Published online 1 December 2011

Nucleic Acids Research, 2012, Vol. 40, No. 4 ***e27***
doi:10.1093/nar/gkr1124

GReEn: a tool for efficient compression of genome resequencing data

Armando J. Pinho*, Diogo Pratas and Sara P. Garcia

Signal Processing Lab, IEETA/DETI, University of Aveiro, 3810–193 Aveiro, Portugal

Received August 5, 2011; Revised October 17, 2011; Accepted November 8, 2011

ABSTRACT

Research in the genomic sciences is confronted with the volume of sequencing and resequencing data increasing at a higher pace than that of data storage and communication resources, shifting a significant part of research budgets from the sequencing component of a project to the computational one. Hence, being able to efficiently store sequencing and resequencing data is a problem of paramount importance. In this article, we describe GReEn (Genome Resequencing Encoding), a tool for compressing genome resequencing data using a reference genome sequence. It overcomes some drawbacks of the recently proposed tool GRS, namely, the possibility of compressing sequences that cannot be handled by GRS, faster running times and compression gains of over 100-fold for some sequences. This tool is freely available for non-commercial use at ftp://ftp.ieeta.pt/~ap/codecs/GReEn1.tar.gz.

INTRODUCTION

Inspired by the Biocompress algorithm of Grumbach and Tahi (1), the last two decades have witnessed the proposal of a myriad of algorithms for compressing genomic sequences [(2–16) for a recent review]. The acquired knowledge regarding genome structure that these compression algorithms have been providing, through their representation of genomic sequences using probabilistic models, is likely to surpass in relevance the benefits of the effective storage space reduction provided.

One of the most successful compression algorithms specifically designed for genomic sequences is XM, a statistical method proposed by Cao *et al*. (14), though other approaches may present competitive or even superior results for some classes of genomes (15,17). XM relies on a mixture of experts for providing symbol by symbol probability estimates that are fed to an arithmetic encoder. The XM algorithm comprises three types of experts: (i) order-2 Markov models; (ii) order-1 context Markov models, i.e. Markov models that rely on statistical information from a recent past (typically, the 512 previous symbols); (iii) the copy expert, which considers the next symbol as part of a copied region from a particular offset. The probability estimates provided by the set of experts are then combined using Bayesian averaging and sent to the arithmetic encoder.

Common practice continues to rely on standard and general purpose data compression methods, e.g. *gzip* or *bzip2*. However, this practice may be close to a turning point, as the rate at which genomic data is being produced is clearly overtaking the rate of increase in storage resources and communication bandwidth.

The development of high-throughput sequencing technologies that offer dramatically reduced sequencing costs enables possibilities hardly foreseeable a decade ago (18). Large-scale projects such as the 1000 Genomes Project (http://www.1000genomes.org/) and The Cancer Genome Atlas (http://cancergenome.nih.gov/), as well as, prizes that reward cheaper, faster, less prone to errors and higher-throughput sequencing methodologies (e.g. http://genomics.xprize.org/) are paving the way to individual genomics and personalized medicine (19). As such, huge volumes of genomic data will be produced in the near future. However, as a very significant part of the genome is shared among individuals of the same species, these data will be mostly redundant. Some ideas for storing and communicating redundant genomic data have already been put forward, based on, for example, single nucleotide polymorphism (SNP) databases (20), or insert and delete operations (21).

Recently, Wang *et al*. (22) proposed a compression tool, GRS, that is able to compress a sequence using another one as reference without requiring any additional information about those sequences, such as, a reference SNPs map. The algorithm proposed by Kuruppu *et al*. (23) RLZ, is also able to perform relative Lempel–Ziv compression of DNA sequences, though its current implementation cannot fully handle sequences that have characters outside the {a,c,g,t,n} set.

*To whom correspondence should be addressed. Tel: +351 234 370500; Fax: +351 234 370545; Email: ap@ua.pt

Other approaches propose encoding sequence reads output by massively parallel sequencing experiments (24–27), which is also a very important problem. The compression of short reads shares some points with the problem being addressed here, though it needs to cope with other requirements, such as, the efficient representation of base calling quality information.

In this article, we describe GReEn (Genome Resequencing Encoding), a new tool for compressing genome resequencing data using a reference genome sequence. As such, it addresses the same problem as GRS (22), RLZ (23) or XM (14). However, as will be demonstrated, GReEn outperforms GRS in storage space requirements and running times, though GRS can handle some sequences in a very effective way, and it overcomes RLZ's and XM's lack of support for arbitrary alphabets and inferior performance.

GReEn is a compression tool based on arithmetic coding that handles arbitrary alphabets. Its running time depends only on the size of the sequence being compressed. Moreover, it provides compression gains of over 100-fold for some sequences, when compared to GRS, and even larger gains when compared to RLZ. Finally, GReEn handles without restriction sequences that cannot be compressed with GRS due to excessive difference to the reference sequence.

MATERIALS AND METHODS

Dataset

We use the same data as in (22), for ease of comparison with GRS: two versions of the first individual Korean genome sequenced, KOREF_20090131 and KOREF_20090224 (28); two versions of the genome of the thale cress *Arabidopsis thaliana*, TAIR8 and TAIR9 (29,30); and two versions of the rice *Oryza sativa* genome, TIGR5.0 and TIGR6.0 (31). We also present results for four additional human genome assemblies, namely, the genome of J. Craig Venter referred to as HuRef (32), the Celera alternate assembly referred to as Celera (33), the genome of a Han Chinese individual referred to as YH (34), and the human genome reference assembly build 37.p2, as made available by the National Center for Biotechnology Information and referred to as NCBI37 (35).

Software availability

The codec (encoder/decoder) is implemented in the C programming language and is freely available for non-commercial purposes. It can be downloaded at ftp://ftp.ieeta.pt/~ap/codecs/GReEn1.tar.gz.

The compression method

As with GRS (22), GReEn relies on a reference sequence for compressing the target sequence. The reference sequence is generally only slightly different from the target sequence, although this is not mandatory. In fact, it is possible to use a sequence from a different species as reference, though, as expected, the compression efficiency depends on the degree of similarity between both reference and target sequences. Moreover, in order to recover the target sequence, the decoder needs access to exactly the same reference sequence as that used by the encoder.

The codec developed in GReEn is able to handle arbitrary alphabets, although it automatically ignores all lines beginning with the '>' character, as well as, all newline characters. We denote by $\mathcal{C} = \{c_1, c_2, , c_{|\mathcal{C}|}\}$ the set of all different characters, or symbols, that are found in the target sequence, where $|\mathcal{C}|$ denotes the number of elements in $\mathcal{C}$, i.e. the alphabet size.

Each character of the target sequence is encoded by an arithmetic encoder (36). As with any arithmetic encoder, besides the symbol to encode, it is necessary to provide the probability distribution of the symbols. One major advantage of arithmetic coding is its ability to adjust the probabilistic model as the encoding proceeds, in response to the changing probability distribution from one encoded symbol to the next.

We denote by $\theta(c)$ the relative frequency of character $c \in \mathcal{C}$ in the target sequence, and by $P_n(c)$ the estimated probability of character $c \in \mathcal{C}$ when encoding the character at position n in the target sequence. The set of probabilities $\{P_n(c), c \in \mathcal{C}\}$ are passed down to the arithmetic coder. Note that, whereas $\theta(c)$ values are fixed for a given target sequence, $P_n(c)$ values usually change along the coding process. For a sequence $x^N = x_1 x_2 x_N,\ x_i \in \mathcal{C}$, with N characters, the arithmetic coder produces a bitstream with

$$-\sum_{n=1}^{N} \log_2 P_n(x_n) \quad (1)$$

bits, which demonstrates the importance of providing good probability estimates to the arithmetic coder.

The probability distribution, $P_n(c)$, can be provided by two different sources: (i) an adaptive model (the copy model) which assumes that the characters of the target sequence are an exact copy of (parts of) the reference sequence; (ii) a static model that relies on the frequencies of the characters in the target sequence, i.e. $\theta(c)$. The adaptive model is the main statistical model, as it allows a high compression rate of the target sequence, particularly in areas where the target and reference sequences are highly similar. However, this adaptive, or copy, model will at times not be used (the reasons why will be detailed shortly), and the static model will act as a fallback mechanism, feeding the arithmetic coder with the required probability distribution.

The copy model

The copy model is inspired by the copy expert of the XM DNA compression method (14), relying on a pointer to a position in the reference sequence that has a 'good chance' of containing a character identical to that being encoded. As encoding of the target sequence proceeds, the pointer associated with the copy model may be repositioned to different locations of the reference sequence. When this repositioning occurs, all parameters of the model are reset. Besides accounting for the number of times, t_n,

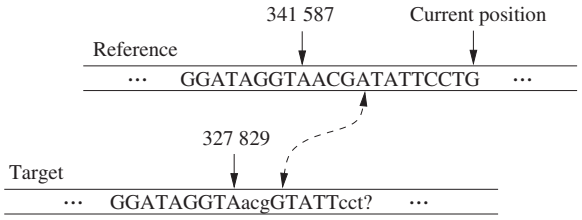

Figure 1. The copy model. In this example, the copy model was restarted at position 341 587 of the reference sequence, corresponding to position 327 829 of the target sequence. Since then, it has correctly predicted 5 characters, if the case is considered, and a total of 11 characters if the case is ignored. The dashed arrow indicates a failed prediction. According to this example, the next character to be predicted is 'G'.

that the copy model was used after the previous repositioning, two more counters are maintained: h_n^1 stores the number of times the model guessed the correct character including the correct case (uppercase or lowercase), and h_n^2 records the number of times the model guessed the character but failed the case (for example, it guessed 'A' but the correct character was 'a').

Figure 1 exemplifies the operation of the copy model. Consider the most recent repositioning occurred at position 341 587 of the reference sequence, corresponding to position 327 829 of the target sequence (in this example, the reference is ahead of the target, but this may be different in other cases). Assuming the codec is going to compress the character marked with '?', then the character predicted by the copy model would be 'G' (the one under the 'Current position' arrow), with $t_n = 12$, $h_n^1 = 5$ and $h_n^2 = 6$. The characters linked by the dashed arrow indicate a prediction error (the predicted character was 'A', whereas the correct one was 'G').

Computing the probabilities. Let us denote by p_n^1 the character predicted by the copy model ('G' in the example in Figure 1) and by p_n^2 the case converted p_n^1 ('g' according to the example in Figure 1). If $p_n^1, p_n^2 \in \mathcal{C}$ (note that characters of the reference sequence that do not appear in the target sequence do not belong to $\mathcal{C}$), the probabilities that are passed down to the arithmetic coder are given by

$$P_n(c) = \begin{cases} \dfrac{h_n^1 + 1}{t_n + 3}, & \text{for } c = p_n^1 \\ \dfrac{h_n^2 + 1}{t_n + 3}, & \text{for } c = p_n^2 \\ \dfrac{1 - P_n(p_n^1) - P_n(p_n^2)}{1 - \theta(p_n^1) - \theta(p_n^2)}\,\theta(c), & \text{for } c \neq p_n^1, p_n^2. \end{cases} \quad (2)$$

The first two branches of Equation (2) correspond to Laplace probability estimators of the form

$$P(\mathcal{E}_k) = \frac{N_{\mathcal{E}_k} + 1}{\sum_{k=1}^{K} N_{\mathcal{E}_k} + K}, \quad \mathcal{E}_k \subset \mathcal{C}, \quad (3)$$

where the $\mathcal{E}_k$s form a set of K collectively exhaustive and mutually exclusive events, and $N_{\mathcal{E}_k}$ denotes the number of times that event $\mathcal{E}_k$ has occurred in the past. In Equation (2) we considered three events, namely, $\mathcal{E}_1 = \{p_n^1\}$, $\mathcal{E}_2 = \{p_n^2\}$ and $\mathcal{E}_3 = \mathcal{C} \setminus \{p_n^1, p_n^2\}$. The third branch of Equation (2) defines how the probability assigned to $\mathcal{E}_3$, i.e. $1 - P(\mathcal{E}_1) - P(\mathcal{E}_2)$, is distributed among the individual characters of $\mathcal{E}_3$. This distribution is proportional to the relative frequencies of the characters, $\theta(c)$, after discounting the effect of treating p_n^1 and p_n^2 differently.

If only p_n^1 or p_n^2 belongs to $\mathcal{C}$, the probabilities are given by

$$P_n(c) = \begin{cases} \dfrac{h + 1}{t_n + 2}, & \text{for } c = p \\ \dfrac{1 - P_n(p)}{1 - \theta(p)}\,\theta(c), & \text{for } c \neq p \end{cases}, \quad (4)$$

where $h = h_n^1$ if $p = p_n^1$, or $h = h_n^2$ if $p = p_n^2$. As such, we have considered only two events, namely, $\mathcal{E}_1 = \{p\}$ and $\mathcal{E}_2 = \mathcal{C} \setminus \{p\}$, where the distribution of probabilities among the characters of $\mathcal{E}_2$ is performed as before.

Finally, if both $p_n^1, p_n^2 \notin \mathcal{C}$, the probabilities communicated to the arithmetic coder are the character frequencies of the target sequence, i.e.

$$P_n(c) = \theta(c). \quad (5)$$

Starting and stopping the copy model. Typically, the codec starts by constructing a hash table with the occurrences and corresponding positions in the reference sequence of all k-mers of a given size (the default size is $k = 11$, but it can be changed using a command line option). Figure 2 shows an example where $k = 8$ and k-mers 'CTNANGTC' and 'AAAGTTGG' have been mapped by the hashing function into the same index (index 4 529 821). As usual in hashing schemes, disambiguation is achieved by direct comparison of the k-mers that originated the index, which have to be stored in the data structure in order to be compared. Using the hash table, it is easy to find in the reference sequence the characters that come right after all occurrences of a given k-mer.

Before encoding a new character from the target sequence, the performance of the copy model, if in use, is checked. If $t_n - h_n^1 - h_n^2 > m_f$, where m_f is a parameter that indicates the maximum number of prediction failures allowed, the copy model is stopped. The default value for m_f is zero, but this may be changed through a command line option.

Following this performance check, if the copy model is not in use, an attempt is made to restart the copy model before compressing the character. This is accomplished by looking for the positions in the reference sequence where the k-mer composed of the k-most-recently-encoded characters occurs. If more than one position is found, the one closest to the encoding position is chosen. If none is found, the current character is encoded using the static model and a new attempt for starting a new copy model is performed after advancing one position in the target sequence.

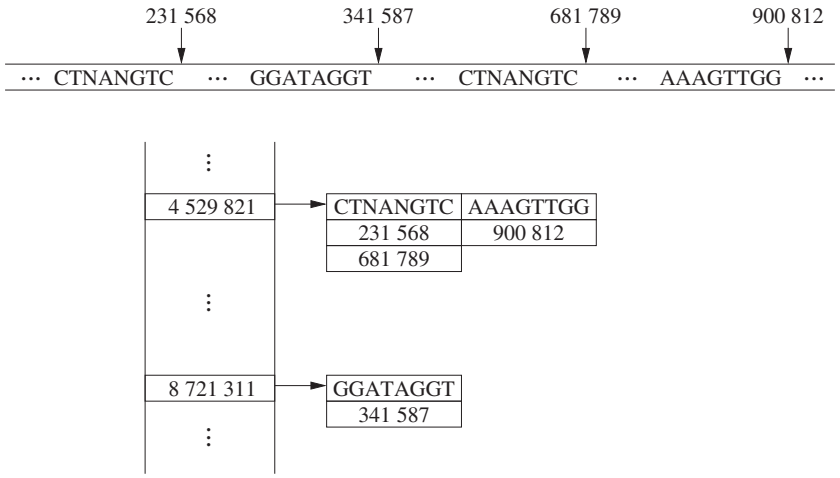

Figure 2. Data organized in a hash table.

Special case for equal size sequences. When the reference and target sequences have the same size, the codec assumes that both sequences are aligned. Therefore, whenever the copy model is restarted, it is forced to use the current encoding position as reference. This avoids constructing the hash table, hence, increasing the codec speed and generally producing better results. However, this mode of operation may be overridden by a command line option, as it may lead to poor performance for same-sized sequences that are not aligned.

RESULTS

We compare the performance of the method proposed here, GReEn, to the performance of GRS (22), the most recently proposed approach for compressing genome resequencing data that handles sequences drawn from arbitrary alphabets. We also include results of the RLZ algorithm (23) for some sequences, due to its restriction to sequences drawn from the alphabet $\{a, c, g, t, n\}$.

Tables 1–3 present both the number of bytes produced and the time taken by the respective methods for compressing the sequences used in (22). The results regarding both the GRS and RLZ methds have been obtained using the software publicly provided by the authors. All experimental results were obtained using an Intel Core i7-2620M laptop computer at 2.7 GHz with 8 GB of memory and running Ubuntu 11.04. The best results are highlighted in boldface.

Table 1 displays the compression results for the TAIR9 version of the thale cress genome using the TAIR8 version as reference. Globally, GReEn required 6559 bytes for storing the sequences, whereas GRS needed a little more (6644 bytes). While GReEn took 48 s to encode the data, GRS needed only 28. Therefore, in this case, GRS is equivalent to the proposed method in terms of storage space, but faster.

Table 2 displays the compression results for the TIGR6.0 version of the rice genome using the TIGR5.0 version as reference. In this case, the outcome varies dramatically to the previous results (Table 1). The first

Table 1. *Arabidopsis thaliana* genome: compression of TAIR9 using TAIR8 as reference

Chr	$\|\mathcal{C}\|$	Size	GRS		GReEn	
			Bytes	Secs	Bytes	Secs
1	11	30 427 671	715	7	1551	13
2	11	19 698 289	385	4	937	8
3	10	23 459 830	2989	6	1097	9
4	7	18 585 056	1951	5	2356	7
5	5	26 975 502	604	6	618	11
Total	–	119 146 348	6644	28	6559	48

Size of the compressed target sequences (in bytes) and corresponding compression time (in seconds). The original sequence alphabets have been preserved. The $|\mathcal{C}|$ column indicates the size of the alphabet of the target sequence.

significant difference can be observed in both the compressed size and compression time of chromosome 1 : 1 502 040 bytes in 708 s using GRS versus 4972 bytes in 18 s (more than a 300-fold improvement) using GReEn. A similarly significant difference can be observed in chromosome 11 (with a gain of over 160-fold). Globally, GRS required 4 901 902 bytes and 2290 s, whereas GReEn was able to store the entire genome in just 125 535 bytes (39-fold improvement) using only 123 s of computing time.

The main conclusion from these results is that, under certain conditions not yet investigated, GRS fails to find large-scale similarities between the two sequences. Therefore, the number of bytes generated is much larger than necessary and, probably as a consequence, the running time explodes. Moreover, when the target sequence is exactly equal to the reference sequence (as in chromosomes 6, 9 and 12), the GRS reports a number of bytes that is essentially zero [in (22) they are shown as zero, although we opted to display the number of bytes effectively used], while GReEn uses a few hundred bytes. However, if critical, this could be easily reduced to almost zero using a sequence comparison before starting encoding

Table 2. *Oryza sativa* genome: compression of TIGR6.0 using TIGR5.0 as reference

Chr	$\|\mathcal{C}\|$	Size	RLZ		GRS		GReEn	
			Bytes	Secs	Bytes	Secs	Bytes	Secs
1	5	43 268 879	185 715	35	1 502 040	708	4 972	18
2	5	35 930 381	210 295	28	1 409	5	1 906	14
3	6	36 406 689	–	–	47 764	28	17 890	15
4	5	35 278 225	175 663	27	36 145	20	6 750	14
5	5	29 894 789	120 625	21	6 177	5	5 539	12
6	5	31 246 789	61 038	23	14	4	482	2
7	5	29 696 629	167 822	21	4 067	8	2 448	12
8	5	28 439 308	109 608	20	118 246	43	9 507	11
9	5	23 011 239	44 953	16	14	4	366	2
10	9	23 134 759	–	–	788 542	339	60 449	9
11	11	28 512 666	–	–	2 397 470	1 122	14 797	12
12	5	27 497 214	53 714	19	14	4	429	2
Total	–	372 317 567	–	–	4 901 902	2 290	125 535	123

Size of the compressed target sequences (in bytes) and corresponding compression time (in seconds). The original sequence alphabets have been preserved. The $|\mathcal{C}|$ column indicates the size of the alphabet of the target sequence. The missing RLZ values correspond to sequences with characters that cannot be handled by the current implementation of this algorithm.

(note that, due to the requirement that the probabilities communicated to the arithmetic coder should be represented as integers, a lower bound exists in the minimum number of bits that can be generated in each coding step).

Table 3 displays the compression results for the KOREF_20090224 version of the human genome using the KOREF_20090131 version as reference. In this case, GReEn gives consistently better results, both in terms of storage requirements and computing time. In fact, this latter aspect deserves a special note because contrarily to GRS, the running time of GReEn varies linearly with the size of the sequence. Therefore, GReEn allows for an *a priori* good estimate of the time that is required to compress a given sequence.

Besides considering the datasets in (22), we also investigate four human genome assemblies, in order to provide a more comprehensive comparison of both GRS and GReEn compression approaches. However, our intention fell short because GRS failed to compress most of the sequences due to an excessive difference between the reference and target sequences. Table 4 displays the results obtained when the YH genome was compressed using KOREF_20090224 as reference. It is clear that GRS gave unacceptable results, both regarding the size of the compressed sequences and the time required to compress them, for the few chromosomes that could be compressed with GRS.

Table 5 displays the compression results, using GReEn, for four different human genome assemblies (HuRef, Celera, YH and KOREF_20090224) using the NCBI37 version as reference. As this article is about sequence compression, not sequence analysis, we refrain from elaborating too much on the differences observed. Nevertheless, we hint at what we believe may be possible explanations. First, the HuRef and Celera assemblies are not resequencing assemblies and this, *per se*, accounts for

Table 3. *Homo sapiens* genome: compression of KOREF_20090224 using KOREF_20090131 as reference

Chr	Size	GRS		GReEn	
		Bytes	Secs	Bytes	Secs
1	247 249 719	1 336 626	222	1 225 767	32
2	242 951 149	1 354 059	230	1 272 105	31
3	199 501 827	1 011 124	165	971 527	26
4	191 273 063	1 139 225	193	1 074 357	25
5	180 857 866	988 070	173	947 378	23
6	170 899 992	906 116	146	865 448	22
7	158 821 424	1 096 646	167	998 482	20
8	146 274 826	764 313	125	729 362	19
9	140 273 252	864 222	134	773 716	18
10	135 374 737	768 364	122	717 305	17
11	134 452 384	755 708	119	716 301	17
12	132 349 534	702 040	114	668 455	17
13	114 142 980	520 598	87	490 888	15
14	106 368 585	484 791	81	451 018	14
15	100 338 915	496 215	79	453 301	13
16	88 827 254	567 989	91	510 254	11
17	78 774 742	505 979	81	464 324	10
18	76 117 153	408 529	71	378 420	10
19	63 811 651	399 807	62	369 388	8
20	62 435 964	282 628	48	266 562	8
21	46 944 323	226 549	40	203 036	6
22	49 691 432	262 443	41	230 049	6
M	16 571	183	1	127	1
X	154 913 754	3 231 776	500	2 712 153	20
Y	57 772 954	592 791	96	481 307	7
Total	3 080 436 051	19 666 791	3,188	17 971 030	396

Size of the compressed target sequences (in bytes) and corresponding compression time (in seconds). The original sequence alphabets have been preserved. The size of the alphabet in the target sequence is 21 for all chromosomes, except for the M chromosome where it is 11.

greater compression differences with respect to the reference assembly.

The HuRef assembly is an individual genome sequenced with capillary-based whole-genome shotgun technologies and *de novo* assembled with the Celera Assembler. Hence, this assembly is the farthest apart (i.e. with a larger number of bytes required for its compression) from the reference NCBI37 assembly.

The Celera assembly represents one of the two pioneering efforts in sequencing a human genome. Its consensus sequence is derived from the genomes of five individuals using a capillary-based whole-genome shotgun sequencing approach. Unlike the reference assembly generated by the International Human Genome Sequencing Consortium (here represented in the NCBI37 assembly), which used a clone-based hierarchical shotgun strategy that is more likely to output a high-quality finished genome sequence as the sequence assembly is local and anchored to the genome, the Celera Genomics Sequencing Team opted for a whole-genome shotgun strategy where sequence contigs and scaffolds must be individually anchored to the genome, rendering assembly more complex and more prone to long-range misassembly. Moreover, this whole-genome shotgun assembly resulted from a combined analysis of the genomic data generated by the Celera Genomics Sequencing Team and some data generated by the International Human Genome Sequencing

Table 4. *Homo sapiens* genome: compression of YH using KOREF_20090224 as reference

Chr	Size	GRS		GReEn	
		Bytes	Secs	Bytes	Secs
1	247 249 719	–	–	2 349 124	22
2	242 951 149	–	–	2 420 007	22
3	199 501 827	17 410 946	2879	1 730 477	18
4	191 273 063	–	–	1 877 056	17
5	180 857 866	–	–	1 792 278	16
6	170 899 992	25 815 446	7526	1 588 739	15
7	158 821 424	–	–	1 820 425	14
8	146 274 826	–	–	1 358 770	13
9	140 273 252	–	–	1 476 495	13
10	135 374 737	–	–	1 353 193	12
11	134 452 384	–	–	1 274 433	12
12	132 349 534	16 136 610	2120	1 174 966	12
13	114 142 980	11 227 954	3181	866 266	10
14	106 368 585	–	–	826 672	10
15	100 338 915	–	–	892 429	9
16	88 827 254	–	–	1 015 246	8
17	78 774 742	–	–	864 710	7
18	76 117 153	13 187 892	4061	713 787	7
19	63 811 651	–	–	589 422	6
20	62 435 964	8 409 776	1449	493 404	6
21	46 944 323	726 269	664	374 383	4
22	49 691 432	–	–	444 932	5
M	16 571	321	1	127	1
X	154 913 754	–	–	3 258 188	11
Y	57 772 954	–	–	859 688	4

Size of the compressed target sequences (in bytes) and corresponding compression time (in seconds). The original sequence alphabets have been preserved. The missing values are due to the inability of GRS to compress sequences differing more than a predefined value.

Table 5. *Homo sapiens* genome: compression with GReEn of the HuRef, Celera, YH and KOREF_20090224 versions using the NCBI37 as reference

Chr	HuRef	Celera	YH	KOREF
1	6 652 184	5 106 720	1 979 661	2 074 258
2	4 109 606	3 271 105	2 205 102	1 833 388
3	1 718 683	1 125 544	2 868 462	2 808 941
4	2 440 255	1 675 878	1 815 309	1 844 448
5	2 084 630	1 962 869	1 327 235	1 289 709
6	1 926 853	1 846 101	1 460 666	1 436 168
7	2 216 643	2 345 859	1 381 234	1,511 664
8	1 755 512	1 084 584	1 323 845	1 310 275
9	3 939 856	2 906 969	1 049 456	1 152 997
10	2 235 388	2 025 459	1 075 899	1 237 129
11	1 565 536	1 459 854	1 068 335	1 104 478
12	1 495 696	1 559 635	1 199 709	1 260 183
13	4 429 154	3 023 681	1 065 006	1 052 608
14	3 480 676	2 325 885	803 902	854 166
15	3 358 239	2 944 889	946 244	958 050
16	1 848 172	2 319 629	747 166	802 956
17	1 091 917	1 163 879	955 918	905 359
18	893 600	625 364	726 165	765 927
19	697 898	621 943	2 777 894	2 832 746
20	611 521	433 253	468 215	490 498
21	884 601	415 412	434 679	481 691
22	929 001	655 089	404 354	431 417
X	3 159 205	3 259 716	492 893	740 530
Y	565 746	1 157 801	138 838	279 461

Number of bytes after compressing each sequence. For ease of comparison we transformed all characters to lowercase and mapped all unknown nucleotides to 'n' before compression. Therefore, after this transformation, all sequences were composed only of characters from the alphabet {a,c,g,t,n}.

Consortium, hence, it has been claimed that this Celera assembly is not a totally independent human genome assembly (37). We believe this may be part of the explanation for the smaller compression values in Table 5 with respect to this assembly, than those of the HuRef assembly.

The YH assembly is an individual genome based on resequencing data from massively parallel sequencing technologies and assembled with the Short Oligonucleotide Alignment Program, using the NCBI human genome assembly as reference. Essentially, it is a map of SNPs with respect to the reference assembly, hence it displays very low compression values in Table 5.

The KOREF_20090224 assembly is also an individual genome based on resequencing data from massively parallel sequencing technologies and assembled with the Mapping and Assembly with Qualities program, using the NCBI human genome assembly as reference. As with the YH assembly, resequencing renders the resulting assembly very redundant with respect to the reference (NCBI37) assembly, hence also displaying very low compression values in Table 5.

The compression values for chromosome 19 in the YH and KOREF_20090224 assemblies are unexpectedly high. This chromosome has the highest GC content (48.4%) and the lowest (median) sequence depth (28-fold) in the YH genome (34), hence constraining the quality of the final sequence. Not surprisingly, chromosome 19 in the YH genome has a very large number (more than twice those of the reference NCBI37 assembly) of unsequenced bases ('N' symbols in our encoding). Chromosome 19 in the KOREF_20090224 assembly faces the same hurdles, which we assume to be a consequence of the similar sequencing methodology.

Finally, Table 6 displays again the compression results for the KOREF_20090224 version of the human genome using the KOREF_20090131 version as reference. However, for allowing the comparison of GReEn to GRS and RLZ on a larger genome, we converted the sequences to the {a,c,g,t,n} alphabet.

DISCUSSION

The GRS tool recently introduced by Wang *et al.* (22) for compressing DNA resequencing data using a reference sequence allows to significantly reduce data storage space requirements. However, this tool seems to be effective only when the target sequence is very similar to the reference sequence, preventing the compression of many sequences of interest. Moreover, as we have shown, for example, in chromosomes 1 and 11 of the TIGR6.0 version of the rice genome, it may fail to give reasonable results even for similar sequences. Another drawback of GRS is that the encoding time does not depend only on

Table 6. *Homo sapiens* genome: compression of KOREF_20090224 using KOREF_20090131 as reference

Chr	RLZ	GRS	GReEn
1	591 629	152 388	90 555
2	576 769	146 754	89 440
3	472 814	117 544	72 708
4	471 157	134 628	83 611
5	428 287	108 407	66 597
6	411 404	109 866	67 264
7	395 524	119 223	71 898
8	350 337	94 139	56 650
9	357 584	119 647	68 607
10	335 464	101 486	60 303
11	326 836	91 380	54 966
12	320 444	89 170	55 408
13	266 378	64 313	36 962
14	248 165	58 865	34 245
15	235 094	56 569	32 693
16	217 748	60 580	35 315
17	193 700	55 582	33 836
18	182 604	48 098	29 191
19	162 826	53 355	30 505
20	149 403	38 114	22 969
21	112 822	29 048	16 620
22	119 791	32 562	18 423
M	56	75	54
X	428 878	224 997	129 497
Y	150 901	61 306	33 312
Total	7 506 615	2 168 096	1 291 629

Number of bytes after compressing each sequence. For allowing the comparison to RLZ and GRS, all characters were transformed to lowercase before compression and all unknown nucleotides were mapped to 'n'. Therefore, after this transformation, all sequences were composed only of characters from set {a,c,g,t,n}.

the sequence size, but mainly on the similarity between the target and reference sequences (lower similarity implying greater compression times), resulting in a large unpredictability regarding the time that a certain sequence requires to be compressed.

To overcome these limitations, we propose a statistical compression method that uses a probabilistic copy model. The probabilities are estimated for every character of the target sequence and are used to feed an arithmetic coder. The compression tool has two control parameters, namely, the size of the k-mer that is used for searching copies (with a default value of $k = 11$), and the number of prediction failures that are tolerated by the copy model before it is restarted (with a default value of 0). Changing these parameters may change the performance of the codec, degrading the performance for some sequences while improving it for others. It is left to the user the decision of trying to optimize these parameters or, as we have done when producing the experimental results included in this article, to use the default values.

CONCLUSION

In this article, we described a computational tool, GReEn, aiming at compressing genome resequencing data using another sequence as reference. This tool is able to handle arbitrary alphabets and does not pose any restrictions or requirements on the sequences to compress. Several examples of its efficiency in compressing genomic data and its improvements with respect to other recently proposed tools have been included, rendering evident the practical interest of the tool here proposed.

With the generation of increasingly larger volumes of genome sequencing and resequencing data, and the increasing costs associated to storing and transmitting those data, compression tools that efficiently recognize redundancies are in demand. However, the interest in such compression methodologies goes beyond data storage and communication. By being a probabilistic model of the underlying genomic sequence(s), compression tools reveal similarities and differences that are paramount for studies of human genomic variation between individuals, hence, key for progress in personal medicine efforts.

FUNDING

European Fund for Regional Development (FEDER) through the Operational Program Competitiveness Factors (COMPETE); Portuguese Foundation for Science and Technology (FCT) through project grants FCOMP-01-0124-FEDER-010099 (FCT reference PTDC/EIA-EIA/103099/2008) and FCOMP-01-0124-FEDER-022682 (FCT reference PEst-C/EEI/UI0127/2011); European Social Fund (to S.P.G.); Portuguese Ministry of Education and Science (to S.P.G.). Funding for open access charge: Portuguese Foundation for Science and Technology (FCT) project grant FCOMP-01-0124-FEDER-022682 (FCT reference PEst-C/EEI/UI0127/2011).

Conflict of interest statement. None declared.

REFERENCES

1. Grumbach,S. and Tahi,F. (1993) Compression of DNA sequences. In *Proceedings of the Data Compression Conference, DCC-93*. IEEE, Snowbird. Utah, pp. 340–350.
2. Grumbach,S. and Tahi,F. (1994) A new challenge for compression algorithms: genetic sequences. *Inform. Process. Manag.*, **30**, 875–886.
3. Rivals,E., Delahaye,J.-P., Dauchet,M. and Delgrange,O. (1996) A guaranteed compression scheme for repetitive DNA sequences. In *Proceedings of the Data Compression Conference, DCC-96*. IEEE, Snowbird. Utah, p. 453.
4. Loewenstern,D. and Yianilos,P.N. (1997) Significantly lower entropy estimates for natural DNA sequences. In *Proceedings of the Data Compression Conf., DCC-97*. IEEE, Snowbird. Utah, pp. 151–160.
5. Chen,X., Kwong,S. and Li,M. (1999) A compression algorithm for DNA sequences and its applications in genome comparison. In Asai,K., Miyano,S. and Takagi,T. (eds), *Genome Informatics 1999: Proc. of the 10th Workshop*. Universal Academy Press, Inc, Tokyo, Japan, pp. 51–61.
6. Matsumoto,T., Sadakane,K. and Imai,H. (2000) Biological sequence compression algorithms. In Dunker,A.K., Konagaya,A., Miyano,S. and Takagi,T. (eds), *Genome Informatics 2000: Proceedings of the 11th Workshop*. Tokyo, Japan, pp. 43–52.
7. Chen,X., Kwong,S. and Li,M. (2001) A compression algorithm for DNA sequences. *IEEE Eng. Med. Biol. Mag.*, **20**, 61–66.

8. Chen,X., Li,M., Ma,B. and Tromp,J. (2002) DNACompress: fast and effective DNA sequence compression. *Bioinformatics*, **18**, 1696–1698.
9. Tabus,I., Korodi,G. and Rissanen,J. (2003) DNA sequence compression using the normalized maximum likelihood model for discrete regression. In *Proceedings of the Data Compression Conference, DCC-2003*. Snowbird. Utah, pp. 253–262.
10. Manzini,G. and Rastero,M. (2004) A simple and fast DNA compressor. *Softw. Pract. Exp*, **34**, 1397–1411.
11. Korodi,G. and Tabus,I. (2005) An efficient normalized maximum likelihood algorithm for DNA sequence compression. *ACM T. Inform. Syst*., **23**, 3–34.
12. Behzadi,B. and Le Fessant,F. (2005) DNA compression challenge revisited. In *Combinatorial Pattern Matching: Proceedings of CPM-2005*, Vol. 3537 of *LNCS. Springer, Jeju Island, Korea, pp. 190–200*.
13. Korodi,G. and Tabus,I. (2007) Normalized maximum likelihood model of order-1 for the compression of DNA sequences. In *Proceedings of the Data Compression Conference, DCC-2007*. IEEE, Snowbird. Utah, pp. 33–42.
14. Cao,M.D., Dix,T.I., Allison,L. and Mears,C. (2007) A simple statistical algorithm for biological sequence compression. *Proceedings of the Data Compression Conference, DCC-2007*. IEEE, Snowbird. Utah, pp. 43–52.
15. Pinho,A.J., Ferreira,P.J.S.G., Neves,A.J.R. and Bastos,C.A.C. (2011) On the representability of complete genomes by multiple competing finite-context (Markov) models. *PLoS ONE*, **6**, e21588.
16. Giancarlo,R., Scaturro,D. and Utro,F. (2009) Textual data compression in computational biology: a synopsis. *Bioinformatics*, **25**, 1575–1586.
17. Pinho,A.J., Pratas,D. and Ferreira,P.J.S.G. (2011) Bacteria DNA sequence compression using a mixture of finite-context models. In *Proceedings of the IEEE Workshop on Statistical Signal Processing*, IEEE, Nice, France.
18. Lander,E.S. (2011) Initial impact of the sequencing of the human genome. *Nature*, **470**, 187–197.
19. Venter,J.C. (2010) Multiple personal genomes await. *Nature*, **464**, 676–677.
20. Christley,S., Lu,Y., Li,C. and Xie,X. (2009) Human genomes as email attachments. *Bioinformatics*, **25**, 274–275.
21. Brandon,M.C., Wallace,D.C. and Baldi,P. (2009) Data structures and compression algorithms for genomic sequence data. *Bioinformatics*, **25**, 1731–1738.
22. Wang,C. and Zhang,D. (2011) A novel compression tool for efficient storage of genome resequencing data. *Nucleic Acids Res*., **39**, e45.
23. Kuruppu,S., Puglisi,S.J. and Zobel,J. (2011) Optimized relative Lempel-Ziv compression of genomes. In Reynolds,M. (ed.), Optimized relative Lempel-Ziv compression of genomes. *Proceeding, of ACSC 2011, 34th Australasian Computer Science Conference (ACSC 2011), Conferences in Research and Practice in Information Technology (CRPIT),* Vol. 113, Australian Computer Society Inc, Perth Australia.
24. Tembe,W., Lowey,J. and Suh,E. (2010) G-SQZ: compact encoding of genomic sequence and quality data. *Bioinformatics*, **26**, 2192–2194.
25. Deorowicz,S. and Grabowski,S. (2011) Compression of DNA sequence reads in FASTQ format. *Bioinformatics*, **27**, 860–862.
26. Fritz,M.H.-Y., Leinonen,R., Cochrane,G. and Birney,E. (2011) Efficient storage of high throughput DNA sequencing data using reference-based compression. *Genome Res*., **21**, 734–740.
27. Kozanitis,C., Saunders,C., Kruglyak,S., Bafna,V. and Varghese,G. (2011) Compressing genomic sequence fragments using SlimGene. *J. Comput. Biol*., **18**, 401–413.
28. Ahn,S.-M., Kim,T.-H., Lee,S., Kim,D., Ghang,H., Kim,D.-S., Kim,B.-C., Kim,S.-Y., Kim,W.-Y., Kim,C. *et al*. (2009) The first Korean genome sequence and analysis: full genome sequencing for a socio-ethnic group. *Genome Res*., **19**, 1622–1629.
29. Huala,E., Dickerman,A.W., Garcia-Hernandez,M., Weems,D., Reiser,L., LaFond,F., Hanley,D., Kiphart,D., Zhuang,M., Huang,W. *et al*. (2001) The *Arabidopsis* Information Resource (TAIR): a comprehensive database and web-based information retrieval, analysis, and visualization system for a model plant. *Nucleic Acids Res*., **29**, 102–105.
30. Rhee,S.Y., Beavis,W., Berardini,T.Z., Chen,G., Dixon,D., Doyle,A., Garcia-Hernandez,M., Huala,E., Lander,G., Montoya,M. *et al*. (2003) The *Arabidopsis* Information Resource (TAIR): a model organism database providing a centralized, curated gateway to *Arabidopsis* biology, research materials and community. *Nucleic Acids Res*., **31**, 224–228.
31. Ouyang,S., Zhu,W., Hamilton,J., Lin,H., Campbell,M., Childs,K., Thibaud-Nissen,F., Malek,R.L., Lee,Y., Zheng,L. *et al*. (2007) The TIGR Rice Genome Annotation Resource: improvements and new features. *Nucleic Acids Res*., **35**, D883–D887.
32. Levy,S., Sutton,G., Ng,P.C., Feuk,L., Halpern,A.L., Walenz,B.P., Axelrod,N., Huang,J., Kirkness,E.F., Denisov,G. *et al*. (2007) The diploid genome sequence of an individual human. *PLoS Biol*., **5**, 2113–2144.
33. Venter,J.C., Adams,M.D., Myers,E.W., Li,P.W., Mural,R.J., Sutton,G.G., Smith,H.O., Yandell,M., Evans,C.A., Holt,R.A. *et al*. (2001) The sequence of the human genome. *Science*, **291**, 1304–1351.
34. Wang,J., Wang,W., Li,R., Li,Y., Tian,G., Goodman,L., Fan,W., Zhang,J., Li,J., Zhang,J. *et al*. (2008) The diploid genome sequence of an Asian individual. *Nature*, **456**, 60–66.
35. The International Human Genome Sequencing Consortium (2001) Initial sequencing and analysis of the human genome. *Nature*, **409**, 860–921.
36. Rissanen,J. (1976) Generalized Kraft inequality and arithmetic coding. *IBM J. Res. Develop*, **20**, 198–203.
37. Waterston,R.H., Lander,E.S. and Sulston,J.E. (2002) On the sequencing of the human genome. *Proc. Natl Acad. Sci. USA*, **99**, 3712–3716.