Advances in Computational Research
ISSN: 0975-3273 & E-ISSN: 0975-9085, Volume 6, Issue 1, 2014, pp.-153-158.
Available online at http://www.bioinfopublication.org/jouarchive.php?opt=&jouid=BPJ0000187

NEW SIMULATED ANNEALING ALGORITHM FOR COMPUTING THE MINIMUM DISTANCE OF LINEAR BLOCK CODES

AYLAJ B.[1]* AND BELKASMI M.[2]

[1]Department of Mats, MMID, Faculty of Science El jadida, Chouaib Doukkali University, Morocco.
[2]SIME Labo, National School of Computer Science and Systems Analysis (ENSIAS), Rabat, Mohammed V-Souisi University, Morocco.
*Corresponding Author: Email- bouchaib_aylaj@yahoo.fr

Received: August 02, 2014; Accepted: September 01, 2014

Abstract- In this paper we propose a new idea of the use of simulated annealing method to estimate the good value of the minimum distance of linear block codes which remains an open problem and cannot be solved by classical methods because it is in general a NP-hard problem. The main points of the idea is to define in the first hand a new mechanism of moving the search in different regions of solution space by using so called degeneration of energy, notion comes from statistical physics for particles, and in the second hand a new perturbations scheme and acceptance rules to search nearby neighborhoods. To illustrate quality of the proposed approach, we validate the algorithm for some linear codes which the minimum distance is known, and we present the computational experiment results compared to those of previous work which used classic Simulated Annealing algorithm, and those using heuristic techniques such as genetic algorithms and local search algorithms.

Keywords- Minimum Distance, Simulated Annealing, Proposed Simulated Annealing, Degenerated States System (DSS), Non Degenerated States System (NDSS), Perturbation, Linear Error Correcting Codes, BCH Codes, QR Codes

Citation: Aylaj B. and Belkasmi M. (2014) New Simulated Annealing Algorithm for Computing the Minimum Distance of Linear Block Codes. Advances in Computational Research, ISSN: 0975-3273 & E-ISSN: 0975-9085, Volume 6, Issue 1, pp.-153-158.

Introduction

The idea of Simulated Annealing (SA), is Based on statistical mechanics reasoning, applied to a solidification problem, Metropolis, et al [1] introduced a simple algorithm that can be used to accomplish an efficient simulation of a system of atoms in equilibrium at a given temperature. The concepts of annealing in combinatorial optimization are introduced in the early 1983's by Kirkpatrick, et al [2], these concepts are based on a strong analogy between the idea of the Metropolis algorithm applied to a physical annealing process of solids and the problem of solving large combinatorial optimization problems to search for feasible solutions and converge to an optimal solution.

SA method is studied extensively and has many successful solutions [3-6]. It is a non-numerical algorithm which is simple and globally optimal, mathematically SA has been proved that it is possible to converge to globally solution [7], and SA is suited well for solving the large-scale combinatorial optimization problems and its application [8] to diverse areas are in progress used in VLSI design, image processing, molecular physics, job shop scheduling, event based learning situations, strategy scheduling for capital products with complex product structure, umpire scheduling in US open tennis tournament and jigsaw puzzle to name a few . In coding theory, El Gamal, et al [9] used SA to the construction of good source codes, constant weight error-correcting codes, and spherical codes for certain sets of parameters codes that are better than any other known previously have been found. Zhang, et al [10] applied SA to computer the minimum distance of linear block codes, computer simulations indicate that SA is useful in providing good upper bounds to the minimum distance of general linear block codes.

In this paper we present a method to find a good estimate of minimum distance of linear block codes using a new idea of the use of simulated annealing, with this method we can approach the lower bound of the minimum distance which can be achieved by linear block codes especially Bose-Chaudhuri-Hocquenghem (BCH) that are characterized by their designed minimum distance: ***d*-design** (see chapter 7. Section 6 in [23]) and Quadratic Residue (QR) codes [11,12].

The minimum distance of a linear block code is an important parameter as it determines the error-correcting and detecting capabilities of the code. But it's not easy to estimate its true value by classical methods because it is a NP-hard problem [13], therefore the problem has been attacked in the literature with heuristic methods such as genetic algorithms [14-17], and search local error using a Soft-In decoder when applied to the problem of determining the true minimum distance of a linear block code [18], large excellent works have found the minimum distance in [19] Walis, et al have presented a different genetics techniques applied to find an estimate of the minimum distance for some (BCH) codes, in [20] Askali, et al have computed the minimum distance of linear block codes by heuristic methods, in [22] Ajitha, et al have studied the performance of the Metropolis algorithm for the problem of finding the minimum weight code.

Minimum Distance and Encoding of Linear Block Codes

Minimum Distance

Let *C(n,k)* denote a linear block code of length *n* and dimension *k*, each element $V \in C$ is called a codeword there are a total of 2^k codeword which is a k-dimensional subspace of the vector space $GF(2)^n$, where the modulo-2 sum of two codewords is also a codeword. The Hamming weight $W_H(V)$ is the number of nonzero components of *V* and the Hamming distance $d(V_1,V_2)$ between two codewords V_1 and V_2 is the number of places in which V_1 and V_2 differ. The minimum Hamming distance d_{min} (or the minimum distance) of the code *C* is the Hamming distance between the pair of codewords with smallest Hamming distance. That is

$$d_{min} = \min_{V_i,V_j \in C} d(V_i,V_j) \quad (1)$$

It can easily be proved that the minimum distance of the linear block code *C* is also equal to the minimum Hamming weight of non zero codewords is written as follows

$$d_{min} = \min_{\substack{V \in C \\ V \neq 0}} W_H(V) \quad (2)$$

Encoding for a Linear Block Codes

The encoder transforms each input information vector *U* into n-tuple codeword *V*, with $n > k$, [Fig-1]

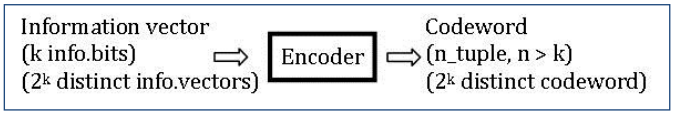

Fig. 1- A simplified Encoder representation

The General Case

Let $G = (g_{ij})_{kxn}$ be the generator matrix of the code *C*, whose rows constitute a set of basis vectors for subspace. Then every codeword $V = (v_1, v_{2,...,} v_n)$ can be uniquely represented as a linear combination of the rows of *G*.

$$\forall \ 1 \le j \le n \qquad v_j = \bigoplus_{i=1}^{k} u_i g_{ij} \quad (3)$$

Where $\oplus$ denotes modulo-2 sum, $U= (u_1, u_{2,...,} u_k) \in \{0,1\}^k$ information vector. Substituting [Eq-3] for [Eq-2] and noting the fact that

$$W_h(V) = \sum_{j=1}^{n} v_j$$

We have

$$d_{min} = \min_{\substack{U \in \{0,1\}^k \\ U \neq 0}} \sum_{j=1}^{n} \left(\bigoplus_{i=1}^{k} u_i g_{ij} \right) \quad (4)$$

Let

$$E(V) = \sum_{j=1}^{n} \left(\bigoplus_{i=1}^{k} u_i g_{ij} \right) \ where \ U(u_1,...,u_k) \in \{0,1\}^k - \{0\} \quad (5)$$

The Case where the Linear Block Code is Cyclic

In case where the linear block code *C*(*n*,*k*) is also a cyclic code, the encoding operation can be efficiently using simple shift registers and combinatorial logic elements, on the basis of their polynomial representation.

Let *P*(*x*) generator polynomial of degree *(n-k)* of the linear cyclic block code *C*

To every codeword vector *V* a polynomial *V(x)* is associated

$$V = (v_0, v_1,..., v_{n-1}) \ \rightarrow \ V(x) = v_0 + v_1 x + ... + v_{n-1} x^{n-1} \quad (6)$$

And the generator polynomial of *C* is *p(x)*:

$$P = (p_0, p_1,..., p_{n-k}) \ \rightarrow \ P(x) = p_0 + p_1 x + ... + p_{n-k-1} x^{n-k} \quad (7)$$

The information vector *U* can be written by the polynomial representation as follows:

$$U = (u_0, u_1,..., u_{k-1}) \ \rightarrow \ U(x) = u_0 + u_1 x + ... + u_{k-1} x^{k-1} \quad (8)$$

$$Where \ v_i, \ u_j, p_i \ \in \{0,1\}$$

Encoding the codewords of a binary cyclic code can be either nonsystematic or systematic, depending on how the information is processed:

A Nonsystematic Encoding by Multiplying with *p(x)*

$$\forall V \in C(n,k) \rightarrow V(x) = U(x).P(x) \quad modulo \ x^n + 1 \quad (9)$$

$$\forall \ 0 \le j \le n-1 \qquad v_j = \bigoplus_{i=0}^{j} u_i p_{j-i} \quad (10)$$

Then

$$E(V) = \sum_{j=0}^{n-1} \left(\bigoplus_{i=0}^{j} u_i p_{j-i} \right) \quad where \ U(u_1,...,u_k) \in \{0,1\}^k - \{0\} \quad (11)$$

A Systematic Encoding by Dividing with *p(x)*

$$\forall V \in C(n,k) \rightarrow V(x) = x^{n-k} U(x) + \left[(x^{n-k} U(x)) mod p(x) \right] \quad (12)$$

$$\rightarrow V(x) = x^{n-k} U(x) + R(x) \quad (13)$$

$$Where \quad R(x) = r_0 + r_1 x + ... + r_d x^d \quad (14)$$
$$And \quad deg(R(x)) = d < n - k$$

$$Then \quad E(V) = \sum_{j=0}^{n-1} v_j \quad (15)$$

$$Where \quad v_j = \begin{cases} r_j & 0 \le j \le d \\ 0 & d < j < n-k \\ u_{j-n+k} & n-k \le j \le n-1 \end{cases} \quad (16)$$

$$And \ r_j \in \{0,1\}, \quad U(u_0,\ldots,u_{k-1}) \in \{0,1\}^k - \{0\}$$

Then the minimum distance d_{min} of the code *C* is the minimum value of the function *E(V)* developed in [Eq-5], [Eq-11] or [Eq-15] depending on the type of encoding used.

Simulated Annealing Algorithm

Simulated annealing is physically referred to the process of heating up a solid and then cooling slowly until it crystallizes. Atoms of this material have high energies at very high temperatures. This gives the atoms a great deal of freedom in their ability to restructure themselves. As the temperature is reduced the energy of these atoms decreases, until a state of minimum energy is achieved.

The analogy between Physical Annealing and SA (optimization problem) is shown in [Table-1] [21].

Beginning with a current atoms configuration, this configuration is equivalent to the current solution of an optimization problem. The energy of the atoms is analogous to the cost of the objective function and the final frozen state corresponds to the global minimum of the cost function. Using these mappings any combinatorial optimization problem can be converted into an annealing algorithm [2], especially in our case the estimation of the minimal distance of linear block codes, we present below SA algorithm [Algorithm-1].

Table 1- Analogy between simulated annealing and optimization problem

Physical Annealing	Optimization Problem
State	Feasible solution
Energy	Cost
Change of State	Neighboring Solutions (Perturbation)
Temperature	Control Parameter
Frozen state	Optimal solution

Procedure SA()

Initialize SA parameters: number of iterations (N_i), a reducing rate, starting (T_s) and final (T_f) temperatures and a random starting solution (s)

While (*temperature (T) > T_f***)**

{

while (*iteration (I) <* N_i**)**

{

get the neighborhood solution (s_n);

Evaluate dE = eval(s_n) – eval(s);

if dE ≤ *0* **then** *s←s_n;*

else if *random(0,1) ≤ Exp(-dE/T)* ***then*** *s←s_n;*

end if

end if

}

T←cool(T);

}

end procedure

Algorithm 1- The basic procedure of the SA.

In [10], the SA algorithm used for finding the good upper bounds of the minimum distance of linear codes, is similar to that given in the [Algorithm-1], ie a classical SA algorithm, such as that given by Kirkpatrick[2].

In [10] the authors defined the state by the information vector and the cost function by the hamming weight of the codeword. In the next section we will present a new SA algoritm to find good lower bounds for general linear block codes.

The Proposed Simulated Annealing Algorithm

If we conceptualize hamming weight of codewords as an atom states of energy, then the analogy between the lowest energy of the atom and the minimum of hamming weight of codewords is appearent.

In statistical physics for particles, it happens that an atom energy level had sublevels and then we talk about the degeneration of energy and the atom is called a degenerated atom.

Considering a degenerated atom, we model the energy levels as hamming weights of the information vector and each distribution of the nonzero components of this vector, which generates a unique codeword, as energy sublevel. We distinguish then two system of energy states : degenerated states system (DSS) (energy level : hamming weights of the information vector) and non degenerated states system (NDSS) (energy sublevel : hamming weights of codword).

For Example

Let H(7,4,3) the Hamming code and $U \in \{0,1\}^4$ information vector.

If the encoding operation is systematic then we have 5 Energy levels:

- Energy level= $W_H(U)=0$ with 1 energy sublevel (1codword).
- Energy level= $W_H(U)=1$ with 4 energy sublevels (4codwords).
- Energy level= $W_H(U)=2$ with 6 energy sublevels (6codwords).
- Energy level= $W_H(U)=3$ with 4 energy sublevels (4codwords).
- Energy level= $W_H(U)=4$ with 1 energy sublevel (1codword).

In this case SA is brought to search the global state of minimum energy between the two systems DSS and NDSS.

Having two systems allows us to search the best solution locally in the case of NDSS system ensured by the perturbation2 mechanism (see next subsection) and to make an external research in the case of DSS system ensured by the perturbation1 mechanism (see next subsection). Thus, the exploitation of the search space and the movements between its different regions became possible. Also, in each system, research is intensified by accepting all movements improvers and it is diversified by accepting the movements non improvers. Note that diversification decreases with the decrease in temperature throughout of the algorithm.

The Proposed Simulated Annealing (PSA) algorithm used for estimating the minimal distance of a linear block codes is outlined as follows [Algorithm-2]:

Procedure PSA ()

Define Initial Configuration

Random Starting State = U_0

Starting states system= *DSS*

Starting temperature= T_S

While (*T > T_f***)**

{

Calculate the number of iterations (N_i) associated to T_i;

while (*iteration (I) <* N_i**)**

{

if (states system== DSS**) then** *get the neighborhood state (U_{i+1}) from perturbation1;*

else *get the neighborhood state (U_{i+1}) from perturbation2;*

end if

Evaluate ΔE = E(U_{i+1}) – E(U_i);

if ΔE ≤ *0* **then** U_i←U_{i+1};

else if (*random(0,1) ≤ Exp(-ΔE/T))* ***then*** U_i←U_{i+1};

end if

end if

}

if (the criterion of transition is satisfied**) then** switch states system;

end if

T←cool(T);

}

end procedure

Algorithm 2- The basic procedure of the PSA.

At a certain temperature Ts, starting from a randomly selected state of the DSS system, here the state corresponds to the vector information (Hamming weight), we subject the information vector to the perturbation1 and then we check the cost function which is the function *E (V)* already established above. We iterate this process, keeping the temperature constant, until the equilibrium is reached for the current system; on other words, after a sufficient number of perturbations. When the equilibrium is reached, according to a criterion of transition we check the possibility of transition between DSS and NDSS systems and we decrease the temperature, before making a new series of perturbations on the current system.

Perturbation Schemes

The fact that we have two systems of states allows us to define two types of perturbations: Perturbation-1 and Perturbation-2.

The perturbation mechanism is an operation to create new state from the current state. In other words it is an operation to explore the neighborhood of the current state creating small changes in the current state. The perturbation mechanism is an important factor for the convergence of SA algorithm, a good choice usually gives a good quality solution (global or a better optimum).

The exploration of neighborhood states can be made as follow. A state U_i is defined as a vector information representing the hamming weights of level or sublevel energy in the search states space $S= \{0,1\}^k-\{0\}$. A new state U_{i+1} is generated using a switch vector $b_i = (b_{i1},..., b_{ik})$ or a circular permutation σ_L of order L to create a perturbation from the current state.

Perturbation-1

Let $U_i = (u_{i1},..., u_{ik}) \in S$ the current state where $u_{ij} \in \{0,1\}$ and $\beta_i = (\beta_{i1},..., \beta_{ik})$ where $\beta_{ij} \in \{0,1\}$ is a switch vector of the Hamming weight $W_H(b_i)$ which is between *1* and *k*; β_i is generated randomly. A neighbor state U_{i+1} is then produced from U_i according to the following rules:

$$U_{i+1} = U_i \oplus \beta_i \tag{17}$$

Where $\oplus$ denotes modulo-2 sum between the elements of U_i and b_i

The hamming weight $W_H(U_{i+1})$ must be between 1 and d_u, where d_u is an uperbound on the minimum distance of linear block code and $d_u<k$

$U_{i+1} \in S$

Perturbation-2

Let σ_L is the circular permutation of order *L*, an operator which shift a rank of elements to the right with *L* positions.

The current state U_i is perturbed to generate U_{i+1} according to the following rule:

We generate a random integer *L* between *1* and *(k-1)*, then we apply the circular permutation σ_L on U_i:

$$U_{i+1} = \sigma_L(U_i) \tag{18}$$

Starting and Final Temperature

The starting temperature T_s must be large enough to enable the algorithm to move off a local state. This is related to the acceptance probability P_H of a worst state that depends on temperature and magnitude of cost function E. In this context, the starting temperature was determined as follow:

$$P_H = exp(-\Delta E_{max}/T_s) \tag{19}$$

Where $\Delta E_{\max} \le (d_u + n - k)$ P_H should be high probability

Then

$$T_s \le -(d_u + n - k)/(lnP_H) \tag{20}$$

The final temperature T_f must be small enough to enable the algorithm not to move off a global state. A similar reasoning can be applied to determine the final temperature T_f as follow:

$$P_l = exp(-\Delta E_{min}/T_f) \tag{21}$$

Where $\Delta E_{min} \ge 1$, P_l *should* be small probability

$$T_f \ge -1/(lnP_l) \tag{22}$$

T_f is used as stop criterion for the algorithm during the simulation

Cooling Schedule

The cooling schedule was implemented using a geometric rule for temperature variation:

$$T_{i+1} = \alpha T_i \tag{23}$$

A value of *a = 0.9* was found to give good results.

The number of iterations varies according to the temperature, in order to allow the global system to reach thermal equilibrium. The number of iterations at temperature T_i is set as follows:

$$N_i = k * (1 + 1/T_i) \tag{24}$$

This allows us to have a large number of iterations (several perturbations) at low temperature, and a small number of iterations at high temperature.

Criterion of Transition

To switch between the two systems DSS and NDSS, we defined two types of transition criterions:

- **Criterion 1:** the transition is made, if the probability $P_c = Exp(-|\Delta F|/T)$ is higher than R

Where R is a random value between (0-1) and $\Delta F = F_{i-1} - F_i$ *is* the difference between the average energy (Σ(energy of state accepted)/ number of iterations) of the current and the previous system

- **Criterion 2:** the transition is done randomly

Computational Experiment Results

Verification and Validation of the PSA Algorithm

In this work, we used a simulation program of PSA developed by language C and a computer with Intel Core (TM) 2 Duo CPU T5750 @ 2.00 GHz processor and 2 GB RAM.

All computational experiments were made with PSA parameters following:

- A systematic encoding by dividing with *p(x)*
- transition criterion: Criterion2
- $P_H = 0.9$, $P_l = 0.000001$

As a first step, we verified the PSA algorithm by comparing himself with the classic SA algorithm developed by authors in [10] by verifying the minimum distance for BCH codes, which the minimum distance is known.

Table 2- comparison between our proposed simulated annealing with a classic simulated annealing

Codes BCH (n, k, d-design)	SA			PSA		
	d_{SA}	Number of iterations	Run Time (s)	d_{PSA}	Number of iterations	Run Time (s)
BCH(15,11,3)	3	792	1	3	17	<1
BCH(31,26,3)	3	2082	7	3	25	<1
BCH(63,24,15)	15	2064	16	15	71	<1
BCH(127,64,21)	27	411520	495	21	17028	1
BCH(255,91,51)	75	1295840	2537	54	934120	468

d_{SA}: minimum distance obtained by the classic simulated annealing developed by [10]

d_{PSA}: minimum distance obtained by the proposed simulated annealing

In the [Table-2], for the three first BCH codes listed, SA and PSA algorithms found the true minimum distance. However, for the two last BCH codes, the gap between the minimum distance obtained by the SA algorithm and the true value is still large, while our PSA algorithm finds this true minimum distance. Moreover we note that the PSA is generally less complex than the SA in term of run time, and PSA uses a very small number of iterations compared to SA.

After that, as a second step, we validated the algorithm for some linear codes: BCH codes, Quadratic residue Codes, for which the minimum distance is known. The results are summarized in the [Table-3], [Table-4].

Table 3- Validation of some QR codes with known minimum distance

Codes QR (n, k, d)	d_{PSA}	Number of iterations	Run Time (s)
QR (127, 64, 19)	19	13899	1
QR (137, 69, 21)	21	1736	1
QR (151, 76, 19)	19	30704	2
QR (223, 112, 31)	32	188037	17

Table 4- Validation of some BCH codes with known minimum distance

Codes BCH (n, k, d-design)	d_{PSA}	Number of iterations	Run Time (s)
BCH (127, 50, 27)	27	13285	1
BCH (127, 85, 13)	13	23491	1
BCH (255, 45, 87)	87	9178	1
BCH (255, 55, 63)	63	953631	58

PSA Algorithm vs Other Optimization Techniques

The [Table-5] shows the results obtained by the other optimization techniques: developed by Askali, et al [18] which use two versions of genetic algorithm (GA-A and GA-B), the work of Walis, et al [19] which use techniques (genetic algorithm (GA), Hill-Climbing and Tabu Search) and the work of Ajitha, et al [22] which use Metropolis Algorithms for the Minimum Weight Code Word Problem.

We note that the PSA algorithm outperformed the results of Walis, et al and it has the same results as Askali's Genetic Algorithms and Ajitha's Metropolis Algorithms for BCH codes less than 255 in length; and outperformed the results of

Askali's Genetic Algorithms and Ajitha's Metropolis Algorithms for length equal to 511.

In our computational experiment, we observed that the PSA algorithm performances are affected by the parameters of the perturbations mechanism and the choice of the transition criterion. We noted in perturbation1, On the one hand the PSA converges fastest when the value of the Hamming weight W_b of the switch vector b ranges from 1 to d_u for a large code length n>127, and for n< 127 the value W_b ranges from 1 to k for QR codes and BCH codes. On the other hand It appears that the value of hamming weight $W_H(Vi+1)$ between 1 and d_u/y where $1 \leq y \leq 4$, significantly improves the performance of the PSA algorithm for certain codes.

In [Table-6], we analyze the choice between the two transition criterions on the convergence of the PSA algorithm for QR(223 ,112,31), BCH(127,64,21) and BCH(255 ,131 ,37) codes.

We show that the randomly transition (criterion2) seems to perform significantly better than the criterion1 in term of minimum distance, run time and number of iterations.

We also determined the probabilities of transition between DSS and NDSS systems; we show that the values of the Transition Probabilities converge almost towards the same value at the end of simulation.

We refer by:

P_{11} the transition probability from DSS to DSS

P_{12} the transition probability from DSS to NDS

P_{21} the transition probability from NDSS to DSS

P_{22} the transition probability from NDSS to NDSS

Table 5- Comparison of our PSA algorithm with other optimization techniques for some BCH codes

Codes BCH (n, k, da-design)	PSA	Askali's GA-A 10000 individuals	Askali's GA-B 10000 individuals	Wallis's GA	Hill-Climbing	Tabu Search	Ajitha's Metropolis Algorithm
BCH (127, 64, 21)	21	21	21	21	28	24	21
BCH (255, 79, 55)	56	57	57	60	74	64	57
BCH (255, 91, 51)	54	58	53	59	72	69	54
BCH (511, 304, 51)	70	87	74	79	90	85	73
BCH (511, 286, 55)	73	98	84	84	96	92	78

Table 6- The transition criterions of the PSA algorithm

Transition Criterions	Codes (n, k, d)	d_{PSA}	Run Time (s)	Number of iterations	Number of transitions	Transition Probabilities			
						P_{11}	P_{12}	P_{21}	P_{22}
Criterion1	BCH (255,131, 37)	38	394	2567830	507	0.75	0.25	0.003	0.996
Criterion2		37	230	1543678	507	0.474	0.525	0.519	0.48
Criterion1	QR (223,112,31)	32	146	1564786	502	0.995	0.004	0.058	0.411
Criterion2		32	15	159687	502	0.484	0.515	0.574	0.425
Criterion1	BCH (127, 64, 21)	21	3	20452	484	0.956	0.043	0.008	0.991
Criterion2		21	1	17676	484	0.529	0.47	0.485	0.514

Conclusion and Perspectives

In this paper we have used the property of degeneration of energy into a new modified simulated annealing algorithm to find a good value of minimum distance of linear block codes. The Simulation results of the proposed simulated annealing algorithm shows that we have successfully speeded up the optimization process while achieving to quite good optimum solutions compared to many optimization methods that are applied to estimate of minimum distance of codes. In the perspectives of this work, we plan to apply this rapid method to construct good linear block codes, and use this method in combination with other optimization methods such as genetic algorithms.

Conflicts of Interest: None declared.

References

[1] Metropolis N., Rosenbluth A.W., Rosenbluth M.N., Teller A.H. & Teller E. (1953) *The Journal of Chemical Physics*, 21(6), 1087-1092.

[2] Kirkpatrick S., Gelatt C.D. & Vecchi M.P. (1983) *Science*, 220 (4598), 671-680.

[3] Bryan K., Cunningham P. & Bolshakova N. (2006) *IEEE Transactions on Information Technology in Biomedicine*, 10(3), 519-525.

[4] Hung M.H., Shu L.S., Ho S.J., Hwang S.F. & Ho S.Y. (2008) *IEEE Transactions on Systems and Humans*, 38(2), 319-330.

[5] Tiwari M. & Cosman P.C. (2008) *IEEE Signal Processing Letters*, 15, 249-252.

[6] Thompson D.R. & Bilbro G.L. (2005) *IEEE Transactions on Cybernetics*, 35(3), 625-632.

[7] Gidas B. (1985) *Journal of Statistical Physics*, 39(1-2), 73-131.

[8] Chibante R. (2010) *Simulated Annealing, Theory with Applications*, Sciyo.

[9] El Gamal A.A., Hemachandra L.A., Shperling I. & Wei V.K.W. (1987) *IEEE Transactions on Information Theory*, 33(1), 116-123.

[10] Muxiang Z. & Fulong M. (1994) *International Journal of Electronics*, 76(3), 377-384.

[11] Grass M. (2000) *IEEE International Symposium on Information Theory*, 25-30.

[12] Brouwer A.E. (1998) *Bounds on the size of linear codes*, *Handbook of Coding Theory*, 1, 295-461.

[13] Berlekamp E.R., McEliece R.J. & Van Tilborg H.C. (1978) *IEEE Transactions on Information Theory*, 24(3), 384-386.

[14] Chen H., Flann N.S. & Watson D.W. (1998) *IEEE Transactions on Parallel and Distributed Systems*, 9(2), 126-136.

[15] Cotta C., Alba E. & Troya J.M. (1998) *Parallel Problem Solving from Nature*, 1498, 305-314.

[16] Daida J.M., Ross S.J. & Hannan B.C. (1995) 6*th International Conference on Genetic Algorithms*, Pittsburgh, 328-335.

[17] Dontas K. & De Jong K. (1990) 2*nd International IEEE Conference on Tools for Artificial Intelligence*, 805-811.

[18] Askali M., Nouh S. & Belkasmi M. (2012) *International Conference on Multimedia Computing and Systems*, 318-324.

[19] Wallis J.L. & Houghten S.K. (2002) 6*th IASTED International Conference on Artificial Intelligence and Soft Computing*, 164-169.

[20] Askali M., Azouaoui A., Nouh S. & Belkasmi M. (2013) *Int'l J. of Communications, Network and System Sciences*, 5(11), 774-784.

[21] Aarts E. & Korst J. (1988) *Simulated Annealing and Boltzmann Machines: a Stochastic approach to Combinatorial Optimization and Neural Computing*, John Wiley & Sons, Inc. New York, NY, USA .

[22] Shenoy K.B.A., Biswas S. & Kurur P.P. (2014) *ACM Conference on Genetic and Evolutionary Computation*, 485-492.

[23] MacWilliams F.J. & Sloane N.J.A. (1977) *The Theory of Error-Correcting Codes*, Elsevier, 16.