Can General-Purpose Compression Schemes Really Compress DNA Sequences?

Toshiko Matsumoto
toshikom@is.s.u-tokyo.ac.jp
Kunihiko Sadakane
sada@is.s.u-tokyo.ac.jp
Hiroshi Imai
imai@is.s.u-tokyo.ac.jp
Takumi Okazaki
takumi@is.s.u-tokyo.ac.jp
Department of Information Science, University of Tokyo, Tokyo 113-0033, Japan

Keywords: DNA sequence, compression

1 Introduction

Today, more and more DNA sequences are becoming available. The information about DNA sequences are stored in molecular biology databases. The size and importance of these databases will be bigger and bigger in the future, therefore this information must be stored or communicated efficiently. Because the DNA sequences consist of four bases, two bits are enough to store each base, but they are hard to compress further. There have been developed several special-purpose compression algorithms for DNA sequences [2]. These DNA-oriented compression algorithms use characteristic structures of DNA such as palindromes, approximate matches, and can compress them less than two bits per base.

Nevertheless, the question of the title remains. The reasons are twofold. For practical applications, general-purpose compressions schemes are used in ordinary communications. From the viewpoint of compression theory, it is a big challenge whether newly devised general schemes can really compress DNA sequences. This note answers affirmatively to the question. An implementation (*CTW* for short) [5] of the general context tree weighting method surely reduces the size of DNA sequences. Some interesting aspects of other standard compression schemes are also revealed.

2 Algorithms and Compression Ratios

Compression results are shown in Table 1. The unit of compression ratio is *bit per base*. The DNA sequences are standard benchmark data used in [2]. *gzip -9* corresponds to the widely used *gzip* with option *-9*. *lz* adopts the LZ77 scheme, as *gzip*, with the sliding dictionary of size 32KB to 1MB [4]. *arith* [5] implements the arithmetic coding, which is the *CTW* program with order 0 [5]. *arith+* [3] enhances the arithmetic coding with an LZ77-like function encoding a repeat by the length and the distance to it by the dictionary of size 32KB and 1MB. *normal PPMD+* gives the results for the statistical compression program *PPMD+* [6], and *adapted PPMD+* for modified *PPMD+* program whose value of the maximum *order* is adapted. The values in parenthesis are the best maximum of *order* for each sequence. *PPMD+ escape* is an improved *PPMD+* program that does not use "escape" after four alphabets all appear in the context. The values of *normal CTW* are results of *CTW* program with alphabet size 256. *CTW-4* is an improved *CTW* program whose size of alphabet is changed to 4.

As for widely used *compress*, *gzip*, and *bzip2* with default options, in all cases *compress* or *gzip* only expand in size. The average compression ratio is 2.185 for *compress* and 2.271 for *gzip*. With the option *-9*, *gzip* can really compress "HUMGHCSA", a typical example having so many repeats. When *bzip2* is used, the compression ratio of "HUMGHCSA" is 1.729 per base, but the ratios for the other sequences are all more than 2 bits per base with average value 2.138. *LZ* can compress only "HUMGHCSA" as in *gzip -9*, *bzip2*. It does not take account of long distances of repeat in DNA, hence if the size of the buffer becomes big the compression ratio becomes worse in many cases.

Table 1: The compression ratio of each algorithm

DNA sequence name	Sequence length	*gzip -9*	*lz (32K)*	*lz (1M)*	*arith*	*arith+ (32K)*	*arith+ (1M)*	*normal PPMD+*	*adapted PPMD+*	*PPMD+ escape*	*normal CTW*	*CTW -4*
CHMPXX	121024	2.220	2.234	2.276	1.867	1.866	1.866	1.977	1.840(1)	1.839(3)	1.879	**1.838**
CHNTXX	155844	2.291	2.300	2.352	1.957	1.957	1.956	2.062	1.934(1)	1.935(3)	1.974	**1.933**
HEHCMVCG	229354	2.279	2.286	2.344	1.985	1.985	1.985	2.053	1.965(3)	1.959(3)	1.997	**1.958**
HUMDYSTROP	38770	2.377	2.427	2.432	1.949	1.948	1.948	2.237	1.921(1)	1.931(3)	1.960	**1.920**
HUMGHCSA	66495	1.551	1.580	1.513	2.001	1.488	1.438	2.077	1.694(11)	1.514(11)	1.376	**1.363**
HUMHBB	73308	2.228	2.255	2.286	1.969	1.913	1.911	2.116	1.921(2)	1.923(3)	1.917	**1.892**
HUMHDABCD	58864	2.209	2.241	2.264	1.999	1.951	1.950	2.130	1.948(2)	1.938(3)	1.909	**1.897**
HUMHPRTB	56737	2.232	2.269	2.287	1.972	1.943	1.942	2.130	1.932(2)	1.926(3)	1.922	**1.913**
MPOMTCG	186609	2.280	2.289	2.326	1.984	1.972	**1.961**	2.075	1.966(2)	1.964(3)	1.989	1.962
PANMTPACGA	100314	2.232	2.249	2.285	1.880	1.873	1.873	2.018	1.872(1)	1.869(3)	1.902	**1.866**
SCCHRIII	315339	2.265	2.268	2.308	1.962	1.955	**1.935**	2.023	1.950(2)	1.948(3)	1.976	1.945
VACCG	191737	2.190	2.194	2.245	1.919	1.862	1.862	2.002	1.910(2)	1.908(3)	1.897	**1.857**

arith can compress the sequences in which the ratios of 'a', 't', 'g', and 'c' differ, but it expands a little the sequence in which the ratios are nearly equal (the arithmetic coding cannot utilize the repeat structure). In all cases, *arith+* can compress less than 2 bits per base, because *arith+* considers not only the length of repeat but also the distance of repeat, and then determines whether there is a profit in using it. Hence, the compression ratio of *arith+* can be improved by enlarging the buffer size.

Although in most cases *normal PPMD+* only expand in size, *adapted PPMD+* can really compress all of the sequences. In many cases the compression ratio of *PPMD+ escape* are a little better than that of *adapted PPMD+*. We examined the effect of the value of the maximum of *order*. It is known that, in English texts, the optimal value of the maximum of *order* is 5. But, for DNA, *adapted PPMD+* and *PPMD+ escape* mostly achieve the best compression ratio when the maximum of *order* is less or equal 3. In some papers *PPMD+* fails to compression of DNA sequence [1], but we thus found that adapting the maximum of *order* in *PPMD+* is important. It is interesting to see the best maximum of *order* for *PPMD+ escape* is three, which may correspond to protein encoding.

In many sequences, *CTW* can achieve better compression ratio than *PPMD+*, and if we change the size of alphabet to 4 the compression ratio becomes better. The *order* is set to 32, and the effect of the value of γ is examined; the result is that in most cases about 0.005 is optimal. In Table 1, the best values are given. It is remarkable that the order of *CTW* can be made larger than the corresponding value 3 in *PPMD+ escape*, which demonstrates the robustness and power of the context tree weighting method. *CTW* [5] is still slow and uses large memory, which should be improved further.

References

[1] Balkenhol, B., Kurts, S., and Shtarkov, Y. M., Modifications of the burrows and wheeler data compression algorithm, *Proc. of IEEE Data Compression Conference*, 188–197, 1999.

[2] Chen, X., Kwong, S., and Li, M., A compression algorithm for DNA sequences and its applications in genome comparison, *Genome Informatics*, 10:51–61, 1999.

[3] Matsumoto, T., DNA sequences compression algorithm using context tree weighting method, Senior Thesis, Department of Information Science, University of Tokyo, 2000, to appear.

[4] Sadakane, K., and Imai, H., Improving the speed of LZ77 compression by hashing and suffix sorting, submitted.

[5] Sadakane, K., Okazaki, T., and Imai, H., Implementing the context tree weighting method for text compression, *Proc. of IEEE Data Compression Conference*, 2000, to appear.

[6] Teahan, W. J. and Cleary, J. G., The entropy of English using PPM-based models, *Proc. of IEEE Data Compression Conference*, 53–62, 1996.