An Algorithmic Approach towards Construction of Long Binary Sequences using Modified Jacobi Sequences

B.Suribabu Naick
Asst Prof, Dept of ECE
GITAM University
Visakhapatnam
Andhra Pradesh

P.Rajesh Kumar
Head of Department
Dept. of ECE
Andhra University
Visakhapatnam
Andhra Pradesh

ABSTRACT
Construction of long low autocorrelation binary sequences (LABS) is a complex process which involves many limitations. LABS have many practical applications. In pulse coding schemes, sequences with low autocorrelation side lobe energies are required to reduce the noise and to increase the capability of radars to detect multiple targets. In literature, numerous techniques were employed to solve the LABS problem. For short length sequences, search algorithms can be applied as the search space is manageable. But in our case of long length binary sequences, construction methods are suitable. The major limitations of search algorithms are time and computational power. DH Green [1] in their research utilized modified Jacobi sequences to construct merit factors for long binary sequences. In our case, we used the same construction methods and applied them to various search algorithms. We obtained better results with this implementation. We achieved a merit factor of 6.4534 whereas Green [1] managed to 5.99.

Keywords
Autocorrelation, Modified Jacobi sequences, Merit Factor, prime step algorithm, steepest descent algorithm.

1. INTRODUCTION
Pulse Compression is used in radars to detect the targets. The transmitted signal is phase modulated and then correlated with received signal. If the resulting spectrum consists of low SNR, then the target detection becomes difficult. In order to solve this problem, the sequences used for phase modulation should have low autocorrelation side lobe energies. As the energy decreases, the noise in the spectrum decreases, and the targets become apparent. There are many other applications of LABS like icing spin glasses, etc. Generating LABS is a very laborious process which involves time complexity. If N is the length of the binary sequence, then the search space has 2^Nelements. The process is similar to searching for a needle in a haystack. Lack of proper technique would result in chaos. We can classify the methods in the literature into three types. The first one is for short length (N<50) binary sequences, which involve linear search. It looks in the entire search space. As the length of the sequences increases, regular search is not feasible. Various stochastic and optimization algorithms were developed to approach the problem for medium length sequences (N<200).For long binary sequences, Search algorithms takes infinite time to address the problem. To overcome this limitation, various construction methods are implemented. Legendre, Jacobi. Modified Jacobi Sequences can generate LABS with good merit factor range. We moved a step further by applying these sequences to prime step and steep descent algorithms, and we observed significant improvement in merit factor values. We modified these algorithms to our requirements for better results.

2. LITERATURE
Linder used computer search as a tool to approach the Labs problem. He employed a linear search algorithm for first 32 sequences. Golay described Lindner's research in his publication [4].Later, Mertens [5] used branch and bound technique up to a sequence length of 48.Mertens later employed an exhaustive search for the same element set, and he applied the results to construct accurate states of Bernasconi model. Mertens work was subsequently improvised by Bauke [6], and he conducted his research for first 60 sequences. We compared all the results of various linear search algorithms, their performance is miserable with an increase in sequence size. Computing power is limited, and it is one of the limitations of the exhaustive linear search.

As the sequence length increases, the size of the search space grows exponentially. The search algorithms must be intelligent enough to guess the location of a global optimum. It should not look for a solution in a blunt method. Numerous stochastic algorithms, optimization methods, mutation procedures were employed for medium length binary sequences to obtain the solution. In the beginning, even these methods failed to produce proper outputs. With trial and error methods, better approaches were developed. Prestwich [7] used a CLS algorithm that utilized constraint programming to obtain results up to a length of N=48. After that, Dotu used a Tabu search algorithm to get results up to a range of N=48. Tabu search is a memory aided search process with some restrictions known as "Tabu". Instead of searching the whole space, the Tabu algorithm tries to estimate the location of the global optimum using memory. This mechanism will reduce the search time enormously. But if the sequence length increases beyond 200, even this method will not be able to handle the problem. For long binary sequences, we use construction methods.

When the sequences length increases to five hundred, it becomes almost impossible to use search algorithms. At this point, generation methods will be used. Mathematical functions like Legendre symbols, Jacobi symbols, etc. are used to construct the required sequences. The first method in the literature is the Legendre sequence construction. If N is the length of the binary sequence, we want to obtain the merit factors as N tends to infinity. There is an observation by Turyn that, a quarter rotated periodic Legendre sequence tends to a merit factor value of 6.0 as N tends to infinity [2]. These same phenomena were proved to be true for modified Jacobi sequences of length (pq). Recent improvements include the achievement of merit factor observations up to a range of

3000 by two students [3]. Matthew Parker and Kristiansen [9] used small complexity search to obtain good results; they have extended the Legendre sequences. If N is the length of the binary sequence, we want to get the merit factors as N tends to infinity. We consider only those sequences which have good 2 valued- periodic autocorrelation which are Quadratic residue sequences, GMW sequences, Twin Prime Jacobi sequences-sequences etc. All these sequences are used in cryptography as they exhibit ideal aperiodic correlation properties and have very high linear-equivalence. D.H Green [1] investigated modified Jacobi sequences and implemented a 2-dimensional array to solve their autocorrelation values. We have analyzed Green's results and observed all the other results in the literature. We choose modified Jacobi sequences and have applied them to steepest descent and prime step algorithms with some modifications. We have achieved a merit factor of 6.4534 with these algorithms.

3. BACKGROUND

3.1 Autocorrelation, Energy, Merit Factor

In statistics, we use the correlation as a tool to measure the similarity between two sequences. Autocorrelation is the correlation of a sequence with itself with the presence of some time lag. Autocorrelation function is used to detect the presence of repeated patterns or periodic patterns in a sequence which are otherwise covered by noise

Assume that A is a binary sequence with length N. We will represent the sequence by $a_1a_2a_3\ldots a_N$ with $a_i \in \{-1, 1\}$ for $1 \leq i \leq N$. The *aperiodic autocorrelation* of elements in the sequence *A* is

$$c_m = \sum_{i=1}^{N} a_i\, a_{i+m} \qquad (-N < m < N) \tag{1}$$

Autocorrelation requires only one sequence, whereas cross-correlation requires two.If A and B are two different sequences, the cross-correlation between A and B is X=A*B

$$c_m = \sum_{i=1}^{N} a_i\, a_{i+m} \qquad (-N < m < N) \tag{2}$$

LABS problem for the length N is represented as LABS(N). For a given a given length N, LABS has 2^N solutions. There are 2^N solutions in the search space. We have to select the sequence that provides the least autocorrelation energy value. For example, assume that N=3, we have eight possible solutions. The solutions are$\{1,-1,-1\},\{-1,1,1\},\{-1,-1,1\},\{1,1,-1\}$.The energies of all these sequences are same which is one. As the sequence length is very small, we got the results instantly. But for long length sequences, the complexity increases exponentially

It is not necessary to find all the global optimum in LABS. If we know one result, we can complement it to get another. Even if the obtained result is complemented or reversed, the energy value will remain the same. LABS exhibit symmetry.

The quality of the sequence is measured by its energy value. As the energy value of the sequence decreases, its quality increases. To have a standard approach, we require a measuring function for LABS sequences. Golay for the first time introduced the concept of merit factors. The merit factor indicates the autocorrelation side lobe energy of a given sequence.

$$F(A) = \frac{N^2}{2\sum_{m=1}^{N-1} c_{m^2}} \tag{3}$$

3.2 Binary Sequence – Element Flipping

There are three steps involved in achieving the merit factors. The first step is to generate the Modified Jacobi sequence. The last step is to apply the advanced search algorithms to modified Jacobi sequences to achieve the merit factors. The whole procedure will be carried out in sequential steps to reduce the execution time. An intermediate step is present which involves single element or multi-element flips. The element flips boosts the performance of search algorithms and reduces the execution time of search process.

3.2.1 One – Element Flips

In single element flips, we flip one element at a time. We repeat the process for N elements in the sequence. The sequence with the element flip that gives the least energy will be the required solution. We will take the generated modified Jacobi sequence and perform single element flips on that sequence. The energies of all the flipped sequences will be noted. Then we will calculate the difference between primary sequence energy and current flipped sequence energy. After that, we will compare all the differences to see which flip got the least energy. Let us say that with one element flip the difference in energy is δ_1 .For all the flips, the difference energies will be saved in the vector Δ.

So $\Delta = [\delta_1, \delta_2, \delta_3, \delta_4 \ldots, \delta_N]$. Here, δ_j represents the complete difference in auto-correlation energy between primary and flipped sequence (produced by flipping element j). We compare all the values in Δ. While doing this lot of time wastage would be there. So instead of computing Δ in this procedure we have designed a better approach, by expressing Δ in its correlation and autocorrelation terms and applying Fast Fourier transform techniques (FFT) to it.

The total energy of the binary sequence is given by

$$E = 2\sum_{m=1}^{N-1} c_m^2 \tag{4}$$

When single element flip is performed on the sequence, the change in energy of the sequence is given by

$$\delta_j = 2\sum_{m=1}^{N-1} d_m^2 - E \tag{5}$$

Here $D=[d_1, d_2 \ldots d_N]$ are the auto-correlation energies of the flipped sequence A. The auto-correlation side-lobe energies differ whenever $a_i$or a_{i+m}in (A.1) change. That is when i=j or i+m=j

$$d_m = \sum_{i=1}^{N} (a_i\, a_{i+m}) - 2a_j a_{j+m} - 2a_{j-m} a_j \tag{6}$$

We can further simplify equation 6

$$d_m = c_m - 2a_j[a_{j+m} + a_{j-m}] \tag{7}$$

$$S_{j,m} = a_{j+m} + a_{j-m} \tag{8}$$

$$P_{j,m} = a_{j+m} a_{j-m} \tag{9}$$

From equation 8 and 9, the modified energies will be

$$d_m = c_m - 2a_j S_{j,m} \tag{10}$$

And the side lobe autocorrelation energy m is given by

$$d_m^2 = c_m^2 - 4a_j S_{j,m} c_m + 4a_j^2 S_{j,m}^2 \quad (11)$$

The total change in energy of autocorrelation side lobes with an element flip j is given by

$$\delta_j = \sum_{m=1}^{N-1}(-4ajSj, mcm + 4a_j^2 S_{j,m}^2) \quad (12)$$

$$= \sum_{m=1}^{N-1} (-a_j S_{j,m} c_m + S_{j,m}^2) \quad (13)$$

The above procedure will take a lot of time to compute the δj for all values of j. We have to simplify this process to execute the steps in less time. We have to remember that

$$S_{j,m}^2 = a_{j+m}^2 + a_{j-m}^2 + 2P_{j,m} \quad (14)$$

So from 13 we get

$$\delta_j = \sum_{m=1}^{N-1}(-ajSj, mcm + 2P_{j,m}) + \sum_{m=1}^{N-1}(a_{j+m}^2 + a_{j-m}^2) \quad (15)$$

$$= \sum_{m=1}^{N-1}(-ajSj, mcm + 2P_{j,m}) + 8(N-1) \quad (16)$$

Now we can rewrite the change in energy as

$$\delta_j = 8a_j \sum_{m=1}^{N-1} Sj, mcm + 8\sum_{m=-N+1}^{N-1} P_{j,m} + 8(N-2) \quad (17)$$

Observe the equation 17 keenly. We can relate this equation to correlations pair.

$$= \sum_{m=1}^{N-1}(Cma_{j+m} + Cma_{N+1-j+m}^r) \quad (19)$$

Here r indicates the reverse of the corresponding sequence. Equation (19) is now in the form of cross-correlation (2)

$$\sum_{m=1}^{N-1} Sj, mcm = (C * A)j + (C * Ar)N - j + 1 \quad (20)$$

Here the alphabet C represents the autocorrelation sidelobe energies of A*A. The second term in (7) is also in the form of correlation

$$\sum_{m=-N+1}^{N-1} P_{j,m} = \sum_{m=-N+1}^{N-1} a_{j+m} a_{j-m} \quad (21)$$

$$= \sum_{m=-N+1+j}^{N-1+j} am a_{N+1-2j+m}^r \quad (22)$$

We already know that $1 \le am \le N$, so we have to change the limits of the summation to make the sum congruent to cross-correlation (2).We have to do this in such a way that it will not affect the aggregate value

$$\sum_{m=-N+1}^{N-1} P_{j,m} = \sum_{m=1}^{N} am a_{N+1-2j+m}^r \quad (23)$$

$$= (A*A^r)_{N+1-2j} \quad (24)$$

We have to mix the two summations to get the δ_j value

$$\delta_j = -8a_j(C*A)_j + (C*A^r)_{N-j+1} + 8(A*A^r)_{N+1-2j} + 8(N-2) \quad (25)$$

If we want to compute the above equation directly, lot of time is required. To simplify the process, we make use of Fourier transform. FFT is a perfect way to compute this sum.

$$FFT(A*B) = FFT(A)\ FFT(B^r) \quad (26)$$

When we apply inverse Fourier transform to above equation we get

$$A*B = F^{-1}(FFT(A)\ FFT(B^r)) \quad (27)$$

Here F^{-1} indicates the inverse Fourier transform function.

Equation (25) consists of multiple correlation pairs. So the Fourier transform implementation involves multiple FFTS. The minimum length of each FFT is 2N-1.If N is of smaller length, equation (13) is a better approach than (27).This change will not create any problems in our calculations because in our experiments we only consider very long sequences.

3.2.2 Multi – Element Flips

So far, we have explained the mechanism of single element flips calculations. Now we want to explain multiple elements flip calculations. In multiple element flips, more than one element flips will be allowed to give the solution, and the whole process repeats for several iterations. We want to describe the procedure for calculations here

Assume that s be a set that consists of all element flips. Let f be another set that consists of all the distances between the flipped elements. Let g_m be the pair wise sum of all the products of all flipped elements. The indices all these elements differ by m. The value of g_m will be zero when m∉f.

$$g_m = \sum_{j \in s, (j+m) \in s} a_j a_{j+m}, \text{ if } (m \in f) \quad (28)$$

$g_m = 0$, otherwise

Now the changed autocorrelation side lobe energies will be described as

$$d_m = c_m - 2\sum_{n \in S} a_n S_{n,m} + 4g_m \quad (29)$$

We can rewrite Equation (29) as

$$d_m^2 = -4c_m \sum_{n \in S} a_n S_{n,m} + 4\left(\sum_{n \in S} a_n S_{n,m}\right)^2 + 8g_m c_m - 16g_m \sum_{n \in S} a_n S_{n,m} + 16g_m^2 \quad (30)$$

The total energy of side lobes is given by

$$\delta_S = 2\sum_{m=1}^{N-1}(d_m^2 - c_m^2) \quad (31)$$

$$= 2\sum_{m=1}^{N-1} n - 4c_m \sum_{m \in S} a_n S_{n,m} + 4\left(\sum_{n \in S} a_n S_{n,m}\right) * \left(\sum_{n \in S} a_n S_{n,m}\right) a + 2\sum_{m \in f}(8g_m c_m 16g_m \sum_{n \in S} a_n S_{n,m} 16g_m^2) \quad (32)$$

$$\delta_S = \sum_{n \in S} \delta_n + 8\sum_{m=1}^{N-1}\sum_{n \in S}\sum_{p \in S, p \neq n} a_n a_p S_{n,m} S_{p,m} + 16\sum_{m \in f}(g_m c_m - 2g_m \sum_{n \in S} a_n S_{n,m} + 2g_m^2) \quad (33)$$

Some part of this equation replicates single element changes. We will use (13) to transform this equation. So we get

We will arrange the triple summations according the requirements of correlations structures.

$$\delta_S = \sum_{n \in S} \delta_n + 8\sum_{n \in S}\sum_{p \in S, p \neq n} a_n a_p \sum_{m=1}^{N-1} S_{n,m} S_{p,m} + 16\sum_{m \in f}(\delta_m c_m - 2g_m \sum_{n \in S} a_n S_{n,m} + 2g_m^2) \quad (34)$$

Now we have to expand the term Sn,mSp,m

$$\sum_{m=1}^{N-1} S_{n,m} S_{p,m} = \sum_{m=1}^{N-1}(a_{n+m} + a_{n-m})(a_{p+m} + a_{p-m}) \quad (35)$$

$$= \sum_{m=1}^{N-1}(a_{n+m}a_{p+m} + a_{n+m}a_{p-m} + a_{n-m}a_{p+m} + a_{n-m}a_{p-m}) \quad (36)$$

$$= \sum_{m=-N+1}^{N-1}(a_{n+m}a_{p+m} + a_{n+m}a_{p-m} - 2a_n a_p) \quad (37)$$

$$= \sum_{m=-N+1+n}^{N-1+n} (a_m a_{p-n+m}) + \sum_{m=-N+1}^{N-1} (a_{n+m} a^r_{N+1-p+m} - 2a_n a_p) \quad (38)$$

$$= c_{p-n} + (A * A^r)_{N+1-p-n} - 2a_n a_p \quad (39)$$

$$\delta_s = \sum_{n\in S} \delta_n + 8 \sum_{n\in s} \sum_{p\in s, p\neq n} (a_n a_p c_{n-p} - 2) + 8 \sum_{n\in s} \sum_{p\in s, p\neq n} a_n a_p (A * A^r)_{N+1-p-n} + 16 \sum_{m\in f} \left(g_m c_m - 2g_m \sum_{n\in s} a_n S_{n,m} + 2 g_m^2 \right) \quad (40)$$

$$8 \sum_{n\in s} \sum_{p\in s, p\neq n} a_n a_p c_{n-p} = \sum_{m\in f} 16\, g_m c_m \quad (41)$$

$$8 \sum_{n\in s} \sum_{p\in s, p\neq n} -2 = -16 N_s (N_s - 1) \quad (42)$$

$$\delta_s = \sum_{n\in s} \delta_n - 16 N_s (N_s - 1) + 8 \sum_{n\in s} \sum_{p\in s, p\neq n} a_n a_p (A * A^r)_{N+1-p-n} + 32 \sum_{n\in s} \delta_n - 16 N_s (N_s - 1) 16 \sum_{n\in s} \sum_{p\in s, p>n} a_n a_p (A * A^r)_{N+1-p-n} + 32 \sum_{m\in f} \left(g_m c_m - g_m \sum_{n\in s} a_n S_{n,m} + 2 g_m^2 \right) \quad (43)$$

Here NS represents the number of flipped elements. We will use 41 and 42 on 40 to give

The speed of (43) depends on the number of flipped elements. If Δ is calculated first then δ_S does not depend upon the length of the sequence N. The time required to implement 43 is very high. So in order to execute the calculation faster 43 is approximated by the following equation

$$\approx -32 \sum_{m\in f} \left(g_m \sum_{n\in s} a_n S_{n,m} \right) \approx -32\, N_s^2 \quad (44)$$

If the N value increases, the accuracy of (A.44) decreases. In our experience, our inaccuracies have been minimum.

3.2.3 Reduction of Delta Complexity

Our algorithms implement an iterative method in which single or multi-element flips will occur. Once the flip occurs, the Δ will become useless. We should calculate new Δ for every iteration, which makes the program execution very complicated. We found a method to overcome this problem.

We will flip an element k in the sequence. Indicate the modified sequence by a dot. The modified sequence is $\dot{A}$

In the similar notation we can write $\dot{\Delta} = [\dot{\delta}_1, \dot{\delta}_2, \ldots \dot{\delta}_N]$. $\dot{\Delta}$ is the new vector for the changed energy. When j = k,

$$\dot{a}_i = \begin{cases} -a_i, & i = k \\ a_i, & \text{otherwise} \end{cases} \quad (45)$$

we get $\dot{\delta_j} = -\delta_k$. When $j \neq k$, we get $\delta_{j,k} = \delta_k + \delta_j$. When we apply this change in (A.43) we get

$$\dot{\delta_j} = \delta_{j,k} - \delta_k = \delta_j + 16 a_j a_k \left((A * A^r)_{N+1-j-k} + 2c_{|k-j|} \right) - 32 a_j a_{2j-k} - 32 a_k a_{2k-j} - 64 \quad (46)$$

Equation 46 represents the autocorrelation $A*A^r$ and the side lobe energies C. If we assume that Δ is calculated earlier using (25), then A and A^r need not be calculated again. It is done already at the time of (25). When m = N + 1 - 2k, the modified elements in A and A^r overlap. So $A*A^r$ will not have any change.

$$A * A^r = \sum_{i=1}^{N} a_i a_{N-i+1-m} - 4a_k a_{N+1-m-k} = (A * A^r)_m - 4a_k a_{N+1-m-k} \quad (47)$$

We can calculate side lobe energies without any problem. They are given by

$$\dot{c}_m = c_m - 2a_k (a_{k+m} + a_{k-m}) \quad (48)$$

With the help of (46), we reduced the complexity of Δ calculation. Thus, it will update Δ value continuously

3.3 Construction of Modified Jacobi Sequences

In order to understand Jacobi and modified symbols, we should first analyze the Legendre symbol.

3.3.1 Modified Jacobi Sequences

Twin prime Jacobi sequences do not create good autocorrelation function values, which are out of phase to p and q. In order to overcome this limitation, we go for Modified Jacobi sequences. Out of phase autocorrelation values depend on difference k between p and q. The modified Jacobi sequence can be defined as

$$\boldsymbol{j_a} = \begin{cases} \left(\frac{a}{p}\right) \oplus \left(\frac{a}{q}\right) \; for\; (a, N) = 1, 0 \le a \le N \\ \mathbf{0} \text{ fora} = 0 \bmod q \\ \mathbf{1} \text{ Otherwise} \end{cases} \quad (49)$$

We force $\boldsymbol{j_a}$ of the normal Jacobi sequence to 0 for all a, which are multiples of q and should be 1 for all a, other than a=0, which are the multiples of p. The values of a which are multiples of q($a \equiv 0 \bmod q$) will be present on the first column of the array. The values of a which are multiples of p($a \equiv 0 \bmod p$) will be present on the first row of the array. To perform autocorrelation, the first column should hold all zeros and the first row should contain all ones. The implementation for this process is shown below for a length of 35

Consider our old pq =35 length example, it is demonstrated below in table

3.4 Advanced Algorithms

We have developed two algorithms for Modified Jacobi sequences. We achieved satisfactory results with these algorithms. The two algorithms are Prime step algorithm and steepest descent algorithm.

3.4.1 Prime step algorithm

It is a non-iterative algorithm, and the computation is very simple. The algorithm improves the merit factor by flipping all the elements in the isolated sequence. We will consider only those sequences that have a negative Δ. Consider a new sequence D produced by inverting all the elements in sequence A which has a negative Δ. The formula describes the prime step algorithm.

The prime step algorithm first calculates Equation (25) and then applies (55) to it with no iterations.

$$d_j = \begin{cases} a_j & \delta_j \geq 0 \\ -a_j \delta_j & < 0 \end{cases} \quad (50)$$

3.4.2 Steep descent algorithm (modified version of steepest descent algorithm)

When there is no analytic procedure to obtain the minimum value of a given function, we will go for iterative technique for achieving approximate solution. Newton has defined a method to achieve the minimal value, but it is not reliable. To solve these kinds of problems, we go to steepest descent method. The first step in the algorithm is to calculate the gradient of the given function at a given point. In order find the local optimum, we should proceed stepwise towards negative gradient of given function. We will begin at a place x0 and slowly we will proceed from x_i to x_{i+1} by taking minimal values along the straight line from xl towards the course of –ve gradient.

1	0	1	1	0	1	0	1	1	0	1	0	1	1	0	1	0	1	1	0	1	0	1	1	0	1	0	1	1	0	1	0	1	1	0
1	0	0	1	0	1	1	1	0	0	1	0	1	1	1	0	0	1	0	1	1	1	0	0	1	0	1	1	1	0	0	1	0	1	1
1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	1	1	0	1	0	1	0	0	0	0	1	0	0	1	1	1	0	1	1	1	1	1	0	0	0	1	1	1	0	1

Consider an example: If we apply this algorithm to a random function f(x), the structure of the iteration will be

Consider an example: If we apply this algorithm to a random function f(x), the structure of the iteration will be

$$x_i = x_{i-1} - \varepsilon f'(x_{i-1}) \quad (65)$$

We applied steepest descent algorithm to LABS problem. Results were satisfactory, but still we tried to improve them. For very long sequences, the inability of steepest descent method to create multiple flips in a single iteration became a limitation. We were not able to get the merit factors in quick time. The algorithm flips the elements one at a time, and the flip that gives the highest merit factor will be the solution. Without multiple element flips and a better iterative procedure, it is almost impossible to say that this method has the best approach. The steepest descent algorithm is very simple, but it has its limitations when applied to long Jacobi sequences

We modified the steepest descent algorithm and named it as steep descent algorithm. This algorithm allows multiple element flips in each iteration. So the number of iterations will be less, which improves the speed of the algorithm. In every iteration, all the elements in the sequence with negative Δ value will be flipped. If the merit factor value improves with this iteration, Δ is again calculated for the new sequence, and the iterations will go on. If the value of the merit factor is not increasing then, we will consider the element subsets with – Δ (subsets range from 10%,20% ….so on).Algorithm selects the subset that provides the best merit factor implements it, and the process of iteration will continue. When additional flips do not produce any improvement in merit factors, the iteration procedure will terminate. This termination will also occur when large number of iterations is over.

We have tried several variations of steep descent method. We varied the number of element flips in each iteration, and we observed the results. For almost all the variations, the merit factor values were identical. The changes are not discussed here as they are not important. It became apparent to us that the steep descent algorithm is far more efficient than steepest descent algorithm. We named, both the prime step algorithm and steep descent algorithm as modified Jacobi algorithms because with the application of these algorithms the merit factors of Jacobi sequences has reached above 6.4.

4. RESULTS

To get the results, we applied Modified Jacobi sequences to both of the enhanced algorithms. We got significant improvements in merit factors for various sequences. We conclude that, on a relative basis, steep descent algorithm performed well than prime step algorithm. In this part, first we will discuss the Prime step results and then the Steep descent results. Later, we will compare the results of Steep descent and prime Step algorithms using two statistical methods. Finally, we will discuss the highest merit factors obtained in our experiments.

4.1 Prime Step Algorithm Results

There was a slight improvement in merit factor values when we applied the prime step algorithm to Modified Jacobi sequences. We have selected Modified Jacobi sequences because even without the application of prime step algorithm, they are capable of producing good results by themselves. Once we have applied the algorithm, it further improved merit factor values.

Table 1

Sequence Length	**Modified Jacobi - Merit factor**	**Prime step - Merit factor**
9797	6.3321	6.4482
194477	6.3411	6.4466
205193	6.345	6.4423
390589	6.3429	6.4362
471953	6.3421	6.4419
557993	6.3412	6.4427
644773	6.3419	6.4391
741317	6.3421	6.44
804509	6.3404	6.4383

974153	6.3443	6.4349
1005973	6.3416	6.4394

Table 1 shows the merit factor values for some of the Modified Jacobi sequences. The generated sequences tend to a merit factor range of 6.3(approximately).When we applied prime step algorithm to it the merit factor range improved significantly to 6.4.

We have conducted experiments for elements up to a length of one million. We have observed the same phenomenon in all these sequences. We tried to represent all those merit factors in a graph, but the graph lacks clarity. So we sampled some of the best sequences from our results and represented them in a graph(Figure 1). So for a length of one million we got sampled sequences that would efficiently determine the performance of our results.

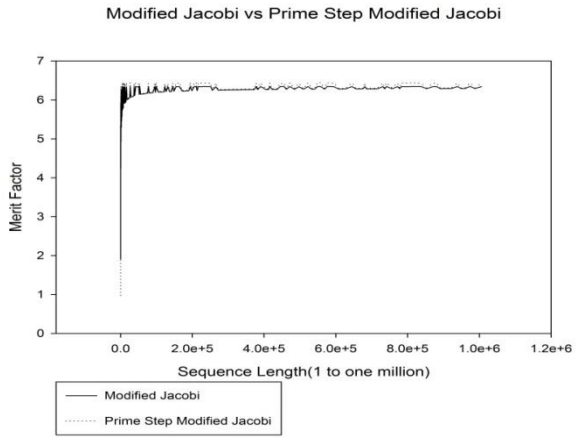

Figure 1

4.2 Steep Descent Algorithm Results

So far in our experiments this algorithm provided the best results. In single step algorithm case, there was a slight improvement in merit factor range. But in steep descent case, we achieved our best merit factors.

Table 2

Sequence Length	modified Jacobi - Merit factor	Steep descent- Merit Factor
9797	6.3321	6.4534
194477	6.3411	6.4491
205193	6.345	6.4446
390589	6.3429	6.4384
471953	6.3421	6.444
557993	6.3412	6.446
644773	6.3419	6.4418
741317	6.3421	6.4434
804509	6.3404	6.4413
974153	6.3443	6.4373
1005973	6.3416	6.4415

Similar to Prime step algorithm case, we want to compare the results of Modified Jacobi sequences with Steep descent Modified Jacobi sequences .Figure 2 describes an exact representation of that. We can clearly observe the domination of steep descent merit factors over Modified Jacobi merit factors. In Table 2 we have represented some of our steep descents Modified Jacobi results. Our merit factor reached around 6.4534.

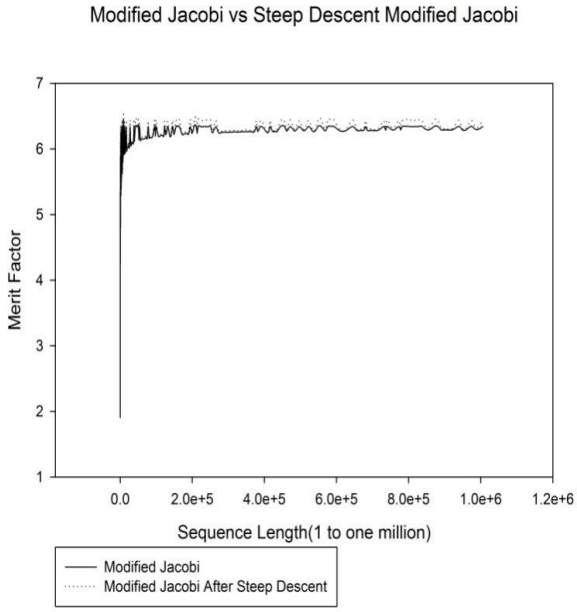

Figure 2

4.3 Steep Descent VS Prime Step Results

The results indicate that the steep descent results dominate single step results for almost all the sequences. We compared the two results in Figure 3.

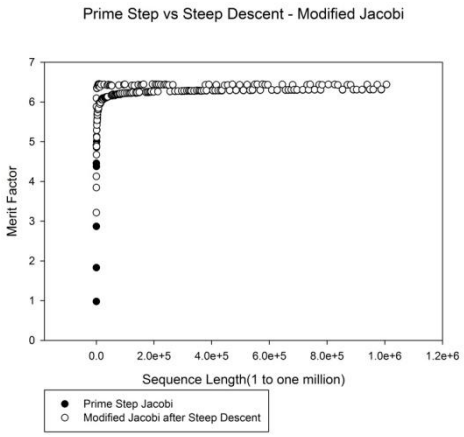

Figure 3

The graph lacks clarity, so we need an alternate method which exactly demonstrates the difference in results between the two algorithms.

We want to represent the difference between the two algorithms statistically, so we used two statistical methods to compare the results.

1. Mann-Whitney Rank Sum test
2. Kruskal-Wallis one-way Analysis of Variance on Ranks

4.3.1 Mann-Whitney Rank Sum test

We compared the results of Prime step Modified Jacobi and Steep descent Modified Jacobi algorithms using this tool. This statistical test ranks the merit factors for each and every condition between two groups (in our case Prime step and Steep descent) and then it will analyze how different the two rank sums are. If there exists a systematic change between the two selected groups, then most of the high ranks will represent one group(Steep descent).Then most of the low ranks will represent another group(Prime step).Due to this, the rank totals will change for each cluster. The statistic "U" represents the difference between the two ranks. We can clearly observe the difference in median value, and the 25% and the 75% values.

Mann-Whitney U Statistic = 13243.000

Table 3

GROUP	Number of samples	Median	25%	75%
Prime step modified Jacobi	167	6.290	6.190	6.428
Steep Descent modified Jacobi	167	6.296	6.203	6.430

With test results in Table 3, we can conclude that that the steep descent algorithm performs well than Prime step algorithms in almost all the instances.

4.3.2 Kruskal-Wallis one-way Analysis of Variance on Ranks

We compared the results of Prime step Modified Jacobi, Steep descent Modified Jacobi algorithms with original modified Jacobi sequence using this tool. It is an extension of Mann-Whitney Rank Sum test for three different groups. We can conclude from the results that prime step and steepest descent algorithms performed well over modified Jacobi sequences.

Table 4

GROUP	Number of samples	Median	25%	75%
Modified Jacobi	167	6.270	6.175	6.340
Prime step modified Jacobi	167	6.290	6.190	6.428
Steep Descent modified Jacobi	167	6.296	6.203	6.430

Table 5

Comparison between	Difference of ranks	q	P<0.05
Jacobi vs. steep descent modified Jacobi	9477.000	5.066	yes
Prime step vs steep descent modified Jacobi	1759.500	0.940	no
Modified Jacobi vs. Prime step Jacobi	7717.500	4.125	yes

In Table 5,"Difference of ranks is very important. High-rank differences indicate that the compared second group has performed exceedingly well over the first group

4.4 Best Merit Factors

In table 6, we include our top ten merit factors with their corresponding sequence lengths. In figure 4 we compared the top ten merit factors of Modified Jacobi, steep descent and prime step algorithms. Some of the bubbles overlap, so we cannot see all the ten results.

Table 6

Sequence length	Modified Jacobi merit factors	Prime step merit factors	Steep descent merit factors
9797	6.3321	6.4482	6.4534
194477	6.3411	6.4466	6.4491
6557	6.3322	6.4434	6.4467
557993	6.3412	6.4427	6.446
164009	6.3414	6.4423	6.4451
205193	6.345	6.4423	6.4446
576077	6.3424	6.4423	6.4443
471953	6.3421	6.4419	6.444
741317	6.3421	6.44	6.4434
239117	6.3406	6.4404	6.4433

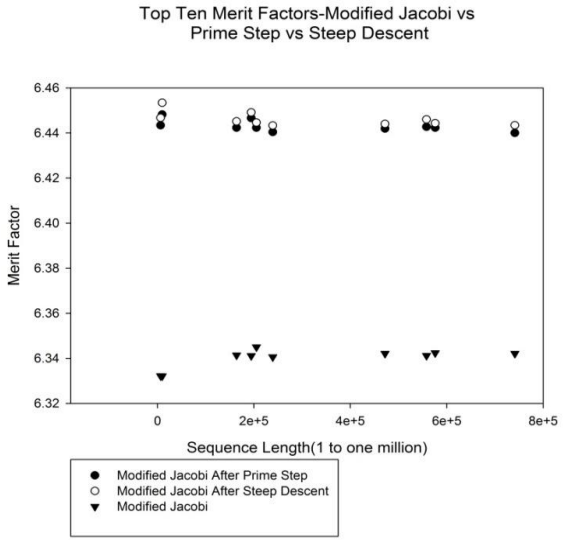

Figure 4

5. CONCLUSION

D.H. Green [1] constructed Modified Jacobi sequences to obtain a merit factor value of 5.99.To improve the results; these sequences were applied to Prime step and steep descent algorithms. There is a significant improvement in merit factor range. A highest merit factor value of 6.4534 was achieved. The merit factors of Modified Jacobi sequences were calculated up to a length of one million to determine the asymptotic behavior of merit factors for long length binary sequences, generated using modified Jacobi symbol.

The two major limitations for LABS research are long execution times and low computational power. .Efforts was made to simplify these limitations. As the computation power increases, the generation of LABS becomes easier. If the computational power permits the work, experiments will be conducted for more sequences. In future, we want to increase the sequence length to a value of ten millions to observe the behavior of these sequences under the influence of our enhanced algorithms.

6. REFERENCES

[1] D.H. Green and P.R. Green, " Modified Jacobi Sequences", IEE Proc-comput.Digit-tech ,Vol. 147,No 4,July 2000.

[2] M. J. E. Golay, "The merit factor of Legendre sequences," IEEE Trans.Inf. Theory, vol. IT-29, no. 6, pp. 934–936, Nov. 1983.

[3] A. Kirilusha and G. Narayanaswamy, "Construction of New Asymptotic Classes of Binary Sequences based on Existing Asymptotic Classes," Tech. Rep. Dept. Math. Comput. Sci., Univ. of Richmond, Richmond, VA, 1999.

[4] M. J. E. Golay, The merit factor of long low autocorrelation binary sequences.,IEEE Transactions on Information Theory 28 (3) (1982) 543–549.

[5] S. Mertens, Exhaustive search for low-autocorrelation binary sequences, Journalof Physics A: Mathematical and General 29 (1996) 473–481.

[6] S.Mertens,H.Bauke,Ground statesof the Bernasconi model with open boundary conditions,website available at http://www-e.unmagdeburg.de/mertens/research/labs/open.dat (accessed January2007).

[7] S. Prestwich, A hybrid local search for low autocorrelation binary sequences,Technical Report TR-00-01, Department of Computer science, National University of Ireland, Cork, Ireland (2000).

[8] P. Borwein, K.-K. S. Choi, and J. Jedwab, "Binary sequences with merit factor greater than 6.34," IEEE Trans. Inf. Theory, vol. 50, no. 12, pp. 3234–3249, Dec. 2004.

[9] R.A Kristiansen and M.G.Parker, "Binary Sequences Withmerit factor ≥ 6.3",IEEE Trans.Theory,vol.50,no,12,pp,3385-3389,Dec.2004.

[10] AbhisekUkil, "Low autocorrelation binary sequences Number theory based analysis for minimum energy level, Barker codes".

[11] B. Militzer, M. Zamparelli, D. Beule, Evolutionary search for low auto correlated binary sequences", IEEE Transactions on Evolutionary Computation 2 (1) (1998) 34-39.

[12] J. W. Moon and L. Moser, "On the correlation function of random binary sequences," SIAM J. Appl. Math., vol. 16, pp. 340–343, 1968.

[13] G. E. Coxson and J. Russo, "Efficient exhaustive search for optimalpeak-sidelobe binary codes,"IEEE Trans. Aerospace and Electron. Syst.,vol. 41, pp. 302–308, 2005.

[14] S. Prestwich, A hybrid local search for low autocorrelation binary sequences, Technical Report TR-00-01, Department of Computerscience, National University of Ireland, Cork, Ireland (2000).

[15] P. Moscato, 'Memetic algorithms: A short introduction," in: D. Corne, M. Dorigo, F. Glover (Eds.), New Ideas in Optimization, McGraw-Hill, Maidenhead, Berkshire, England, UK, 1999, pp. 219-234.

[16] R. N. Bracewell, The Fourier Transform and its Applications, 2nd ed.New York: McGraw-Hill, 1986.