2012 International Conference on Bioscience, Biochemistry and Bioinformatics
IPCBEE vol.31(2012) © (2012)IACSIT Press, Singapoore

Qamar – A More Accurate DNA Sequencing Error Correcting Algorithm

Mohammed Sahli [1], Tetsuo Shibuya [2]

[1] Department of Computer Science, Graduate School of Information Science and Technology, University of Tokyo - 7-3-1 Hongo, Bunkyo-ku, Tokyo 113-0033, Japan

[2] Human Genome Center, Institute of Medical Science, University of Tokyo - 4-6-1 Shirokanedai, Minato-ku, Tokyo 108-8639, Japan

Abstract. The current sequencing technologies, especially the next-generation sequencers, provide us with very large amounts of sequencing data that contain many errors in ordinary. Correcting these errors is critical for achieving high accuracies in various next-generation sequencing applications, such as genome assembly, genome resequencing and SNP haplotyping. Therefore, more accurate error correction methods for these sequencer reads are highly desired. In this paper, we present a new algorithm for error-correction of sequencer reads. As many of previous error correcting algorithms do, we also utilize *k*-mer frequency table, but we achieve higher accuracy by considering non-exact matching between *k*-mers. We compared our algorithm with the recent best-known error-correction algorithm to evaluate the accuracy of our algorithm. The experiments revealed that our algorithm shows higher accuracy than the previous best-known algorithm, for both the case of short reads and the case of long reads, whatever the genome sizes and coverage levels are. The source code and binary file are freely available at http://sourceforge.net/projects/dnascissor/files/Qamar/.

Keywords: Sequencing errors, genome assembly, sequencing technology.

1. Introduction

Hundreds of thousands of bacterial, viral, eukaryotic and prokaryotic genomes have been sequenced by a set of sequencing technologies such as Sanger technology [9], Illumina's Genome Analyzer, 454 sequencing technology and ABI's SOLiD [7]. On the other hand, dealing with the gigantic genome database requires some sophisticated tools in order to extract a set of specific information such as SNPs and repetitive sequences or other heavy computational tasks, e.g. sequences similarity, mapping and genome assembly. Sequencing errors have made it very difficult to deal with this enormity of data. Such errors differ from one technology to another [10]. For example, substitutions are the main error type in SOLiD and Illumina/Solexa reads, while insertions and deletions can be found in 454/Roche platform [8]. For instance, the de Bruijn graph that is used to solve the genome assembly problem, is very sensitive to sequencing errors due to its structural nature. As a result, error-correction has been considered as one of the most difficult tasks in the de Bruijn graph and the string graph. Correcting errors may lead the assembly algorithm to construct longer and more accurate contigs.

We will briefly give a review of some recent error-correction methods. Schroder [11] has formulated the problem as a generalized suffix trie in which all reads are stored along with their reverse complements plus adding leaf counts to the edges of internal nodes of the trie; this algorithm has been named SHREC. A generalization of the SHREC algorithm to mixed sets of reads was introduced in [8]. Salmela solved the problem by using the alignments to detect and correct errors; such alignments are computed by a suffix trie. The Euler-SR read correction was presented in [1]. An efficient CUDA implementation of the Euler-SR read correction was done by Shi *et al.* [12]. Besides, a *k*-spectrum approach of Euler-SR was used in Reptile [14]; it works with the spectrum of *k*-mers from the input reads and corrects errors by simultaneously examining two factors: (a) Hamming distance-based correction possibilities for potentially erroneous *k*-mers; and (b)

neighboring *k*-mers from the same read for correcting contextual information [14]. The main idea of Reptile method is to perform approximate multiple alignments with substitutions that could be created by considering all reads with pairwise Hamming distance less than some threshold. However, this threshold requires careful consideration that is not explained in Reptile paper and relying on alignments tends to correct too many false positive errors, which could be the reason why the newer algorithm HiTEC [5] performs better. HiTEC is a recent high throughput error-correction algorithm introduced in [5]. HiTEC algorithm uses a thorough statistical analysis of the suffix array built on the string of all reads and their reverse complements. However, the error-correction of these algorithms is still far from perfect and more accurate error-correction algorithms are desired. Thus in this paper, we propose an algorithm that utilizes *k*-mer table (i.e., *k*-spectrum) with some non-exact matching technique. Our algorithm is not based on suffix trie nor suffix array and does not perform approximate multiple alignments as that used in Reptile [14]. Our algorithm could be classified as a *k*-spectrum approach but it corrects erroneous *k*-mers in the same way as solving the 1-mismatch string problem. We will show that the *k*-spectrum approach can be improved by utilizing a simpler (and faster) 1-mismatch comparison algorithm, and can achieve higher accuracy than the best known HiTEC algorithm. Further explanation is given in the following section. The experimental results are shown in section 3. We will discuss the performance of our algorithm in section 4.

2. Algorithm

Our algorithm utilizes the *k*-mer table. But in ordinary the real *k*-mer table should be different from the observed *k*-mer table due to errors (substitutions) in sequence reads, and it should affect the final error correcting accuracies. Thus, we predict the real *k*-mer table by utilizing similarities between similar *k*-mers to achieve higher accuracies. But such non-exact matching takes large computation time in ordinary. We overcome this problem as follows: First of all, the algorithm derives all possible *k*-mers from the dataset. A specific data structure, called *hash table*, is used to store the *k*-mers. While deriving *k*-mers from the reads, we record two pieces of information: (1) a frequency table T_1 is used to store the *k*-mers along with their frequency values; and (2) an index table T_2 is used in which each *k*-mer should be identified by at least one read index from which it was generated. Let *L* be the total length of all reads; this step is done in linear time O(*L*). Say *N* is the number of *k*-mers obtained; the space complexity is $O(kN)$. Next, the algorithm creates another set T_3 in order to store only *k*-mers which have frequency values less or equal to a predefined frequency threshold *F*. This set represents the expected erroneous *k*-mers. The running time of this step is $O(N)$. For the next step, we consider $M = |T_3|$ such that $M<<N$. After that, we explore each *k*-mer in T_3 by applying the 1-mismatch algorithm. In order to speed up the search, the 1-mismatch string matching problem was converted into the exact matching problem. For each iteration, a *k*-mer is selected from the set T_3, the algorithm then generates some patterns from the current erroneous *k*-mer as shown in Figure 1. Say *k* is the length of a given *k*-mer, 4*k* patterns will be generated. Finding one pattern in the initial set T_1 can be done in a constant time, and the pattern that got the highest frequency value is most likely to be the corrected version of the current erroneous *k*-mer. Each erroneous *k*-mer will be recorded along with its corrected version in a table T_4. Since *M* is size of T_3, the running time is then $O(kM)$ and the space complexity of T_4 is $O(kM)$. The undetected *k*-mers are discarded in order to free the memory, and the erroneous reads are corrected by using the information in T_4 and T_2. This phase runs in $O(M)$. The pseudo-code of the algorithm is shown as follows:

```
Qamar(S, k, f)
1.  Input : S (set of reads), k (the k-mer length),
      f (the frequency).
2.  Ouput: S' (the corrected version of S).
3.  Hashtable T1,T2,T3,T4;
4.  String kmer, pat;
5.  Integer i, j, val;
6.  for i ← 1 to N do
7.    for j← 1 to |S[i]| do
8.      kmer ← S[i][j...j+k];
9.      T2←T2 ∪ {kmer,i};
10.     if ∀ kmer∈T1 then  val ← T1[kmer];
11.                        T1.insert(kmer,val+1);
12.     else               T1.insert(kmer,1);
13. for ∀ kmer ∈T1 do
14.   if  T1 [kmer] ≤ f  then  T3.insert(kmer,T1 [kmer]);
15.   else                     T2.remove(kmer,T1 [kmer]);
16. for ∀ kmer ∈T3 do  T4.insert(kmer , ∅);
17. for ∀ kmer ∈T4 do
18.  for  j←1 to k do
19.   for ∀ c ∈{A,C,G,T} do
20.     pat ← kmer.replace( j, 1, c);
21.     if T1[pat] >T3[kmer] then T4.update(kmer,pat);
22.                          T3.insert(kmer,T1 [pat]);
23. for ∀ kmer ∈T4 do i← T2[kmer];
24.                   j← S[i].indexOf(kmer);
25.              S[index] ← S[i].replace( j, k, T4[kmer]);
```

Table 1. The short read datasets used for comparison

Reference genome (ID)	Access. no.	Size (bp)
Drosophila melanogaster, Chro. 2L (D1)	AC006575	79792
Beta vulgaris genomic clone ZR-47B15 (D2)	ZR-47B15	117150
Acinetobacter sp. ADP1 Chromosome (D3)	NC_005966	3598621
Helicobacter acinonychis (D4)	NC_008229	1553927
Staphylococcus aureus. (D5)	NC_003923	2820462
E. coli str.K-12 substr.DH10B (D6)	NC_010473	4686137
E. coli str.K-12 substr. MG1655 (D7)	NC_000913	4639675
Saccharomyces cerevisiae, Chr. 5 (D8)	NC_001137	576 874
Saccharomyces cerevisiae, Chr. 7 (D9)	NC_001139	1090946
Haemophilus influenzae (D10)	NC_007146	1914490

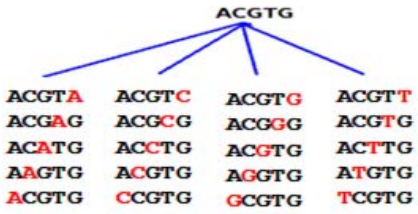

Fig. 1: Generating similar patterns of the k-mer ACGTG

Table 2.Accuracy comparison for the artificial datasets

Dataset		Accuracy					
		35 bp		70 bp		100 bp	
ID	Error %	HiTEC	Qamar	HiTEC	Qamar	HiTEC	Qamar
D7	1	98.52	99.82	99.88	99.90	99.93	99.89
D7	2	95.95	99.87	99.83	99.85	99.88	99.86
D7	3	95.79	99.82	99.76	99.88	99.84	99.87
D10	1	97.18	99.78	99.85	99.80	99.80	99.92
D10	2	96.85	99.73	99.76	99.85	99.84	99.84
D10	3	92.54	99.78	99.39	99.82	99.85	99.76
D8	1	97.64	99.86	99.91	99.81	99.86	99.86
D8	2	97.33	99.86	99.86	99.95	100	100
D8	3	94.92	99.83	99.66	99.94	99.80	99.93
D9	1	98.36	99.92	99.80	99.95	99.79	100
D9	2	96.56	99.91	99.95	99.85	99.83	99.93
D9	3	95.90	99.87	99.61	99.90	99.69	99.93

The coverage is 35x for all datasets.

Table 3. Time and space comparison of Qamar with HiTEC for the artificial datasets

ID	35 bp				70 bp				100 bp			
	Time (sec)		Space (GiB)		Time (sec)		Space (GiB)		Time (sec)		Space (GiB)	
	HiTEC	Qamar	HiTEC	Qamar	HiTEC	Qamar	HiTEC	Qamar	HiTEC	Qamar	HiTEC	Qamar
D7	192	150	3.0	1.5	224	168	3.0	1.4	118	171	3.1	1.3
D7	201	175	3.1	1.6	234	187	3.1	1.4	233	187	2.9	1.3
D7	199	200	3.1	1.7	226	307	3.0	1.4	234	206	3.0	1.3
D10	72	60	1.2	0.6	84	69	1.2	0.5	44	70	1.3	0.5
D10	71	71	1.3	0.6	83	78	1.3	0.5	86	75	1.2	0.5
D10	72	81	1.3	0.6	84	86	1.2	0.5	86	83	1.2	0.5
D8	19	18	0.4	0.2	22	19	0.4	0.2	11	20	0.4	0.2
D8	19	21	0.4	0.2	22	22	0.4	0.2	23	22	0.4	0.2
D8	19	23	0.4	0.2	22	24	0.4	0.2	23	23	0.4	0.2
D9	39	34	0.7	0.4	45	38	0.7	0.3	23	39	0.7	0.3
D9	39	39	0.7	0.4	45	42	0.7	0.3	47	42	0.7	0.3
D9	39	45	0.7	0.4	45	46	0.7	0.3	47	46	0.7	0.3

Table 4. Accuracy comparison between Qamar and HiTEC for several real datasets of Illumina/Solexa reads of a variety of read lengths, coverage levels and error rates.

Dataset	Algorithm	TP	FP	FN	TN	Specificity %	Sensitivity %	Accuracy %	Time (sec)	Space (GB)
NC_001137	Qamar	318890	160	174864	658122	99.98	64.58	64.55	229	1.0
	HiTEC	318600	3258	178249	656261	99.51	64.12	63.47	58	0.7
AC006575	Qamar	80485	7	1312	418342	100	98.40	98.39	48	0.2
	HiTEC	81077	575	1969	417619	99.86	97.63	96.94	19	0.3
SRX001814 (NC_005966)	Qamar	5084914	535625	5783457	11000610	95.36	46.79	41.86	1385	8.6
	HiTEC	--	--	--	--	--	--	--	--	--
ZR-47B15	Qamar	733144	10447	524471	1905949	99.45	58.30	57.47	1144	1.5
	HiTEC	606622	4141	437242	1888891	99.78	58.11	57.72	330	1.4
NC_010473	Qamar	581684	6176	205156	4476683	99.86	73.93	73.14	929	2.3
	HiTEC	551197	1946	205038	4480574	99.96	72.89	72.63	662	3.5

NC_008229	Qamar	4786865	212967	4288902	6378900	96.77	52.74	50.40	2282	10.6
	HiTEC	4762782	20295	2471552	5570231	99.64	65.84	65.56	2655	7.7
NC_003923	Qamar	832788	2172	16716	2579104	99.92	98.03	97.78	975	2.0
	HiTEC	829635	12450	35439	2582248	99.52	95.90	94.46	409	2.3
SRR001665_1 (NC_000913)	Qamar	1221410	1593	113405	9076649	99.98	91.50	91.38	2331	3.5
	HiTEC	1202862	3585	125478	9076381	99.96	90.55	90.28	1542	6.9
SRR001665_2 (NC_000913)	Qamar	1616109	1964	149967	8645620	99.98	91.51	91.40	3528	4.5
	HiTEC	1623282	3950	151483	8643873	99.95	91.46	91.24	2252	6.9

Table 5. The results before and after correction

Genome	**Coverage**	**Error rate %**	**Mapped**						**Unmapped**		**Identity %**	
			All		**Identity >95%**		**Identity >100%**					
			before	**after**	**before**	**after**	**before**	**after**	**before**	**after**	**before**	**after**
Brucella suis 1330	7	0.43	37371	37376	37326	37340	11566	15338	161	156	99.70	99.77
Shewanella oneidensis MR1	7	5.16	72730	72737	72684	72710	45798	58556	3955	3948	99.78	99.91
Staphylococcus aureus COL	10	2.47	59942	59953	59699	59771	35825	47349	1520	1495	99.71	99.86
Staphylococcus epidermidis RP62A	14	0.83	59989	59999	59944	59968	29168	38817	500	490	99.81	99.89
Wolbachia sp	14	41.76	17270	17278	17254	17262	10474	12339	12381	12373	99.86	99.91
Coxiella burnetii rsa 493	13	0.78	39366	39368	39352	39357	11642	15497	308	306	99.71	99.79
Chlamydophila caviae gpic	9	6.21	18709	18716	18641	18675	9193	12577	1239	1232	99.68	99.84
Campylobacter jejuni rm1221	8	0.17	24635	24647	24585	24606	6742	11039	43	31	99.59	99.74
Erythrobacter litoralis HTCC2594	7	0.08	29897	29904	29887	29899	22228	23776	24	17	99.93	99.95
Giraffe coronavirus US/OH3/2003	8	0.23	427	428	427	428	135	256	1	0	99.58	99.82
Calf-giraffe coronavirus us oh3 2006	11	0.56	534	534	534	534	139	272	3	3	99.56	99.81

3. Results

HiTEC was the most current and best-known error corrector at the date of writing this paper. It shows significant results compared to previous work. We evaluated Qamar on several Illumina/Solexa and Sanger datasets. We compared the results with HiTEC version 1.0.2 on real Sanger datasets and both simulated and real Illumina/Solexa datasets. In order to have a clear comparison, we omitted reads with unknown characters denoted by 'N' or '.' although Qamar does not have this limitation. Since HiTEC does not treat Sanger reads we considered another experimental strategy to illustrate the performance of Qamar when dealing with Sanger reads. All the experiments were run on an Intel(R) Core(TM) i7 CPU computer with eight cores operating at 2.93 GHz, 11.8 GB of memory, and Kernel Linux 2.6.32-64-generic operating system (Ubuntu). Qamar was implemented in C/C++ and compiled and run with a cross-platform application framework called Qt version 4.6.2 (64 bit).

3.1. Illumina/Solexa reads

Concerning short reads, we compared Qamar with HiTEC version 1.0.2. The last seven datasets in Table 1 were used for HiTEC. The three first datasets are new. The two first datasets along with the fourth one are available from sharcgs.molgen.mpg.de/download.shtml. The first dataset is a set of simulated 30mer reads with an error rate of 0.6% per base used to test SHARCGS [2]. Both the second and the fourth datasets were initially used in [3] and [2]. Note that AC006575 and ZR-47B15 are just identifiers of the two first datasets and are not their accession number in the GenBank. The third dataset was downloaded from http://trace.ddbj.nig.ac.jp/DRASearch and its accession number is SRX001814; its genome reference was downloaded from GenBank under the accession number NC_005966. The fifth one was used in [11] and previously used in [4] and it was downloaded from www.genomic.ch/edena.php. The sixth dataset was used as new data for HiTEC [5] and it is available as an example of NGS data from the CLCbio website, clcbio.com/index.php?id=1290. We considered two sets of reads of *E.coli* (D7). The first set has accession number SRR001665_1 and it was used in [14] and [5] while the second one which has the accession number SRR001665_2 is a new dataset in this paper. The remaining datasets were used for HiTEC as well. Similar to

HiTEC paper [5], we simulated artificial datasets for D7, D8, D9 and D10. Since 35 bp and 100 bp are the typical read lengths of Illumina/Solexa reads, we therefore considered three read lengths 35, 70 and 100 for the simulated data. We considered very low coverage level 35x and high error rates 1%, 2% and 3%. We evaluated the performance of HiTEC and Qamar using this configuration; the results are shown in Table 2 and Table 3. Three measures should be taken into account for assessing the quality of error-correction: Specificity, Sensitivity and Accuracy. We define these measures as they were mentioned in Reptile paper [14]. We say *Sensitivity* = *TP*/(*TP*+*FN*) and *Specificity* = *TN*/(*TN*+*FP*) such that *FP* (false positive) is the correct reads that were wrongly changed, *FN* (false negative) is the number of erroneous reads that were left unchanged while *TP* (true positive) is the number of erroneous reads that were corrected and *TN* (true negative) is the number of correct reads that were left unchanged. We define *Accuracy* as in HiTEC paper [5]; we say then *Accuracy* = (*TP*−*FP*)/(*TP*+*FN*). Since HiTEC shows good results in comparison with previous works, therefore, making a comparison with HiTEC gives significant results. For mapping the reads to their corresponding reference genome, we used RMAP [13] as was previously done. Ilie et al. [5] allowed 3 mismatches for HiTEC, while 10 mismatches and 15 mismatches were allowed for Reptile [14]. However, in order to have clear comparison and to be more strict we considered only 1 mismatch. As a result, the SNPs can be matched along with few erroneous nucleotides:

./rmap -c chromosome_dir -w <read_length> -m 1 -v -o <output_file> <reads_file> -a <ambigs_file>

In all cases, HiTEC required more memory spaces compared to Qamar (Table 2 and Table 3). Our algorithm used almost half or less of what has been used by HiTEC. Furthermore, in a speed challenge between the two algorithms; the difference was only a few negligible seconds. When the read length is 35 bp, the accuracy of Qamar was higher than that of HiTEC in all cases. Concerning the read length 75 bp, Qamar showed very high accuracy in all cases, except three in which HiTEC was the winner. On the other hand, when the read length is quite long, 100 bp, Qamar achieved good results and won in six cases and lost three cases against HiTEC. Qamar uses three parameters: the *k*-mer length *L*, the minimum frequency *F* and number of iterations *N*. Concerning the simulated datasets, all the parameters of Qamar are given in the Appendix. On the other hand, we let HiTEC perform in its best parameters (the genome size and the error rate). HiTEC automatically adjusts its number of iterations. It used two iterations in almost all datasets. One last thing, Qamar performed better than HiTEC in cases of lower coverage (35x for the simulated data).

Table 4 illustrates the results of some real datasets. Qamar's accuracy was higher than HiTEC's in most cases. Our algorithm won seven cases and lost only two. In the first three datasets Qamar used only one iteration, whereas HiTEC ran in three iterations and, due to the large size of SRX001814 dataset, HiTEC could not fit in our memory. Although Qamar could process SRX001814 dataset, the accuracy was a bit lower when Qamar ran in one iteration. Concerning the last three datasets, Qamar used 6 or 7 iterations and it was significantly more accurate than HiTEC. One point should be mentioned here; when the error rate and the genome size of each datasets are known, HiTEC could then run perfectly. We noticed that HiTEC was sensitive to the predetermination of the error rate when it was mistaken in our experiments. However, in order to have a clear comparison, we provided HiTEC by correct parameters.

3.2. Sanger reads

The five first datasets in Table 5 are AMOS benchmark datasets; they are available from http://www.cbcb.umd.edu/research/benchmark.shtml. The other datasets were downloaded from the NCBI Trace Archive ftp://ftp.ncbi.nih.gov/pub/TraceDB/. Sanger reads are usually characterized by low quality regions at their 3'-end and 5'-end. They may also contain some part of the cloning vector at their 3'-end. We trimmed vector sequences and low quality regions using a trimming tool before processing the error-correction algorithms. The mapping tool RMAP [13] cannot deal with Sanger reads, we therefore used NUCmer, which is a part of the MUMmer system, release 3.22 [6]. NUCmer can map the reads to the genome very fast. The error rates were calculated by the aid of NUCmer before performing the error-correction. One may be surprised to notice the high error rate in Wolbachia sp (A5). As a matter of fact, these data include a high percentage of contamination from Drosophila in the sequencing library as it is mentioned in the AMOS benchmark web page. We used all anchor matches regardless of their uniqueness by allowing the parameter *-maxmatch*; the minimum length of a cluster of matches is 30:

./nucmer -o -c 30 -maxmatch "Reference" "Query"./show-coords -b -I 0 -q -T -H "delta_File" > "coords_File"

Sanger sequencing technology can provide us with very long accurate DNA reads. However, some erroneous nucleotides may be found in almost every read. The goal of the error correcting algorithm is to reduce errors as much as possible. In Table 5, the reads, whose identity values are 100%, contain no erroneous nucleotide. After applying Qamar, the identity values and the overall identity have increased significantly. The *k*-mer length was 30, the frequency value was 1 and the correction was done in only one iteration.

4. Conclusions

In contrast to other error-correcting algorithms, Qamar can deal with long reads of different lengths and it is adapted to handle short reads, too, as in the case of Illumina/Solexa reads. Moreover, our implementation can handle reads with unknown nucleotides (e.g. 'N') as well. Although 21 was the most reliable *k*-mer length, it should be adjusted sometimes when dealing with some huge amounts of data.

5. Acknowledgements

This work was partially supported by the Grant-in-Aid from the Ministry of Education, Culture, Sports, Science and Technology of Japan. We are grateful to Pr. Satoru Miyano (the head of Laboratory of DNA Sequence Analysis and Laboratory of Sequence Analysis, Human Genome Center, University of Tokyo) for his additional support in publishing this work. We thank Mr. Yassine Bouhmadi and Fouad Kharroubi for their corrections.

6. References

[1] M.J. Chaisson, D. Brinza and P.A. Pevzner. De novo fragment assembly with short mate-paired reads: Does the read length matter?. *Genome Res.* 2009, **19**:336–346.

[2] J.C. Dohm, C. Lottaz, T. Borodina and H. Himmelbauer. SHARCGS, a fast and highly accurate short-read assembly algorithm for de novo genomic sequencing. *Genome Research* 2007, **17**: 1697-1706.

[3] J.C. Dohm, C. Lottaz, T. Borodina and H. Himmelbauer. Substantial biases in ultra-short read data sets from high-throughput DNA sequencing. *Nucleic Acids Res* 2008.

[4] D. Hernandez, P. François, L. Farinelli, M. Østerås and J. Schrenzel. De novo bacterial genome sequencing: millions of very short reads assembled on a desktop computer. *Genome Res*. 2008, **18**:802–809.

[5] L. Ilie, F. Fazayeli and S. Ilie. HiTEC: accurate error-correction in high-throughput sequencing data. *Bioinformatics* 2010, **27**:295–302.

[6] S. Kurtz, A. Phillippy, A.L. Delcher, M. Smoot, M. Shumway, C. Antonescu and S.L. Salzberg. Versatile and open software for comparing large genomes. *Genome Biology* 2004, **5**:R12.

[7] E.R. Mardis. The impact of next-generation sequencing technology on genetics. Trends Genet. 2008, **24**:133–141.

[8] L. Salmela. Correction of sequencing errors in a mixed set of reads. *Bioinformatics* 2010, **26**:1284–1290.

[9] F. Sanger, S. Nicklen and A.R. Coulso. DNA sequencing with chain-terminating inhibitors,. *Proc. Natl Acad. Sci.* USA 1977, **74**:5463–5467.

[10] J. Shendure and H. Ji. Next-generation DNA sequencing. *Nat. Biotechnol.* 2008, **26**:1135–1145.

[11] J. Schroder, H. Schroder, S.J. Puglisi, R.S. Sinha and B. Schmidt. SHREC: a short-read error-correction method. *Bioinformatics* 2009, **25**:2157–2163.

[12] H. Shi, B. Schmidt, W. Liu and W. Müller-Wittig. A parallel algorithm for error-correction in high-throughput short read data on CUDA-enabled graphics hardware. *J. Comput. Biol.* 2010, **17**:603–615.

[13] A.D. Smith, Z. Xuan and M.Q. Zhang. Using quality scores and longer reads improves accuracy of Solexa read mapping. *BMC Bioinformatics* 2008, **9**:128.

[14] X. Yang, S.K. Dorman and S. Aluru. Reptile: representative tiling for short read error-correction. *Bioinformatics* 2010, **26**:2526–2533.