Substitutional Tolerant Markov Models for Relative Compression of DNA Sequences

Diogo Pratas(✉), Morteza Hosseini, and Armando J. Pinho

IEETA, University of Aveiro, Aveiro, Portugal
{pratas,seyedmorteza,ap}@ua.pt

Abstract. Referential compression is one of the fundamental operations for storing and analyzing DNA data. The models that incorporate relative compression, a special case of referential compression, are being steadily improved, namely those which are based on Markov models. In this paper, we propose a new model, the substitutional tolerant Markov model (STMM), which can be used in cooperation with regular Markov models to improve compression efficiency. We assessed its impact on synthetic and real DNA sequences, showing a substantial improvement in compression, while only slightly increasing the computation time. In particular, it shows high efficiency in modeling species that have split less than 40 million years ago.

Keywords: Markov models · Tolerant Markov models · Relative compression · Genomic sequences

1 Introduction

Several applications in bioinformatics require the compression of a string, x, given other string, y. This is the case when one needs to analyze or store compactly as possible the data [1–6]. The information in y can be used together with that on x or alone. In the so called conditional approach [7,8], the compressor can explore the information that is contained in y, as well as that from x (assuming causality), according to

$$C(x|y) = \sum_{i=1}^{|x|} -\log_2 P(x_i|x_1^{i-1}, y), \tag{1}$$

where $|x|$ is the size of x and x_i is i^{th} element of x. So, for example, x_3^5 is a substring of x composed by x_3, x_4 and x_5.

The relative approach [6,9–14], $C(x\|y)$, assumes that information comes exclusively from y, according to

$$C(x\|y) = \sum_{i=1}^{|x|} -\log_2 P(x_i|x_{i-\pi}^{i-1}, y), \tag{2}$$

F. Fdez-Riverola et al. (eds.), *11th International Conference on Practical Applications of Computational Biology & Bioinformatics*, Advances in Intelligent Systems and Computing 616, DOI 10.1007/978-3-319-60816-7_32

where $i - \pi$ is the allowed size of elements from x that can be used in order to search for regularities in y. For $i \leq \pi$ we assume a uniform distribution.

In order to calculate the probabilities of Eq. 2, we need data models that describe y efficiently. Both Ziv-Merhav dictionary-based models [9,13,15] and Markov models [5,14,16,17] have been successfully used in diverse data type applications. However, for DNA sequences, Markov models proved to be more efficient [6].

Markov models (MMs), also known as finite-context models (FCMs), are statistical models. A MM of an information source assigns probability estimates to the symbols of an alphabet, Θ, according to a conditioning context computed over a finite and fixed number, k, of past outcomes (order-k MM) [18]. At element i, these conditioning outcomes are represented by $x_{i-k+1}^{i-1} = x_{i-k+1}, \ldots, x_{i-1}$. A non relative MM can store each outcome of the past in memory, while a MM working in relative mode can only store the outcomes seen in y. The number of conditioning states of the model in DNA sequences is 4^k. The cooperation between MM of different orders has proved to be a more efficient solution for representing DNA sequences, instead of competition [19].

High order MM, typically with $k \geq 13$, proved to be one of the most important models for DNA sequence representation [20], as well as to address other applications [21–23]. However, when substitutional mutations occur between two identical sequences, high order MM fall short to represent the data. This happens because, if, for example, we use an order-20 MM and we have a probability of one random substitution for each 20 bases, the probability that the same context is seen again is low. The DNA data between close species is frequently of this nature, because they share a common ancestral. Moreover, the distinct majority of the editions in the DNA sequences are of substitutional nature.

Aware of these characteristics, we have recently proposed a preliminary approach to deal with substitutional mutations in DNA sequences [6]. In this paper, we consolidate the concept of substitutional tolerant Markov models (STMM) and we apply them to the relative compression case. After, we measure its impact on synthetic genomic data, exploring some characteristics of compressing the elements from a reverse order, as well as some combinations between both. Finally, we show some comparative results between whole genomes.

2 Substitutional Tolerant Markov Model (STMM)

A substitutional tolerant Markov model (STMM) is a probabilistic-algorithmic finite-context model. It assigns probabilities according to a conditioning context that considers the last symbol, from the sequence to occur, as the most probable, given the occurrences stored in the memory, such as those from y, instead of the true occurring symbol.

For a symbol $s \in \Theta$, the estimator of a STMM, working in relative mode, is given by

$$P(s|{x'}_{i-k}^{i-1}, y) = \frac{N(s|{x'}_{i-k}^{i-1}, y) + \alpha}{N({x'}_{i-k}^{i-1}, y) + \alpha|\Theta|}, \tag{3}$$

where function N accounts for the memory counts regarding the model and x' is a copy of x, edited according to

$$x'_i = \underset{\forall s \in \Theta}{\operatorname{argmax}}\, P(s|{x'}^{i-1}_{i-k}, y). \tag{4}$$

The parameter α allows balancing between the maximum likelihood estimator and a uniform distribution. For deeper orders, α should be generally lower than one.

When a STMM (relative or non-relative model) is cooperating with any other model, besides being probabilistic, can also be algorithmic, because they can be switched on or off given its performance, according to a threshold, t, defined before the computation.

Both relative and non-relative modes work with a threshold, t, that enables or disables the model according to the number of times that the context has been seen. Listing 1.1. describes the process for enabling or disabling a STMM.

Listing 1.1. Algorithm of a STMM, described in C language, with comments.

```
int GetBestId(int *array){
  int x, best = 0, maximum = array[0];
  for(x = 1 ; x < N_SYMBOLS ; ++x)                // N_SYMBOLS = 4 (bases)
    if(array[x] > maximum){
      maximum = array[x];
      best = x;
    }
  return best;         // RETURN THE HIGHEST ELEMENT POSITION OF AN ARRAY
}

void Fail(Model *M){                                      // ACTION FOR FAIL
  int x, fails = 0;
  for(x = 0 ; x < M->k ; ++x)                        // USING HISTORY COUNT
    if(M->history[x] != 0)                           // THE NUMBER OF FAILS
      ++fails;
  if(fails > M->threshold)                  // FAILS MORE THAN THRESHOLD?
    M->on = 0;                                             // SET STMM OFF
  else                                                     //    OTHERWISE
    ShiftBuffer(M->history, M->k, 1);                      // ADD ONE FAIL
}

void Hit(Model *M){                           // ACTION FOR HIT (SUCCESS)
  ShiftBuffer(M->history, M->k, 0);                         // ADD ONE HIT
}

void CorrectSTMM(Model *M, PModel *P, int sym){
  int best = 0;
  if(M->on == 0){                                             // IF IS OFF
    M->on = 1;                                            // TURNS STMM ON
    memset(M->history, 0, M->k);
  }
  else{                                                   // ELSE IF IS ON
    if((best = GetBestId(P->freqs) == sym){            // IF BEST ID = SYM
      Hit(M);                                        // CALL HIT FUNCTION
    }
    else{                                                 //     OTHERWISE
      Fail(M);                                      // CALL FAIL FUNCTION
      M->seq->buf[M->seq->idx] = best;               // UPDATE NEW SYMBOL
    }
  }
  UpdateCBuffer(M->seq);                         // UPDATE SEQUENCE BUFFER
}
```

The threshold, t, is set at the beginning of the computation. We also need a Boolean cache-array (history) to store the past k hits/fails. For example, consider that $k = 7$ and that $c_0 =$ CACGTCA is the current context. Also, consider that the number of past symbol occurrences following c_0 was $\mathrm{A} = 1, \mathrm{C} = 0, \mathrm{G} = 0, \mathrm{T} = 0$. If the symbol that is being compressed is G (contradicting the probabilistic model), a MM would have as next context $c_1 =$ ACGTCAG. However, the STMM would use a c_1', taking into account the most probable outcome and, hence, $c_1' =$ ACGTCAA. Therefore, the next probabilistic model would be dependent on the past context assumed to be seen and, hence, it assumes that the symbol that was compressed is A.

3 Results

For producing the results, we have used synthetic and real data. The synthetic data made available a controlled comprehension of the STMMs, while the real data shown the characteristics that are also not controlled. The materials to replicate both results on synthetic and real data are available, under GPL v3 license, at the repository https://github.com/pratas/STMM. All experiments were run on Ubuntu Linux v16.04 LTS, with gcc v5.3.1, using only one Intel Core i7-6700K 3.4 GHz CPU, 32 GB of RAM and a solid-state hard drive.

3.1 Synthetic Data

In Fig. 1 we have simulated a sequence y with 200 bases, copied y to x and inserted edits in several positions of x, specifically at positions 50, 100, 102, 150, 152 and 154. Then we have compressed x relatively to y, assuming the order of each element of x as $x_1, x_2, ..., x_{|x|}$ as right direction, $x_{|x|}, ..., x_2, x_1$ as left direction and the minimum complexities of both directions as min.

As it can be seen, the cooperation between MMs and STMMs led to a much better approximation of the data. While the MMs can not address efficiently the data after a substitution occurs, between a period of time that seems related with the k-size, the cooperation between MMs and STMMs address them efficiently, having an almost strict decay to a low complexity value.

In Fig. 2 we have simulated a sequence y. Then, we have made 12 copies, for each one applied some degree of random substitutional mutations, and concatenated all into a final sequence, called x. Then we have compressed, using $C(x_i \| y)$, and plotted it. As it can be seen, with 7.5% of substitutional mutations the cooperation of only MMs reaches the average of 1 BPB (bits per base), while the cooperation between MMs and STMMs reaches the same BPB only at 15% of substitutional mutations.

3.2 Real Data

We have used two eagle whole genomes in non-assembled mode, namely White-tailed eagle (*Haliaeetus albicilla*, 1.14 GB, 26X) and Bald eagle (*Haliaeetus leucocephalus*, 1.26 GB, 88X), from [24]. We have also used the reference genomes of

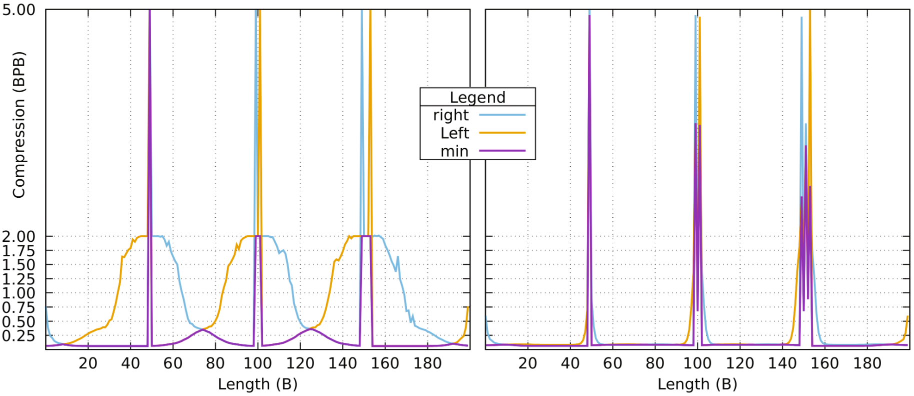

Fig. 1. Relative compression using a cooperative set of MMs (left plot) and a cooperative set of MMs and STMMs (right plot). The compression direction is included for right and left, as well as the minimum (min) between both for each elements. The data is synthetic. The length is in bytes (B). The experiment can be replicated using the script *runSmallBidirection.sh*, from the repository described in this paper.

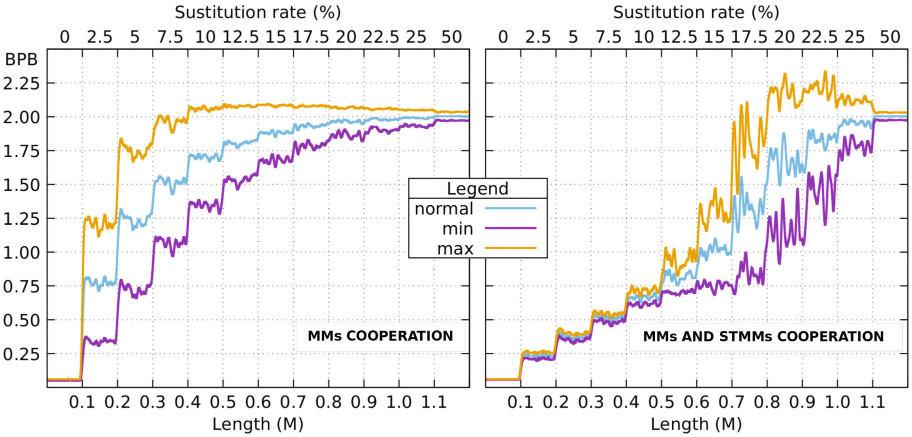

Fig. 2. Relative compression using a cooperative set of MMs (left plot) and a cooperative set of MMs and STMMs (right plot). The synthetic data has been copied from y, creating multiple concatenated x's. For each 100k of data (bottom axis), a substitution mutation rate has been applied (top axis). Besides normal, the legend shows the computation of min and max. These are the minimum (min) and maximum (max) functions of each element processed in left and right directions. The length is in mega bytes (M). The experiment can be replicated using the script *runRelativeBidirection.sh*, from the repository described in this paper.

human, chimpanzee, gorilla, orangutan, and marmoset from the NCBI. We have used a setup of 4 MMs in cooperation with order-k of $\{4, 6, 13, 20\}$ and the α of, respectively, $\{1, 1, 0.5, 0.005\}$. Only one STMM was used with $k = 20$, $\alpha = 0.5$

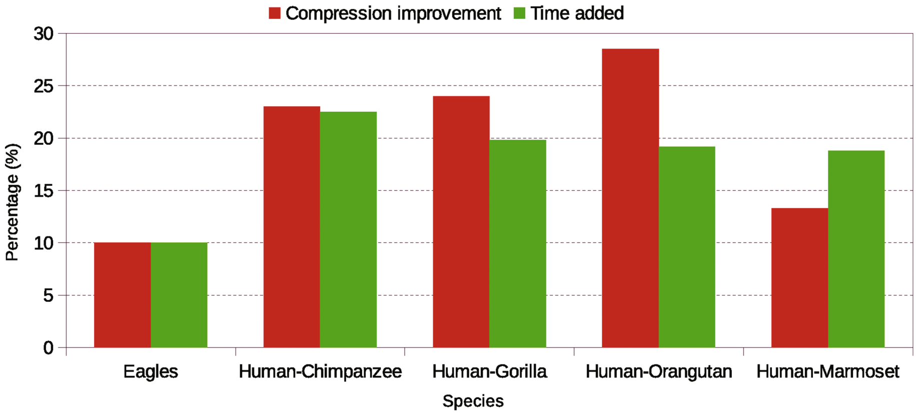

Fig. 3. Compression improvement and compression time added between the relative compression using a cooperative set of MMs and a cooperative set of MMs and STMMs. Percentages are given by $STMM_{bytes}/MM_{bytes} \times 100$ for compression improvement and $MM_{minutes}/STMM_{minutes} \times 100$ for time added.

and $t = 5$. The experiment can be replicated using the script *runBirds.sh* and *runPrimates.sh*.

As can be seen in Fig. 3, to compress the Bald eagle relatively to White-tailed eagle, using only a cooperation between MMs, we needed 31, 561, 247 bytes. Adding the cooperation of the STMMs, we reached 34, 864, 683 bytes, which is around 10% of improvement, using the same RAM memory (13.8 GB) and around 10% more computational time. These species are believed to have diverged ≈1 million years ago (mya) [25].

As can be seen in Fig. 3, to compress a chimpanzee relatively to a human genome, using only a cooperation between MMs, we needed 274, 450, 972 bytes and near 80 min. Adding the cooperation of the STMMs we were able to spend only 210, 691, 987 bytes, which is around 23% of improvement, using the same RAM memory (26.3 GB) and around more 22.5% of computational time. The human and chimpanzee lineages are believed to have diverged ≈3–4.5 mya [26].

To compress a gorilla relatively to a human genome, using only a cooperation between MMs, we needed 262, 271, 376 bytes. Adding the cooperation of the STMMs we were able to spend only 199, 204, 749 bytes, which is around 24% of improvement, using the same RAM memory (26.3 GB) and around 19.8% more computational time. The human and gorilla lineages are believed to have diverged before ≈5–9 mya [26].

To compress a orangutan relatively to a human genome, using only a cooperation between MMs, we needed 418, 481, 411 bytes. Adding the cooperation of the STMMs we were able to spend only 299, 316, 387 bytes, which is around 28.5% of improvement, using the same RAM memory (26.3 GB) and around 19.2% more computational time. The human and orangutan lineages are believed to have diverged before 10 mya [26].

Finally, to compress a marmoset relatively to a human genome, using only a cooperation between MMs, we needed 562, 916, 901 bytes. Adding the cooperation of the STMMs we were able to spend only 488, 238, 361 bytes, which is around 13.3% of improvement, using the same RAM memory (26.3 GB) and around 18.8% more computational time. The human and marmoset lineages are believed to have diverged around ≈40 mya [27].

4 Conclusions

In this paper, we have proposed a new model for relative compression of DNA sequences—the substitutional tolerant Markov model (STMM). We have shown that it addresses efficiently some degree of substitutional mutations, being a model efficient to use between species that divergence less than 40 million years ago, such as between some primates or eagles. The time added by the model to the compressor is affordable, given the compression improvement—for example, between human and orangutan is around 28.5%. This model is, therefore, a strong candidate to be used in ancient DNA analysis, namely because of the high substitutional mutation rates of the data.

Acknowledgments. This work was partially funded by FEDER (POFC-COMPETE) and by National Funds through the FCT - Foundation for Science and Technology, in the context of the projects `UID/CEC/00127/2013` and `PTCD/EEI-SII/6608/2014`.

References

1. Ferragina, P., Giancarlo, R., Greco, V., Manzini, G., Valiente, G.: Compression-based classification of biological sequences and structures via the universal similarity metric: experimental assessment. BMC Bioinform. **8**(1), 252 (2007)
2. Pinho, A.J., Garcia, S.P., Pratas, D., Ferreira, P.J.S.G.: DNA sequences at a glance. PLoS ONE **8**(11), e79922 (2013)
3. Campagne, F., Dorff, K.C., Chambwe, N., et al.: Compression of structured high-throughput sequencing data. PLoS ONE **8**(11), e79871 (2013)
4. Benoit, G., Lemaitre, C., Lavenier, D., et al.: Reference-free compression of high throughput sequencing data with a probabilistic de Bruijn graph. BMC Bioinform. **16**(1), 288 (2015)
5. Pratas, D., Silva, R.M., Pinho, A.J., Ferreira, P.J.S.G.: An alignment-free method to find and visualise rearrangements between pairs of DNA sequences. Sci. Rep. **5**, 10203 (2015)
6. Pratas, D., Pinho, A.J., Ferreira, P.: Efficient compression of genomic sequences. In: Proceedings of the Data Compression Conference on DCC-2016, Snowbird, Utah, pp. 231–240, March 2016
7. Kolmogorov, A.N.: Three approaches to the quantitative definition of information. Probl. Inf. Transm. **1**(1), 1–7 (1965)
8. Li, M., Vitányi, P.: An Introduction to Kolmogorov Complexity and Its Applications, 3rd edn. Springer, New York (2008)
9. Ziv, J., Merhav, N.: A measure of relative entropy between individual sequences with application to universal classification. IEEE Trans. Inf. Theory **39**(4), 1270–1279 (1993)

10. Benedetto, D., Caglioti, E., Loreto, V.: Language trees and zipping. Phys. Rev. Lett. **88**(4), 048702-1–048702-4 (2002)
11. Cilibrasi, R.L., et al.: Statistical inference through data compression. Ph.D. thesis, Institute for Logic, Language and Computation, Universiteit van Amsterdam (2007)
12. Cerra, D., Datcu, M.: Algorithmic relative complexity. Entropy **13**, 902–914 (2011)
13. Coutinho, D.P., Figueiredo, M.: Text classification using compression-based dissimilarity measures. Int. J. Pattern Recogn. Artif. Intell. **29**(5), 1553004 (2015)
14. Pinho, A.J., Pratas, D., Ferreira, P.: Authorship attribution using relative compression. In: Proceedings of the Data Compression Conference on DCC-2016, Snowbird, Utah, March 2016
15. Coutinho, D.P., Figueiredo, M.A.: An information theoretic approach to text sentiment analysis. In: ICPRAM, pp. 577–580 (2013)
16. Fink, G.A.: Markov Models for Pattern Recognition: From Theory to Applications. Springer Science & Business Media, London (2014)
17. Brás, S., Pinho, A.J.: ECG biometric identification: a compression based approach. In: Engineering in Medicine and Biology Society (EMBC), pp. 5838–5841. IEEE (2015)
18. Sayood, K.: Introduction to Data Compression, 3rd edn. Morgan Kaufmann, Burlington (2006)
19. Pinho, A.J., Pratas, D., Ferreira, P.: Bacteria DNA sequence compression using a mixture of finite-context models. In: Proceedings of the IEEE Workshop on Statistical Signal Processing, Nice, France, June 2011
20. Pratas, D., Pinho, A.J.: Exploring deep Markov models in genomic data compression using sequence pre-analysis. In: Proceedings of the 22nd European Signal Processing Conference on EUSIPCO-2014, Lisbon, Portugal, pp. 2395–2399, September 2014
21. Zhao, W., Wang, J., Lu, H.: Combining forecasts of electricity consumption in China with time-varying weights updated by a high-order Markov chain model. Omega **45**, 80–91 (2014)
22. Kwak, J., Lee, C.H., et al.: A high-order Markov-chain-based scheduling algorithm for low delay in CSMA networks. IEEE/ACM Trans. Netw. **24**(4), 2278–2290 (2016)
23. Kárnỳ, M.: Recursive estimation of high-order Markov chains: approximation by finite mixtures. Inf. Sci. **326**, 188–201 (2016)
24. Jarvis, E.D., Mirarab, S., Aberer, A.J., et al.: Whole-genome analyses resolve early branches in the tree of life of modern birds. Science **346**(6215), 1320–1331 (2014)
25. Wink, M., Heidrich, P., Fentzloff, C.: A mtDNA phylogeny of sea eagles (genus haliaeetus) based on nucleotide sequences of the cytochrome b-gene. Biochem. Syst. Ecol. **24**(7–8), 783–791 (1996)
26. Prado-Martinez, J., Sudmant, P.H., Kidd, J.M., Li, H., et al.: Great ape genetic diversity and population history. Nature **499**(7459), 471–475 (2013)
27. Sequencing, T.M.G., Consortium, A., et al.: The common marmoset genome provides insight into primate biology and evolution. Nat. Genet. **46**(8), 850–857 (2014)