task, the problem of factoring the product of two such numbers is considered computationally intractable if the primes are carefully selected. Based on the difficulty of this problem, Rivest, Shamir and Adleman [1] developed the RSA

public-key cryptosystem. Another public-key cryptosystem whose security lies on the intractability of IFP is due to Rabin and Williams [2, 3].
While the integer factorization problem has received some attention over the centuries from well-known mathematicians like Fermat and Gauss, it is only in the past 20 years that significant progress has been made towards its resolution. There are two main reasons for this phenomenon. First, the invention of the RSA cryptosystem in 1978 stimulated many mathematicians to study the problem. And second, high-speed computers became available for the implementation and testing of sophisticated algorithms. Fermat and Gauss would have had little incentive for inventing the number field sieve factorzing algorithm since this algorithm is more cumbersome than trial division for the purpose of factorzing integers by hand.

There are basically two types of factoring algorithms, special-purpose and general-purpose. Special-purpose factoring algorithms attempt to exploit special features of the number n being factored. In contrast, the running times of general-purpose factoring algorithms depend only on the size of n[4].
One of the most powerful special-purpose factoring algorithms is the elliptic curve factoring method (ECM) that was invented in 1985 by Hendrik Lenstra Jr. [5]. The running time of this method depends on the size of the prime factors of n, and hence the algorithm tends to find small factors first. Just prior to the development of the RSA cryptosystem, the best general-purpose factoring algorithm was the continued fraction algorithm [6], This algorithm was based on the idea of using a factor base of primes and generating an associated set of linear equations whose solution ultimately leads to factorization. This is the same idea underlying the best general-purpose algorithms used today: the quadratic sieve (QS) and the number field sieve (NFS). Both these algorithms can be easily parallelized to permit factoring on distributed networks of workstations. Large mainframe computers or supercomputers are therefore not essential to factor large numbers.

2. Tabu Search

Over the past 12 years, the tabu search has established its position as an effective meta-algorithm guiding the design and implementation of algorithms for the solution of large scale combinatorial optimization problems in a number of different areas [7, 8]. A key reason for this success is the fact that the algorithm is sufficiently flexible to allow designers to exploit prior domain knowledge in the selection of parameters and sub-algorithms. Such prior knowledge is frequently obtained by repeatedly solving a few representative test cases using a spectrum of different techniques, attributes and/or

parameters. Frequently referred to as target analysis, an application of this learning approach is described in Laguna and Glover [9]. The tabu search algorithm combines a few simple ideas into a remarkably efficient framework for heuristic optimization. Among the main elements of this framework are:

- A (usually) greedy local search; the next solution is usually the best not-yet visited solution in the current neighborhood.
- A mechanism (the tabu list) discouraging returns to recently visited solutions.
- A mechanism that changes the solution path (perhaps by a random move) when no progress has been made for a long time.

In addition, many tabu search algorithms incorporate other features such as flexible memory structures and dynamic aspiration criteria. A rough overview of a conceptual tabu search algorithm is given figure 1. For difficult searches, techniques such as influential diversification may be used to extend the duration and scope of the search [10]. Although a tabu search is conceptually simple, any implementation of an efficient tabu search algorithm is problem specific, and no generic tabu search software is available at this time. Among the issues faced by designers of tabu search algorithms are [10]:

- The nature of the information included in the tabu list.
- The way the tabu list is organized.
- The lengths of the tabu lists.
- The types of moves used to create new solutions.
- The implementation of the local greedy search algorithm.
- Strategies for diversifying the search when no progress has been made for a while.

3. Proposed Algorithm

The proposed factorization algorithm make use of some concepts specific to prime number, since the prime number can be one of the following number(1,3,5,7, and 9) then any generated number composite of any two prime number must end with one of these prime number . Under this assumption there is four case help in guess the end digit for both numbers these cases summarized as follow:

1. Case 1 : if the number end with digit 1 then it can be generated by multiplying (1× 1) or (7× 3).
2. Case 2 : if the number end with digit 3 then it can be generated by multiplying (1× 3) or (7× 9).

3. Case 3 : if the number end with digit 7 then it can be generated by multiplying (1× 7) or (9× 3).
4. Case 4 : if the number end with digit 1 then it can be generated by multiplying (1× 9) or (3× 3).

3.1 Solution encoding, move and neighborhood

Each solution in the problem space is encoded as a structure of three fields (p, q, fitness), fitness value Represent the difference between n (the number to be factor) and the product of p and q currently guessed. Move operation used to generate a set of solution from the current solution is the addition, each number (p,q) is incremented by 10 to get next prime number end with the same previous guessed digit. Neighborhood is a list always of size (10×10) to store a set of generated solutions from p and q by applying move operation to 10 iterations.

3.2 Tabu and tabu list

Values of p and q are classified as tabu to facility the multiplication operation and move evaluation in order to get the minimum difference. Separate list for p and q value is maintained, each p value is considered as multiply and still tabu active for 2 iterations while q value is considered as multiplier and is still tabu active for 1 iteration. Tabu list is maintained to store the recently visited solutions. Length of this list is 10 and is used in the proposed algorithm to maintain the value of p and q to yield minimum difference and to make sure that the next generated and selected value from the current neighborhood must be less than the previously generated value in the list.

3.3 Local search

Simple search process to explore the current neighborhood and get the minimum difference is implemented in three steps:
1. **Generating move**: a set of 10 values for each p and q is generated by increment the value by 10. These values are used to generate the neighborhood.
2. **Evaluation of move**: after generating p set and q set; each value from p set is multiplied by all values in the q set. A total of 100 values is computed and stored in the neighborhood and the fitness value for each pair is evaluated.

3. **Selection of move**: neighborhood matrices are searched to get the minimum difference and the values of p and q which generate this difference is transferred to the tabu list on one condition that value must be less than values stored in tabu list and the values are not member in tabu list. Neighborhood list can be sorted ascending and retrieving the first location in the list.

3.4 Diversification

If the present solution path does not appear to be promising, it may be efficient to abandon the local search and use another path; in this case the other path is the second guessing value for p and q.

The proposed algorithm

```
Initialize
    L= maximum length of tabu list
    D= √n
    P,q=intial value    " guess value with length(d) and right digit guessed
                            according to right digit in n'
    J=1                 "the first location in p list
    Z=2                 "no of iteration element must be active in p list
     Sol.p=p
    Sol.q=q
    Sol.fit=n-(p×q)
    Best.fit=n
    Done= false
    Set p list to all possible pstart value
    Set q list to all possible qstart value
    Initialize b list all to 1  " this list control  the selected q value to be used
                                    with indexed p value by j
    Initialize tabu matrix to 0   " dicorage use the previous selected q value
                                    with current p
Begin
Do while done=false
      If  sol.fit<best.fit then Best=sol
       Begin
            If  sol.fit<> 0 then Add sol to tl
            If length(tl)> 9 thenDelete old entry from tabu list
            Let j= index of  pstart value in plist
             For i=1 to 9
                  If b(i).n=1 then
                        If tabu(j,i) =0 then Set b(i).n=0
```

```
                B(i).count=1
                Tabu(j,i)=1
                Invert all b.n value with b.count=2
                Generate all possible solution and store it in neigh list
                Find the minimum value in neigh list
                If minimum< all value in tl and min.  not  in tl then  Sol=
        min.
        End For
        If z=0 then Set z=2
        J=j+1
        If j=10 then     J=1
        Iter=iter+1
        If iter >threshold and no solution found yet then
                Generate new p and q and diverse from beginning
    End
Else
        Done=true
Until done =true
End.
```

4. Experimental Results

This section shows the test results for the proposed algorithm, different numbers are factored with different value, the output result for factoring and the value of each parameter in the implementation are evaluated, the most important values are p, q, tabu active element in b list, minimum. These values are explored with each implementation is shown in Figures (1, 2, 3). Figure(1) shows the implementation result of the algorithm to factor n=29143, the output shows that only one iteration needs to discovering p and q, tabu column shows that only one item is used from b list indexed 1 which is inverted after selection to discourage select it again.

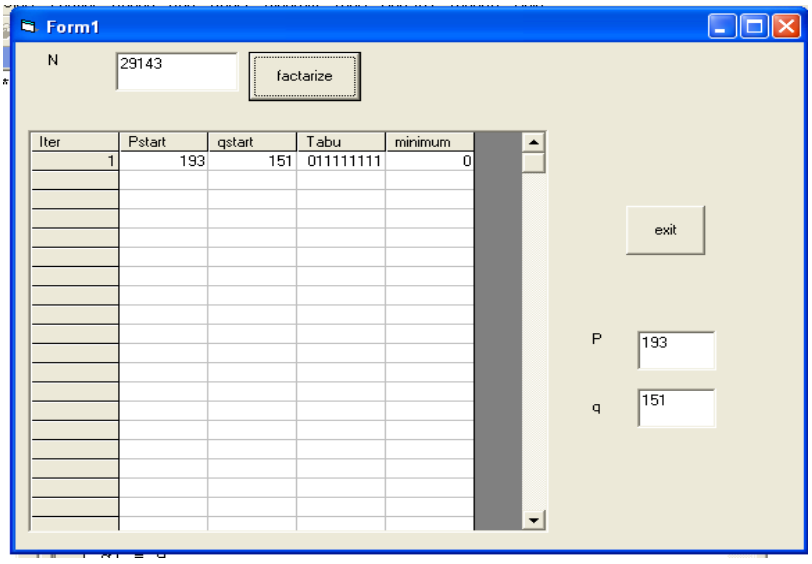

Figure (1)
Result of factorization of n=29143

Figure (2) shows factor of n=122353, and the result p and q is discovered after 153 iterations.

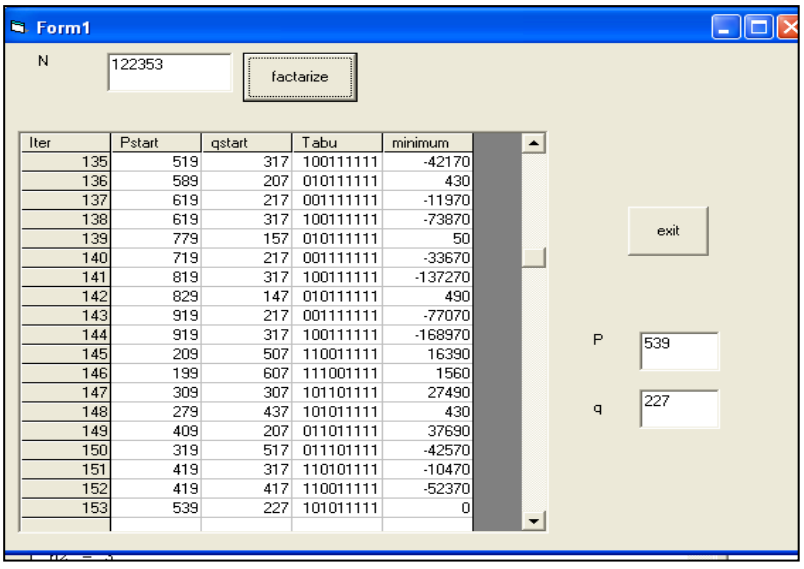

Figure (2)
Result of factorization of n=122353

Figure (3) shows the result of factor n= 124089, the results p and q are discovered after 6 iterations.

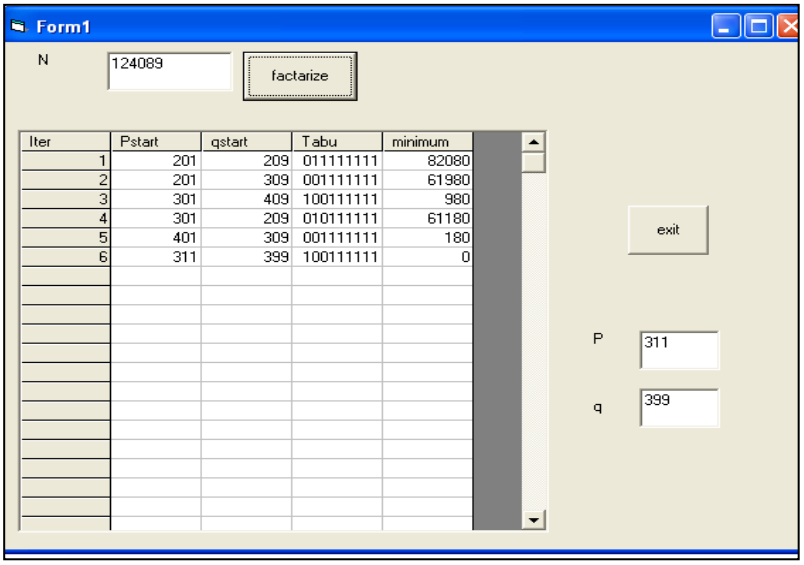

Figure (3)
Result of factorization of n=124089

Figure (4) shows the algorithm running time relative to a number of digit for N, as shown in this figure, the time increases respectively according to the increasing of the number of digit. Also to compare the efficiency of the proposed algorithm with previous work, Genetic Algorithm (GA) is used to solve the factorization problem and attack RSA public key algorithm found in [11].

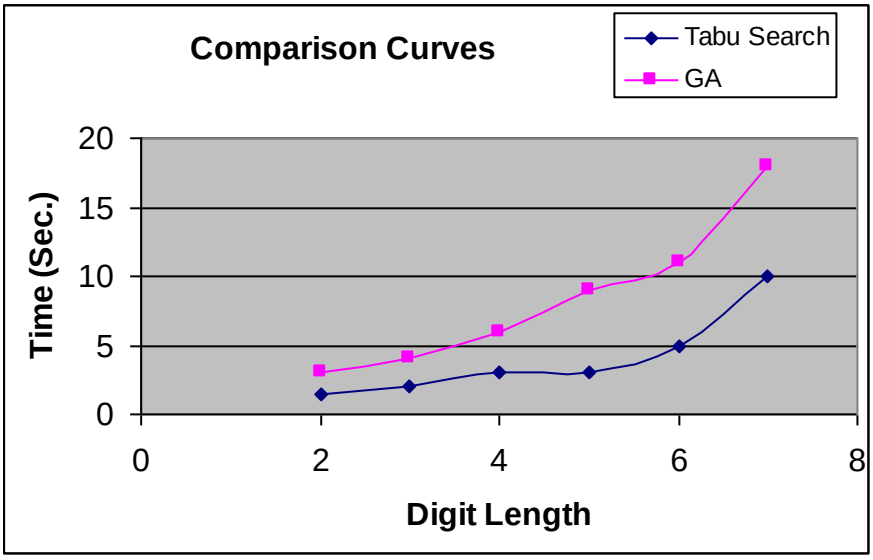

Figure (4)
Comparison Results of Factorization using Tabu Search & Genetic Algorithms

5. Conclusion

This paper presents a special purpose algorithm to factor prime number composite from product of two prime numbers, this algorithm is designed according to the concept of tabu search algorithm and classified as special purpose algorithm since it depends on some concepts in number theory to guess the prime number and guess the size of both numbers depends on n under assumption that both number must be of the same size, experimental result shows that it can factor a number of composite from product of number which has 3 digits correctly and relatively in smaller number of iteration. Our results are compared with those another of algorithm to factorize n which is composite of product of two prime numbers using GA. It shows that the time need factorize n using the proposed algorithm is less than the time needed to factorize it using GA.

References

1. R.L. Rivest, A. Shamir and L.M. Adleman, "**A Method For Obtaining Digital Signatures And Public-Key Cryptosystems**", Communications of the ACM, volume 21, pages 120-126, 1978.

2. M.O. Rabin, "**Digitalized Signatures And Public-Key Functions As Intractable As Factorization**", MIT/LCS/TR-212, MIT Laboratory for Computer Science, 1979.

3. H.C. Williams, "**A Modification Of The RSA Public-Key Encryption Procedure**", IEEE, Transactions on Information Theory, volume 26, pages 726-729, 1980.

4. A Certicom Corp., "**Marks On The Security Of The Elliptic Curve Cryptosystem**", July 2000. http://www.certicom.com

[**5**] H.W. Lenstra, "**Factoring Integers With Elliptic Curves**", Annals of Mathematics, volume 126, pages 649-673, 1987.

6. M.A. Morrison and J. Brillhart, "**A Method Of Factoring And The Factorization Of F7**", Mathematics of Computation, volume 29, pages 183-205, 1975.

7. Glover, F. "**Future Paths For Integer Programming And Links To Artificial Intelligence**", Computers and Operations Research 13, 533-549, 1986.

8. Glover, F. and M. Laguna, "**Tabu Search**",. Boston: Kluwer Academic Publishers(1997).

9. Barnes, J.W. and M. Laguna, "**A Tabu Search Experience in Production Scheduling**", Annals of Operations Research 41, 141-156,1993.

10. Arne Thesen, "**Design and Evaluation of Tabu Search Algorithms for Multiprocessor Scheduling**", Department of Industrial Engineering, University of Wisconsin—Madison, 1513 University Ave, Madison, WI, USA, 53706

11. Mohammed A. Nasser, "**Cryptanalysis RSA algorithm Using GA**", M. Sc. Thesis, University of Technology, 2001.

Recived ……..……. (8/10/2008)
Accepted ……...…….... (2/3 /2009)