Ant Colony Optimization for Soft Decoding of Linear Block Codes

Ahmed Azouaoui, Hicham Bouzkraoui and Youssef Hadi

Abstract

Soft decision decoding is considered as a NP-hard problem, hence classical algorithms and exhaustive search methods become impracticable in large problem space. Nowadays, metaheuristic algorithms have the upper hand as an alternative for this problem using smart instead of exhaustive search. In this paper, we introduce a soft decoding algorithm, based on Ant Colony Optimization, such metaheuristic technic has proven its efficiency in optimization problems in general. The implementation of this algorithm, the first of its kind, was run over an AWGN channel, then we proceed with different experiments to tune its parameters, after that, the results were then compared with several previous decoding algorithms. In fact it gives better performances than most up to date decoders such DDGA, CGAD-M and CGAD-HSP algorithms. In other side, we discussed the complexity of the proposed algorith compared most known decoders.

Keywords: Ant Colony Optimization, Soft Decoding, Error correcting codes, Linear Codes, BCH,QR,RS

INTRODUCTION

After Claude Shannon’s famous paper [1], communication scientists have made significant efforts in order to find coding/decoding models that approach channel capacity, especially on the popular AWGN channel. In the first decades of coding theory, algebraic coding/decoding concept was the main axis of researches (Berlekamp [2], Sugiyama et al. [3]).

In this paradigm the optimum decoding rule is to select the closest codeword to the received symbol using hamming distance metric. In fact it makes hard decision on the received signal, thus works only on binary vectors. This approach has had many success especially over a BSC channel, because this decoding is equivalent to the ML decoder. However over an AWGN channel, they have not proved to be suitable, and because such decoding model discards the information accuracy (you may think to information loss in quantization process).

Soft-decoding algorithms came to fill this gap, actually they use signal information received directly from demodulation module before any hard decision (see Fig. 1), hence they use real numbers associated with each codeword symbol in decoding stage, and decision is almost based on Euclidean distance metric. Therefore this decoding is capable of correcting more errors and reach higher performance.

However, the implementation is often more complex and costly in terms of computational resources.

Soft-decoding is classified as a NP-hard problem [4], which means that classical exhaustive search would not be effective and remains impracticable. In this context, several algorithms were proposed, In 1966 G.Forney proposed a pseudo soft decoding system, called “Generalized Minimum Distance Decoding” GMD [5]. Algorithm A* was also used to implement a soft decision decoder by S.Han et al. [6]. After the emergence of Artificial Intelligence (AI) and metaheuristic methods, several works were published of decoder based on neural networks and Genetic Algorithms (GA). In fact GA seems to give good results which is proved by Maini et al. decoder [7], then several GA came after such as the SDGA [8], the AutDAG decoder by S.Nouh et al. [9] and Shakeel et al. GA-Based Decision Soft decoder [10]. In contrast of the most decoders, other algorithms were proposed using dual code, such as CGAD [11], then recently two algorithms, CGAD-M and CGAD-HSP were proposed based on CGAD[12], also Maini was enhanced to DDGA decoder [13], the SDGA was also extended to DSGA [14]. In other hand for LDPC codes GAMD decoder [15] was proposed. In this spirit, we propose in this paper, a novel soft decoder based on Ant Colony Optimization (ACOD), as far as we know, this approach has not been previously used to decode linear codes.

ACO was derived from nature by observing how ants behave collectively to find the optimum path from the nest to the food. In this paper we start our work on linear binary codes, therefore our search space is the messages space and the used metric is the Euclidean distance. However, our decoder is applicable for non-binary and non-cyclic codes. The remainder of this article will be organized as follow: in section 2, we will introduce the Ant Colony Optimization method, then in section 3 we will define our problem context and the adopted ACO algorithm. The next section will report the simulation results and compare with previous works. Finally we conclude our paper by summary of this work and the future perspectives.
ANT COLONY OPTIMIZATION (ACO)

While observing ants when searching food near the nest, researchers have discovered that they have the capacity to find the shortest path to food. After many experiences, scientists have figured out that ants use a chemical communication medium called pheromone. In fact, these ants drop the pheromone on their path. When arriving to some intersection, the ants make a probabilistic choice based on pheromone quantity on each possible path, therefore after several walks, the concentration of pheromone grows on some optimal path. Furthermore, the evaporation of the pheromone trails weakens non optimal paths and potential optimal path which becomes blocked.

The TSP Problem (Travelling Salesman Problem) was the subject of the first ACO algorithm implementation AS (Ant System) [16]. The problem is to find the shortest path visiting n cities; each city must be visited only once. This mechanism is modeled by a graph G where the cities are the vertices and the edges are formed by paths between cities. In the AS algorithm there is a set of iterations y (1<=y<=ymax), every ant x (1<=x<=m) scans the graph and build a complete path of n cities, for each ant the relative path between city i and city j depends on several conditions:

1) The set of cities already visited by ant x which is on city i, noted by Jix
2) \( \eta_{ij} \) The visibility of city j to i, it is a measure which let the ants to choose the near city rather than the far ones. Often \( \eta_{ij} \) is a decreasing function of the distance \( d_{ij} \) between city i and city j (for example \( \eta_{ij} = \frac{1}{d_{ij}} \)).
3) The pheromone quantity dropped on the path between 2 cities i and j at iteration y, called intensity and noted \( \tau_{ij}(y) \) which define the global attractiveness of parts of the whole path and continually updated by the ants.

Arriving to city i the ant x makes a probabilistic choice based on the following formula which defines the probability to walk to city j:

\[
P_{ij}^x(y) = \left\{ \begin{array}{ll}
\left( \frac{(\tau_{ij}(y))^\alpha (\eta_{ij})^\beta}{\sum_{j \in J^*} (\tau_{ij}(y))^\alpha (\eta_{ij})^\beta} \right) & j \in J^*_i \\
0 & \text{otherwise}
\end{array} \right.
\]

(1)

\( \alpha \) and \( \beta \) are 2 parameters which control the relative role of the intensity \( \tau_{ij}(y) \) of pheromone and the visibility \( \eta_{ij} \), a tradeoff should be done to play on diversification and intensification of the algorithm behavior.

After each iteration y every ant x drops a quantity of pheromone over the walked path, depending on solution quality

\[
\Delta \tau_{ij}^x(y) = \left\{ \begin{array}{ll}
\frac{\rho}{L^x(y)} (i,j) \in T^x(y) \\
0 & \text{otherwise}
\end{array} \right.
\]

(2)

Where the \( T^x(y) \) is the path done by ant x on iteration y, \( L^x(y) \) is its length and Q is a fixed parameter.

The evaporation mechanism comes to complete the algorithm, the pheromone intensity decreases on bad paths based on the following rule:

\[
\tau_{ij}(y + 1) = (1 - \rho)\tau_{ij}(y) + \sum_{k=1}^{m} \Delta \tau_{ij}^k(y)
\]

(3)

Where m is the size of ant colony and \( \rho \) is the evaporation ratio.

The Algorithm can be briefly summarized in the next pseudo code:

For each iteration y=1,…, ymax
   For each ant x=1,…, m
      Select randomly a city i
      For each non visited city i
         Select a city j from J^x using equation (1)
      End For
      Deposit pheromone \( \Delta \tau_{ij}^x(y) \) on the path \( T^x(y) \)
      using equation (2)
   End For
   Vaporization of the paths using equation (3)
End For

ACOD Algorithm

We note F2 the binary field. A linear code of length n is a subspace \( C \subseteq F_2^n \). A linear code with length n, dimension k and minimum distance d will be noted \( (n,k,d) \). This code can be described by a \( k \times n \) matrix \( G \) over the Field \( F_2 \) called Generator matrix, a message \( m \) can be then encoded easily to a codeword \( c \) using the equation:

\[
c = c \cdot m \quad (4)
\]

In other side, we define a parity check \( (n-k) \times n \) matrix noted \( H \) which satisfy \( Hg_i = 0 \) which we have the following property:

\[
\forall v \in F_2^n, v \text{ is codeword } \leftrightarrow H v^t = 0
\]

(5)
After that the codeword \( c=(c_1,c_2,...,c_n) \) is BPSK modulated to a signal \( U=(U_1,U_2,...,U_n) \), and sent over a Gaussian channel, an AWG noise represented by a random \( n \)-vector \( N=(N_1,N_2,...,N_n)^T \) independent of \( U \), with iid components given by \( N_i \sim \mathcal{N}(0,N_0/2) \) is added to transmitted signal over the channel. In the receiver side the received signal \( R=(R_1,R_2,...,R_n) \) such that \( R=U+N \). The likelihood probability is given by:

\[
f_{R/U} = \frac{1}{(\pi N_0)^{n/2}} \exp(-\frac{(R-U)^2}{2N_0})
\]

From the above equation we can conclude that ML is equivalent to the minimum Euclidean distance, before any hard decision. At this stage comes the role of our decoder and reacts as follow:

1. Make a hard decision \( v=(v_1,v_2,...,v_n) \) of received signal \( R=(R_1,R_2,...,R_n) \):
   
   \[
v_i = \begin{cases} 
0, & R_i < 0 \\
1, & R_i \geq 0
\end{cases}
\]

2. Calculate the syndrome \( s=Hv \), if \( s=0 \) then output \( v \) and exit, otherwise continue

3. Sort the sequences \( R=(R_1,R_2,...,R_n) \) in decreasing order based on reliability \((|R_i|>|R_{i+1}|)\) to obtain new sequences \( R''=(R''_1,R''_2,...,R''_n) \), let’s note \( \pi_1 \) the permutation \( R''=\pi_1(R) \), we apply \( \pi_1 \) to \( G \) to obtain \( G' \). The most \( k \) reliable \( R_i \) will allow our algorithm to start searching from close solution to the optimum.

4. Apply Gaussian elimination to \( G' \) to obtain a systematic matrix \( G'' \).

5. We apply the ACO algorithm to obtain the best close signal to the received signal and in consequence the code \( c''=(c''_1,c''_2,...,c''_n) \). This algorithm is detailed afterwards.

6. The code \( c'' \) is related to the \( G'' \) matrix, thus our estimated transmitted codeword is then

\[
\hat{c} = \pi_1^{-1}(c''_k)
\]

The ACO algorithm is modeled by the following graph (in case of \( k=4 \)):

![Figure 2: Graph of linear code (k=4)](image)

The codeword \( c'' \) is obtained from message \( m=[m_1,m_2,...,m_k] \in \mathbb{F}_2^k \) via the generator matrix \( G'' \) of the code, therefore the search space will be the field \( \mathbb{F}_2^k \) represented by the above graph of course when \( k=4 \).

For ant \( x \) at iteration \( y \) the probability of setting the bit-\( i \) to 1 is defined as follows:

\[
P_{ij}(y) = \frac{(\tau_{ij}(y))^{\alpha}(\eta_{ij})^{\beta}}{\sum_{l=1}^{m}(\tau_{il}(y))^{\alpha}(\eta_{il})^{\beta}}
\]

The parameters \( \alpha, \beta, \) and \( \eta_{ij} \) are adjusted based on simulation results. At the end of ant \( x \) walk all bits \( mi \) are set and the path is the constructed message \( m=[m_1,m_2,...,m_k] \in \mathbb{F}_2^k \) therefore the solution quality is measured by Euclidean distance of \( U'' \) from \( R'' \), thus we define the pheromone to be dropped by ant \( x \) at iteration \( y \) as follow:

\[
\Delta \tau_{ij}(y) = \frac{Q}{\sum_{l=1}^{m}(|R''_{il} - U''_{il}|)^2}
\]

Where the bit-\( i \) \((1 \leq i \leq k)\) is set to \( j \in \{0,1\} \) by the ant \( x \) and \( Q \) is a parameter defined by simulation results. The algorithm could be then summarized as below:

Set ACO parameters \( \alpha, \beta, \rho, m, Q, \gamma_{max} \)

Initialize pheromone intensity \( \tau_{ij}(0) \) and visibility \( \eta_{ij} \)

For each iteration \( y=1,...,\gamma_{max} \)

For each ant \( x=1,...,m \)

Set randomly the bit-

Set the bit-\( i \) using equation (8)

Compute codeword \( c'' \) using equation (4) and matrix \( G'' \)

Modulate codeword \( c'' \) to a signal \( U'' \)

Compute \( d_{ece} = \sum_{i=1}^{n}(R''_i - U''_i)^2 \)

Update best codeword \( c_b \) based on best \( d_{ece} \)

Deposit pheromone \( \Delta \tau_{ij} \) on the path \( T \)

End For

Vaporization of the paths using equation (3)

End For

RESULTS AND DISCUSSION

In order to find the most suitable set of parameters \( \{\alpha, \beta, \rho, m, Q, \gamma_{max}\} \). We perform several simulations of performance which is the Bit Error (BER) expressed as function of SNR (Signal to Noise Ratio), with different ACO parameters values.

For every parameter optimization we fix the other parameters while varying the concerned parameter. The remaining parameters were set by default according to the below table:
ACO PARAMETERS SETTINGS

Figure 3 shows, that the best performance is achieved when $\alpha=0.1$. Intuitively large values of $\alpha$ tend to amplify the initial paths choosed by ants.

From figure 4, $\beta=2.5$, is almost the best choice, which is confirmed by the recommendation [17].

The performance is quite similar as seen in figure 5, we find $\rho=0.5$ is the most suitable value, which is recommended [17].

By estimating on average over all SNR values in figure 6, we choose to set $m=51$, several simulations prove that it is useless to consider $m>k$, it is recommended to set $m \approx k$ [17].

In figure 7, there is no visible correlation to the minimum distance, we may set $Q=10$ taking into account the average over all SNR values.

Based on figure 8, for $y_{\text{max}} > 50$ iterations, there is no visible improvement of performance, therefore we could set $y_{\text{max}} = 50$.

### Table 1: Parameters Setting

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default code</td>
<td>BCH(63,45,7)</td>
</tr>
<tr>
<td>Channel</td>
<td>AWGN</td>
</tr>
<tr>
<td>Modulation</td>
<td>BPSK</td>
</tr>
<tr>
<td>Minimum number of bit</td>
<td>200</td>
</tr>
<tr>
<td>Minimum number of blocs</td>
<td>1000</td>
</tr>
</tbody>
</table>

**Figure 3:** Impact of parameter $\alpha$ variation on BER.

**Figure 4:** Impact of parameter $\beta$ variation on BER.

**Figure 5:** Impact of parameter $\rho$ variation on BER.
In this subsection, we compare our ACO decoder with the some decoders known by their best performance: AutDAG, SDGA, DSGA, GAMD, Chase, OSD-I, CGAD, CGAD-M, CGAD-HSP, DDGA, Maini and an algebraic soft decoder SIHO proposed by I. Chana [18]. For this reason, we will use different codes as a comparison base.
From the Figure 9, we can see that our algorithm performs better than SIHO and SDGA, in fact we can gain 1dB compared to SDGA at 4.10^{-4} BER, but over performs DDGA only for medium and low noise level, while keeping the same performance as the Maini decoder.

In Figure 10, the ACOD has better performance than Chase-2, cGA-HSP and cGA-M. AutDAG has better performance in high level noise, however for SNR>=4dB, ACOD perform better.

From the Figure 11, our decoder performs better than CGAD, actually we can have a gain of 1.5 dB at 10^{-4}, besides, it is slightly better than Maini and DDGA. In Figure 12, the ACOD has better performance than Chase-2 and SIHO.

Figure 11: Performance of Maini, DDGA, CGAD and ACOD decoders on BCH (63, 51, 5)
From the Figure 13 and figure 14 ACOD over performs slightly Maini, DDGA and OSD-1 but over performs the DSGA and Chase-2

b) Complexity Analysis

The below table gives our algorithm complexity with its competitors:

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chase-2</td>
<td>$O(2^t n^2 \log_2 n)$</td>
</tr>
<tr>
<td>Maini</td>
<td>$O(N_i N_g [kn + \log N_i])$</td>
</tr>
<tr>
<td>DDGA</td>
<td>$O(N_i N_g [kn - k + \log N_i])$</td>
</tr>
<tr>
<td>Shakeel</td>
<td>$O(T_c kn)$</td>
</tr>
<tr>
<td>OSD-1</td>
<td>$O(n^4)$</td>
</tr>
<tr>
<td>SDGA</td>
<td>$O(2^t (N_i N_g [kn^2 + kn + 1\log(N_i)])$</td>
</tr>
<tr>
<td>CGAD</td>
<td>$O(T_c (n - k))$</td>
</tr>
<tr>
<td>ACOD</td>
<td>$O(m y_{max} kn)$</td>
</tr>
</tbody>
</table>

- $t = (d-1)/2$, the error correcting capability
- $N_i$, parameter used is Genetic Algorithm, called population size.
- $N_g$, parameter used is Genetic Algorithm, called number of generations.
- $T_c$, parameter used is CGAD and Shakeel algorithm, called average number of generations.
The Chase-2 has the worst performance in terms of algorithmic complexity either in term of n and even exponentially with t [19], while OSD-1 [20] and SDGA have lower complexity. The Maini, DDGA, CGAD and ACOD have almost the same complexity, however if we observe the parameters setting used by some of these decoders we can express complexity as follow:

**Table III: Complexity in (k, n)**

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Parameters Setting</th>
<th>Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maini</td>
<td>$N_i = 300, N_o = 100$</td>
<td>$O(30000(kn + 2.5))$</td>
</tr>
<tr>
<td>DDGA</td>
<td>$N_i = 300, N_o = 100$</td>
<td>$O(30000(k(n-k) + 2.5))$</td>
</tr>
<tr>
<td>CGAD</td>
<td>$T_c = 1000$</td>
<td>$O(1000k(n-k))$</td>
</tr>
<tr>
<td>ACOD</td>
<td>$m = k, y_{max} = 50$</td>
<td>$O(50k^2n)$</td>
</tr>
</tbody>
</table>

From the above table, and considering asymptotic behaviour, we can see that for $k<200$ ACOD has the lowest complexity, furthermore even for $k<600$ ACOD can still over perform Maini and DDGA

**CONCLUSION**

In this paper we have proposed a Soft decoder based on ACO, when tested on different kinds of codes, provides good performance comparing to existing soft decoding algorithms. We focus also on algorithm parameters tuning to enhance the performance. The main advantage of this algorithm is that it can be used for non-cyclic and non-binary codes and on different kind channel models, while keeping a low implementation complexity. Actually compared to its concurrent decoders, our soft decoding keeps low algorithmic complexity as shown in this article.

The obtained results, open the ACO as new methodology to be used in coding theory and let us hope to implement our algorithm in dual space. Furthermore hybridization with other metaheuristic methods could improve the decoding capabilities of the new algorithm.

**REFERENCES**


and Systems, ICMCS 2012, pp 1176-1182.


