MINIMUM DISTANCE COMPUTATION OF LINEAR CODES VIA GENETIC ALGORITHMS WITH PERMUTATION ENCODING

José Gómez Torrecillas
F. J. Lobillo
Gabriel Navarro

UNIVERSIDAD DE GRANADA

NP-HARD PROBLEM

Compute the minimum Hamming distance of an arbitrary linear code over a finite field, or compute the minimum Hamming weight of an arbitrary linear code over a finite field

Linear Algebra-based algorithms

- Browser-Zimmermann Algorithm
- Hamming Distance
- Entropy
- Genetic Algorithms

or, variations of EIZ Algorithm...

Use an intelligent exhaustive search. Exponential time with respect to dimension and bit-size of elements

HENRY VARADARAJAN

Hierarchical algorithms for solving optimization problems

- Genetic Algorithms
- Standardized Annealing

For binary code

Stiches

Probabilistic approaches

- Leven
- Clustering
- Forest
- Simo

Exhaustive search for some random vector spaces. Impractical for large finite fields

Key-tool: Permutation Encoding

Theorem. Let G be a generating matrix of an \([n,k]\)-linear code \(C\) over a finite field \(F\). There exists a permutation \(P\) in the symmetric group \(S_n\) and a row of \(\text{ref}(GM_P)\) whose weight equals the minimum distance of \(C\), where \(\text{ref}(GM_P)\) is the row reduced echelon form of \(GM_P\) and \(M_P\) is the permutation matrix of \(P\).

Reformulation of the problem

Compute the minimum of fit: \(S_n \rightarrow \mathbb{R}\), mapping \(P\) to the minimum weight of the rows of \(\text{ref}(GM_P)\)

Components

- Representation: Permutation encoding
- Initial population: Selected randomly
- Selection: Random selection
- Fitness: Given by a function \(S_n \rightarrow \mathbb{R}\)
- Genetic operators: Crossover, mutation
- Flowchart

Algorithm

1. \(P(0)\) ← random initial population of size \(p\).
2. \(i = 0\)
3. While \(i < (\text{no of iterations or end time}\) do
4. \(X\) ← \(P(i)\)
5. While \(X\) is \(\not\sim\) \(\varnothing\) do
6. \(X = (X(1)\;\ldots\;X(m))\) \(\varnothing\)
7. \(\varnothing = (\text{random chromosomes of } P(i))\)
8. \(S = P(i)\;\ldots\;S\) \(\varnothing\)
9. \(S = AX(1)\;\ldots\;AX(m)\) \(\varnothing\)
10. If \(S\) is not crossed then
11. \(S = \varnothing\)
12. For \(i = i + 1\) do
13. \(X = X(1)\;\ldots\;X(m)\)
14. \(X = \text{best of } X\)
15. \(\text{end of iteration}\)
16. \(i = i + 1\)
17. Return best

GLN Genetic Algorithm

Pseudocode

Input: \(G\), generating matrix; \(p\), crossover probability; \(p_m\), mutation probability; \(r\), crossover step; \(c\), iterations or execution time; \(p\), population size

Output: \(d\) upper bound of the minimum distance:

1. \(P(0) = \text{random initial population of size } p\)
2. \(i = 0\)
3. While \(i < (\text{no of iterations or end time}\) do
4. \(X = P(i)\)
5. While \(X\) is \(\not\sim\) \(\varnothing\) do
6. \(X = (X(1)\;\ldots\;X(m))\) \(\varnothing\)
7. \(\varnothing = (\text{random chromosomes of } P(i))\)
8. \(S = P(i)\;\ldots\;S\) \(\varnothing\)
9. \(S = AX(1)\;\ldots\;AX(m)\) \(\varnothing\)
10. If \(S\) is not crossed then
11. \(S = \varnothing\)
12. For \(i = i + 1\) do
13. \(X = X(1)\;\ldots\;X(m)\)
14. \(X = \text{best of } X\)
15. \(\text{end of iteration}\)
16. \(i = i + 1\)
17. Return best

Small Example

Let \(G\) generated by \(G = \begin{bmatrix} a^5 & a^3 & a^2 & a^0 \\ a^6 & a^1 & a^0 & a^0 \\ a^2 & a^3 & a^4 & a^0 \end{bmatrix} \in M_{30}(F)\), where \(F = \mathbb{F}_2(x)\)

<table>
<thead>
<tr>
<th>Initial population</th>
<th>Offspring</th>
<th>New population</th>
<th>Replacement</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\mathcal{A})</td>
<td>(\mathcal{B})</td>
<td>(\mathcal{C})</td>
<td>(\mathcal{D})</td>
</tr>
</tbody>
</table>

Experimental Results

<table>
<thead>
<tr>
<th>Field</th>
<th>Length</th>
<th>Basis</th>
<th>Randomly</th>
<th>Summation</th>
<th>Population</th>
<th>Penalty</th>
<th>Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(P_1)</td>
<td>39</td>
<td>14</td>
<td>12</td>
<td>12.000</td>
<td>5.000</td>
<td>0.400</td>
<td>0.000</td>
</tr>
<tr>
<td>(P_2)</td>
<td>60</td>
<td>30</td>
<td>20</td>
<td>20.000</td>
<td>5.000</td>
<td>0.400</td>
<td>0.000</td>
</tr>
<tr>
<td>(P_3)</td>
<td>90</td>
<td>19</td>
<td>40</td>
<td>40.000</td>
<td>10.000</td>
<td>0.400</td>
<td>0.000</td>
</tr>
<tr>
<td>(P_4)</td>
<td>90</td>
<td>56</td>
<td>22</td>
<td>22.000</td>
<td>30.000</td>
<td>0.400</td>
<td>0.000</td>
</tr>
<tr>
<td>(P_5)</td>
<td>90</td>
<td>60</td>
<td>16</td>
<td>16.000</td>
<td>30.000</td>
<td>0.400</td>
<td>0.000</td>
</tr>
<tr>
<td>(P_6)</td>
<td>130</td>
<td>75</td>
<td>28</td>
<td>28.000</td>
<td>150.000</td>
<td>0.400</td>
<td>0.000</td>
</tr>
<tr>
<td>(P_7)</td>
<td>130</td>
<td>85</td>
<td>23</td>
<td>23.000</td>
<td>150.000</td>
<td>0.400</td>
<td>0.000</td>
</tr>
<tr>
<td>(P_8)</td>
<td>130</td>
<td>95</td>
<td>18</td>
<td>18.000</td>
<td>50.000</td>
<td>0.400</td>
<td>0.000</td>
</tr>
</tbody>
</table>

Stop: Use the algorithm in its current form to determine the distances. Examples: \(A_k\), \(B_{20}\), \(P_{20}\), \(M_{20}\), \(P_{20}\), \(P_{30}\), \(P_{40}\), \(P_{50}\), \(P_{60}\), \(P_{70}\), \(P_{80}\)

1. \(p\) in parentheses: \(p\) = number of times that the best bound was reached
2. Average number of iterations and time (in seconds) for the best bound is reached

GLN-GA vs Randomness

Distributions of the distances obtained for some codes of the filters by GLM-GA and the random selection of 200 generations