HAMFAST: Fast Hamming Distance Computation

Francesco Pappalardo
Department of Mathematics and Computer Science
University of Catania
Catania, Italy

Cristiano Calonaci
CINECA
Bologna, Italy

Marzio Pennisi
Department of Mathematics and Computer Science
University of Catania
Catania, Italy

Emilio Mastriani
Consorzio COMETA
Catania, Italy

Santo Motta
Department of Mathematics and Computer Science
University of Catania
Catania, Italy

Abstract

Similarity is a vague concept which can be treated in a quantitative manner only using appropriate mathematical representation of the objects to compare and a metric on the space representation. In biology the mathematical representation of structure relies on strings taken from an alphabet of \( m \) symbols. Very often binary strings, \( m = 2 \), are used. The size of the binary string depends on the complexity of the structure to represent, so the string can be quite long. The Hamming distance is the most used metric with binary strings. The computational effort required to compute the Hamming distance linearly depends on the size of the string. However even a linear effort case may be computational heavy if many computations are required. One of the fastest computational approach to evaluate Hamming distances relies on look-up tables. The computational performance, however, rapidly deteriorates with the size of binary string length, due to cache misses. We present a computational strategy and implementation which can handle huge number of Hamming distance evaluation between binary strings of arbitrary length keeping computational performance competitive.

1. Introduction

Similarity is a vague concept which can be treated in quantitative manner only using appropriate mathematical representation of the objects to compare and a metric on the representation space. The concept of similarity is widely used in many scientific areas like computer science, biology and medicine. In computer science specific fields where similarity concepts occur are information retrieval, signals theory, bioinformatics and others. In biology [2] have pointed out that no theory exists on how to choose the best similarity measure. The most common used are distance-based measures. The choice of a similarity measure is relevant on the final result of the analysis.

The mathematical representation of structures usually relies on strings taken from an alphabet of \( m \) symbols. Very often binary strings, \( m = 2 \), are used. The size of the binary string depends on the complexity of the structure to represent, so the string can be quite long. When a binary representation is used, the similarity measure is, in most cases, computed via the Hamming distance [9] between binary strings.

As an example in analyzing gene expression data, the choice of a binary approach offers several advantages in dealing the problem [16]. These advantages refer both to the modeling aspect, i.e. simplification of models and possible improvement in classification problems, and to the computational one, i.e. noise resilience and computational efficiency.

In the field of gene expression analysis, microarrays data is composed by real valued expressions that would imply norm-based distance measures in \( \mathbb{R} \). In [16] the real values outcomes are converted in binary strings and Hamming distance is used to measure similarity.

Boolean models of Gene Regulatory Network describe
the gene activation state of the network by binary strings of length \( l \), where \( l \) is the number of genes involved in the network [10]. Based on this description [4] have recently studied the evolution of GRN requiring a huge number of Hamming distance computations.

In a different scenario, i.e. comparing two protein databases, one must find all pairs of proteins that align with high score. Distance-based approaches to this problem map each peptide in the database to a point in a metric space. [8] show how to map peptides to bit vectors such that the distance between any two peptides is closely approximated by the Hamming distance.

The immune system is characterized by a dynamic interplay and by a fine-tuned balance at different levels: organism, cellular and molecular level. The immune system is composed of many interdependent cell types, organs, and tissues which jointly protect the body from infections (bacterial, parasitic, fungal, or viral) and from the growth of tumor cells. In doing its job, the immune system has an enormous diversity (for example \( 10^{16} \)–\( 10^{18} \) different T-cell receptors, and \( 10^{11} \) B-cell clonotypes in an individual).

Modelling the immune system presents a formidable challenge because of the large number of components to be considered. A key problem is the representation of the binding between different cells and molecules. A widely used representation is the shape-space representation [5, 15], where diversity is described using strings usually with a two symbols alphabet. Binary strings represent the generic “binding site” between cells (through their receptors) and target molecules (through peptides and epitopes). It is clear that having binary strings of length \( l \), we are able to represent an immunological repertoire of size \( 2^l \). Even if the shape-space approach is unphysical the abstraction captures important features of receptor/ligand binding [6].

Recently we presented models of immune system and related simulators[12, 14, 3, 13]. There, we modeled the immunological repertoire using binary strings of 12 bits, reaching a diversity of \( 2^{12} \) but our final goal is to simulate the immune system at a natural scale. Because a real immunological repertoire has a diversity of \( 10^{14} \sim 2^{42} \), this implies to compute the Hamming distance of binary strings of length \( \geq 42 \).

The computational effort required to compute the Hamming distance linearly depends on the size of the string. However even a linear effort may be computational heavy if many computations are required. The best known method to compute Hamming distance between two bit strings relies on a pre-computed look-up table kept in computer cache memory; it is computationally fast if the cache misses are rare. With this method, the computation of Hamming distance of binary strings of length \( l \) requires a look-up table of \( O(2^{2l}) \) entries. In what follows we present a method that can handle strings of arbitrary length keeping the computational effort competitive.

2 Algorithm

Given two bit strings \( b_1 \) and \( b_2 \) of length \( l \), the Hamming distance is defined as the number of mismatching bits. This is equivalent to count the number of set bits (i.e. the number of bits equal to “1”) of the bit string \( b_{xor} \) defined as the result of logical exclusive or (\( xor \)) of the bit strings \( b_1 \) and \( b_2 \). Our problem can then be reduced to a bit counting problem.

The best bit counting method takes only 12 CPU operations, which avoids the memory and potential cache misses of the look-up table method. The counts of bits set in the bytes is done in parallel, and the sum total of the bits set in the bytes is computed by multiplying and shifting right bits, using binary magic numbers.

Magic numbers are thoroughly discussed in [7]. Magic numbers in mathematics and computer science are numbers that show special properties when used in certain computations. In particular, they are successfully used in algorithms involving bit handling. They offer faster versions of algorithms over those that can be developed without their use, typically by a factor of \( n / \log n \). Our implementation, which is an extension of the well known bit count algorithm [1] implements a branchless computation of the population count. It is based on a \( O(\log(n)) \) algorithm that successively groups the bits into groups of 2, 4, 8, 16, 32, and 64 (using binary magic numbers) while maintaining a count of the set bits in each group. With this implementation, we were able to deal with binary strings of 64 bits. The method is easily extendable to any bits strings length, but keeping the limit of 12 elementary CPU operations, requires CPUs capable of doing \( l \)-bits operations in one time, \( l \geq 64 \). The method however, as shown in the following, performs better than the look-up table method when \( l \geq 16 \). Another method to compute Hamming distance is done breaking the binary string \( b_{xor} \) in three parts. This approach reduces the size of the look-up table and uses a smaller amount of physical computer memory. For example working with binary strings of length 64, one can use a small look-up table of \( 2^{22} \) entries that will be accessed three times.

3 Performance and statistics

We implemented the three methods (look-up table, split-ted look-up table and bitcount) using ANSI C programming language and we performed computational tests in three different architectures, namely Intel(R) Xeon(TM), IBM(R) Power 5(TM) and AMD(R) Opteron(TM) platforms.

Tests were conducted on \( 10^8 \) Hamming distance computations, with binary strings randomly selected using the MT19937 random number generator [11]. Cache
misses were analyzed only for Intel platforms by Valgrind (Cachegrind tool) release 3.1.0.

Figure 1 shows that the look-up table methods perform well when the number of cache misses are lower. With the classical look-up table, we were forced to limit the length of the bit strings to 29 bits (i.e. using “int” data type which requires 4 bytes, we completely fill the available RAM memory, that is 2 GB). The other two methods were tested with binary strings up to 64 bits in length. As one can see from the figures, bitcount method performs better than the others in all examined platforms.

4 Conclusion

Quantifying similarity in biology and in other sciences is a task of fundamental importance. Mathematical representation of biological structures often relies on binary strings and Hamming distance is often used to measure similarity. Huge number of computations and high length of binary strings need an approach that allows easy computer implementation and high performance in terms of computational speed.

We presented HAMFAST, a fast and effective algorithm that computes Hamming distances between binary strings. HAMFAST was implemented using ANSI C programming language and it can be used without additional effort in all the biological applications that require a huge number of Hamming distance computations using binary strings of arbitrary length.

Acknowledgement

This work was partially supported by MIUR under PRIN Modelli del la teoria cinetica matematica per particelle attive nello studio di sistemi complessi, 2007-2008. The authors would like to thank A.Torrisi of the University of Catania Library Service in supporting for document retrieval.

References


