BIOINFORMATICS HITSEQ PAPER

Vol. 26 no. 10 2010, pages 1284–1290
doi:10.1093/bioinformatics/btq151

Sequence analysis

Advance Access Publication April 8, 2010

Correction of sequencing errors in a mixed set of reads

Leena Salmela

Department of Computer Science, PO Box 68 (Gustaf Hällströmin katu 2b), FI-00014 University of Helsinki, Finland

Associate Editor: Alex Bateman

ABSTRACT

Motivation: High-throughput sequencing technologies produce large sets of short reads that may contain errors. These sequencing errors make *de novo* assembly challenging. Error correction aims to reduce the error rate prior assembly. Many *de novo* sequencing projects use reads from several sequencing technologies to get the benefits of all used technologies and to alleviate their shortcomings. However, combining such a mixed set of reads is problematic as many tools are specific to one sequencing platform. The SOLiD sequencing platform is especially problematic in this regard because of the two base color coding of the reads. Therefore, new tools for working with mixed read sets are needed.

Results: We present an error correction tool for correcting substitutions, insertions and deletions in a mixed set of reads produced by various sequencing platforms. We first develop a method for correcting reads from any sequencing technology producing base space reads such as the SOLEXA/Illumina and Roche/454 Life Sciences sequencing platforms. We then further refine the algorithm to correct the color space reads from the Applied Biosystems SOLiD sequencing platform together with normal base space reads. Our new tool is based on the SHREC program that is aimed at correcting SOLEXA/Illumina reads. Our experiments show that we can detect errors with 99% sensitivity and >98% specificity if the combined sequencing coverage of the sets is at least 12. We also show that the error rate of the reads is greatly reduced.

Availability: The JAVA source code is freely available at http://www.cs.helsinki.fi/u/lmsalmel/hybrid-shrec/

Contact: leena.salmela@cs.helsinki.fi

Received on March 1, 2010; revised on April 1, 2010; accepted on April 5, 2010

1 INTRODUCTION

The high-throughput sequencing machines such as SOLEXA/Illumina, Applied Biosystems SOLiD and Roche/454 Life Sciences produce millions of short reads in a single run. The reads may contain errors, which continues to present a challenge to *de novo* assemblers. The error rate of the reads can be reduced with trimming and by correcting the reads.

The different sequencing platforms have their own benefits and shortcomings. For example, the distribution of error types varies from one platform to another (Shendure and Ji, 2008). The dominant error type in SOLEXA/Illumina and SOLiD reads is substitution, reads produced by the 454/Roche platform tend to have many insertions and deletions because of the technology's inability to assess the length of homopolymer runs correctly and the dominant error type in Helicos reads is deletion.

(a)

		Second base			
		A	C	G	T
	A	0	1	2	3
First	C	1	0	3	2
base	G	2	3	0	1
	T	3	2	1	0

The SOLiD color code

(b)

A – G – C – C – A
2 3 0 1

Translation from bases to colors

Fig. 1. SOLiD two base color encoding.

While most sequencing platforms produce *base space* reads, i.e. the reads are sequences of bases A, C, G and T, the SOLiD platform produces reads in *color space* (Applied Biosystems Incorporated, 2008a). The SOLiD sequencer interrogates bases in overlapping pairs so that each base is sequenced twice. The pairs are coded with four colors as shown in Figure 1.

Due to different characteristics of the sequencing platforms, it is an attractive idea to combine reads produced by several platforms. This kind of mixed read sets could improve the results of both error correction and *de novo* assembly. There are not many tools that can take full benefit from a mixed set of reads especially if part of the reads are SOLiD color space reads. We present this kind of tool for error correction in this article.

Many tools for error correction of reads from second generation sequencers use the spectral alignment method first introduced in the EULER-SR assembler (Chaisson *et al.*, 2004; Pevzner *et al.*, 2001). This method first computes the spectrum of the reads which consists of all l-tuples that are frequent enough in the read set. Then a string r^* is computed for each read r such that all l-tuples in r^* are in the spectrum and the distance between r and r^* is minimized. The distance measure can be, e.g. the Hamming distance allowing only substitutions or edit distance allowing both insertions and deletions. A similar approach is taken in several error correction tools such as SOLiD Accuracy Enhancement Tool (SAET, http://solidsoftwaretools.com/gf/project/saet/) and as preprocessing in many assemblers such as ALLPATHS (Butler *et al.*, 2008) and SOAPdenovo (Li *et al.*, 2010).

In the alignment approach, multiple alignments are computed for the reads and then errors are detected and corrected based on the columns of these alignments. This approach has been used with reads from the classical Sanger sequencing (Sanger *et al.*, 1977). Examples of such tools are MisEd (Tammi *et al.*, 2003) and the preprocessing step in Arachne (Batzoglou *et al.*, 2002). The problem of this approach is that it is not feasible to compute the multiple alignments for millions of reads produced by the newer sequencing technologies. Recently, the error correction tool SHREC (Schröder *et al.*, 2009) extended this approach to SOLEXA/Illumina reads by

avoiding the computation of the alignments and traversing a space-efficient suffix trie built of the reads.

Most error correction tools are designed for a single sequencing technology although some approaches can handle base space reads from different sequencing platforms. In this article, we present enhancements to the SHREC tool that allow us to utilize both base space and color space reads from any sequencing platforms.

2 METHODS

We use the following model of DNA sequencing. We have k reads randomly sampled from a genome of length n. The length of the reads can vary. The reads may have errors with some rather small probability. The errors may be substitutions, insertions or deletions. We further assume that the errors are distributed randomly and that the coverage is sufficient so that each position of the genome is present in several reads.

As each position of the genome is sampled several times, it is possible to detect an error in a read by aligning it against other reads. If the errors are distributed randomly, most of the reads that overlap the erroneous read do not have the same error. The alignment is thus very good except for the error position in the erroneous read, and so we can use the alignment to detect and correct the error. We utilize a suffix trie to compute these alignments efficiently.

Note that if the reads are from a diploid organism, they may contain single nucleotide polymorphisms (SNPs) in addition to sequencing errors. SNPs look similar to sequencing errors in the multiple alignment as they also create columns where the reads do not agree with each other. However, a sequencing error occurs only in a few reads, while SNPs should be present in several reads.

We will extend the SHREC algorithm (Schröder *et al.*, 2009), and so next we will give an overview of the data structures and methods used by it. For now we will assume that the reads are in base space.

Let R be the set of reads and their reverse complements. The *generalized suffix trie* ST(R) is a tree that contains all the suffixes of the strings in R. We concatenate a unique symbol $1\ldots2k$ to the end of each string in R so that each suffix is unique. The edges of the tree are labeled with a character from the alphabet {A,C,G,T}. Each node may have only one child labeled with the same character. We call the concatenation of edge labels from the root to a node the *path-label* of that node. For each suffix of a string in R, there is a leaf such that the path-label of the leaf is the suffix. The *weight* of a node is the number of leaves in the subtrie rooted at that node. Note that this is exactly the number of suffixes in this subtrie. The *level* of a node is the length of the path from the root to the node.

In the top levels of the trie, almost all nodes have four children as almost all strings of length equal to the level of the children can be found in the genome. Further down in the trie almost all nodes have only one child. This happens at some level r such that 4^r is greater than the length of the genome, i.e. most strings of length $r+1$ appear at most once in the genome. If a node at this level has more than one child, it is likely that the branching is caused by a sequencing error especially if the weight of one of the children is very small indicating that the subtrie rooted at that child has only a few suffixes in it. Still deeper in the trie the weights of the nodes become too small to distinguish between erroneous and correct children.

The SHREC algorithm traverses the generalized suffix trie and identifies the erroneous children at the intermediate levels of the trie. It then attempts to correct the read suffixes that go through the erroneous child by a substitution. Insertions and deletions can be detected in the same way as substitutions and we will explain our algorithm to correct this kind of errors in the next section.

The above-mentioned methods can be easily adapted to correct a set of color space reads. The reads are now strings from the alphabet $\{0,1,2,3\}$ and the reverse complements of these reads can be formed just by reversing the string. The correction algorithm otherwise works exactly in the same way.

Correcting a combination of base space and color space reads is more intricate. If we transform a color space read to base space, a single error can change all the bases starting from the error, and thus it is not feasible to correct color space reads in base space. However, if we transform a base space read into color space, the errors remain local. As each base is used only in determining two colors in the color space representation of the read, a single error in base space can only affect two colors in color space. Thus, base space and color space reads can be corrected together in color space. If a read was originally a color space read, we will allow substituting, inserting and deleting a color when correcting the read, and if the read was originally a base space read, we will allow such color space transformations that correspond to substituting, inserting or deleting a base in the corresponding base space representation. Note that since SNPs change two colors in a color space read, SNPs are not corrected in color space reads.

∘	0	1	2	3
0	0	1	2	3
1	1	0	3	2
2	2	3	0	1
3	3	2	1	0

Fig. 2. The composite colors for each color pair.

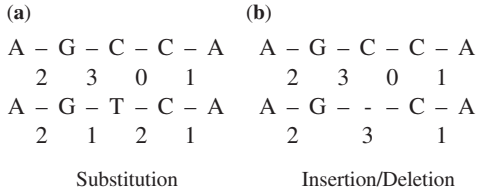


Fig. 3. The color changes caused by errors in base space. When the nucleotide C is substituted with T, two colors change. In this example, 30 is changed to 12. We note that the composite color remains the same, 3. When the nucleotide C is deleted, the two colors 30 are replaced by their composite color 3. We further note that insertion is the reverse of a deletion.

The SOLiD color code has the following algebraic interpretation (Applied Biosystems Incorporated, 2008b). A color represents a pair of bases and it can thus be seen as a function that transforms the first base of the pair to the second one. These color functions form an algebraic group where the operator is function composition. An algebraic group is closed so the composition of two colors is also a color. Figure 2 gives the compositions of all color pairs. If we think of the translation of a base space read to a color space read, then the composite function of two colors corresponds to a function that maps a base in position i to the base in position $i+2$.

The changes reflected in the color space by an error in base space can now be characterized with the composite colors. A substitution causes two colors to change but the composite color remains the same, an insertion changes a color c to two colors c_1 and c_2 such that c is the composite color of c_1 and c_2, and a deletion changes two colors to their composite color. Figure 3 shows some examples.

3 ALGORITHM

First, we will review the SHREC algorithm (Schröder *et al.*, 2009) for correcting substitutions in a set of reads of equal length. After that we will introduce our improvements to this algorithm.

3.1 The SHREC algorithm

The SHREC algorithm (Schröder *et al.*, 2009) starts by building a generalized suffix trie of the reads and then it attempts to correct the errors at the intermediate level of the trie.

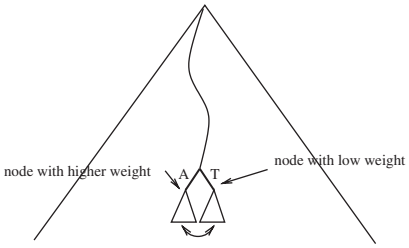


Fig. 4. The subtries to compare when correcting a substitution.

Recall that the weight of a node in the suffix trie is the number of leaves in the subtrie rooted at that node. It can be shown that if m is sufficiently large so that each substring of length m in the genome is unique, the expected weight of a node at level m of the suffix trie is $E(W_m)=ka/n$, where k is the number of reads, n is the length of the genome, $a=l-m+1$ and l is the length of a read. Furthermore, the variance of the weight of a node at level m is $\sigma^2(W_m)=k(a/n-a^2/n^2)$. The algorithm then attempts to correct nodes whose weight is below $E(W_m)-\alpha\cdot\sigma(W_m)$, where α is the *strictness* parameter of the method.

Once we have identified a node with a too small weight, we attempt to correct the error as follows. We compare the subtrie S_{lw} rooted at the low weight node to the subtries rooted at the siblings of the node, see Figure 4 for an example. If we find a sibling subtrie that contains S_{lw}, the error can be corrected by substituting the base of the erroneous node by the base of the sibling node. To transfer this correction to the reads, we find the reads whose suffixes are in the subtrie rooted at the erroneous node and correct the reads. If no such sibling subtrie is found, the reads passing through the erroneous node are marked as erroneous but no correction is made.

3.2 Statistical model for reads of varying length

We will now modify the statistical model of the SHREC algorithm to accommodate for reads of varying length. Let us assume that the read set contains reads of r different lengths and let us denote these lengths by $l_1, l_2, ..., l_r$. Let the number of reads of length l_i be k_i.

Let m be sufficiently large so that each sequence of length m appears only once in the genome. The weight W_m of a node at level m in the generalized suffix trie of the reads is the number of suffixes whose path in the trie passes through that node. We will use $W_{m,i}$ to denote the contribution of reads of length l_i to the weight of a node at level m. Thus, the weight of a node at level m is

$$W_m=\sum_{i=1}^{r}W_{m,i}.$$

Now if $l_i<m$, $W_{m,i}=0$. Otherwise each read is a Bernoulli trial for getting the path-label of the node. There are n substrings of length m in the genome and a read of length l_i samples $a_i=l_i-m+1$ of them. Thus, the probability of success for the Bernoulli trial is a_i/n. The total number of trials is k_i and so $W_{m,i}$ is distributed according to the binomial distribution $Bin(k_i, a_i/n)$. Thus, the expected value of $W_{m,i}$ is

$$E(W_{m,i})=\begin{cases} k_i a_i/n & \text{if } l_i\geq m,\\ 0 & \text{otherwise,}\end{cases}$$

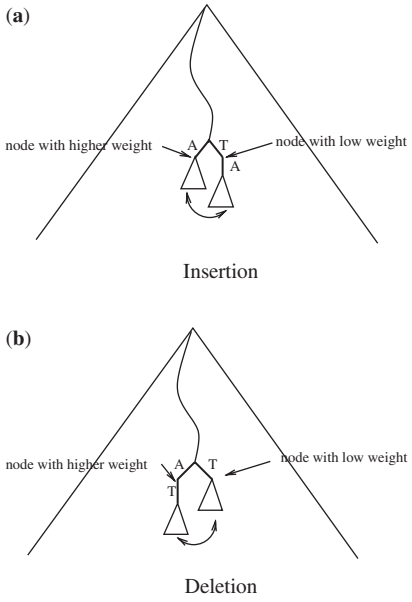


Fig. 5. The subtries to compare when correcting an insertion or deletion.

and the variance of $W_{m,i}$ is

$$\sigma^2(W_{m,i})=\begin{cases} k_i(a_i/n-a_i^2/n^2) & \text{if } l_i\geq m,\\ 0 & \text{otherwise.}\end{cases}$$

By the linearity of expectation we get

$$E(W_m)=E\left(\sum_{i=1}^{r}W_{m,i}\right)=\sum_{i=1}^{r}E\left(W_{m,i}\right).$$

If we assume that $W_{m,i}$ for different i are independent variables then

$$\sigma^2(W_m)=\sigma^2\left(\sum_{i=1}^{r}W_{m,i}\right)=\sum_{i=1}^{r}\sigma^2\left(W_{m,i}\right).$$

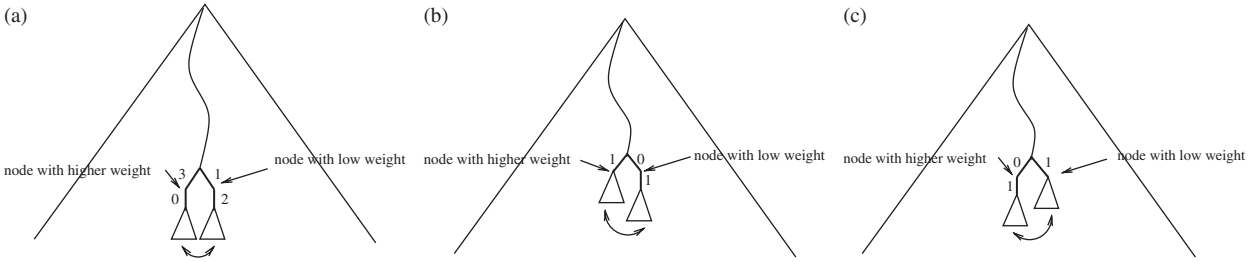
The above assumption of the independence of $W_{m,i}$ may be problematic if the length of a read depends on its genomic location.

As in the original SHREC algorithm, we try to correct nodes whose weight is below $E(W_m)-\alpha\cdot\sigma(W_m)$, where α is the strictness parameter of the method.

3.3 Correcting insertions and deletions

Just like substitutions, insertions and deletions in the reads cause extra branching in the generalized suffix trie of the reads. Figure 5a shows how the insertion of a T in a read has created a low weight node in the generalized suffix trie. The insertion can be corrected by deleting the T in which case the subtries rooted at the children of the low weight node and the corresponding siblings of the low weight node will be merged. Therefore, we compare these subtries to figure out if a deletion is a feasible way to correct a low weight node. If we find sibling subtries that contain the subtries rooted at the children of the low weight node, we have confirmed that a deletion can correct the erroneous node. We then find the reads whose suffixes are in the subtrie rooted at the erroneous node and make the deletion.

If a read contains an insertion error in a homopolymer run (e.g. one of the A's in the read GAAAATC is an insertion error), the



Substitution. The composition of the colors on the edges is the same , i.e. 30 = 12 = 3.

Insertion. The color on the correct edge is the composition of the colors on the edges adjacent to the low weight node, i.e. 1 = 01.

Deletion. The color on the edge to the low weight node is the composition of the colors on the edges adjacent to its sibling node, i.e. 01=1.

Fig. 6. The subtries to compare when correcting a substitution, insertion or deletion of a base space read in color space.

error can be corrected by deleting any one of the bases in the run. However, only the last A creates a low weight node in the suffix trie, and thus we correct the error by deleting the last base in the run.

Figure 5b shows how the deletion of A has created an extra branch in the suffix trie. This deletion can be corrected by inserting an A in the read, and then the subtrie rooted at the low weight node would be merged with the subtrie rooted at the corresponding child of one of its sibling nodes. Therefore, we compare these subtries to determine if an insertion is a feasible way to correct a low weight node. If the subtrie rooted at the low weight node is included in a subtrie rooted at a child node of the sibling nodes, we have identified an insertion that corrects the erroneous node. We then identify the reads whose suffixes are in the subtrie rooted at the erroneous node and insert the appropriate character to these reads. If a read contains a deletion error in a homopolymer run, our algorithm detects a low weight node only after the last base in the run, and thus we correct the error by inserting the missing base to the end of the run.

3.4 Correcting indeterminate bases

It is rather straightforward to adapt the SHREC method to handle indeterminate bases. We now build the generalized suffix trie using the five-letter alphabet {A,C,G,T,N}. When running the correction algorithm over the suffix trie, we always regard a node whose base is N to be erroneous. We then attempt to correct the error by a substitution and a deletion. We do not attempt to correct the node by inserting a character before N as the same could be achieved by correcting the indeterminate base by substitution.

3.5 Combining base space and color space reads

The combined correction of base space and color space reads first transforms all the base space reads into color space. Then the generalized suffix trie is built on all the reads and their reverse complements in color space. The algorithm then traverses the trie and identifies low weight nodes at the intermediate level of the trie. Color space reads whose suffixes are found in subtries rooted at low weight nodes can then be corrected exactly as outlined above.

Base space reads whose suffixes are found in subtries rooted at low weight nodes are corrected by using color transformations that correspond to substituting, deleting or inserting one base in the base space read. A substitution in base space changes two colors c_1 and c_2 into two colors c_3 and c_4 so that the composite color of c_1 and c_2 is the same as the composite color of c_3 and c_4. Therefore to correct a low weight node, we need to substitute both the color on the edge to the low weight node and the color on the edge between the low weight node and its child. To determine if this kind of substitution is a feasible way to correct a read, we compare the subtrie rooted at the child node with such children of the siblings of the low weight node that the compositions of the colors on the edges are the same. Note that because of the rules of the color space, each sibling can have only one such child. See Figure 6a for an example.

An insertion in base space replaces a color c in color space with two colors whose composite color is c. Thus, to correct a low weight node, we replace the colors on the edges adjacent to the low weight node by their composite color c. To determine whether a deletion like this is a good way to correct a base space read, we compare the subtrie rooted at the child of the low weight node with the subtrie rooted at such a sibling of the low weight node that the color on the edge to the sibling is c. See Figure 6b for an example.

A deletion in base space replaces two colors in color space with their composite color. In this case, a low weight node can be corrected by replacing the color c on the edge to the low weight node with two colors whose composite color is c. Now we compare the subtrie rooted at the low weight node with the subtries rooted at such children of the siblings of the low weight node that the composition of the colors on the edges adjacent to the sibling is c. Figure 6c shows an example of correcting a deletion.

After identifying how to correct a low weight node, we fetch the color space representations of the base space reads whose suffixes are in the subtrie rooted at that node and correct them. If the base space reads contain no indeterminate bases, we can correct them fully in color space and after correction translate them back to base space. Indeterminate bases can be handled by consulting the original base space read to figure out the composite color when we encounter indeterminate colors in the reads translated to color space and by correcting the base space read directly whenever we correct the corresponding color space read.

4 IMPLEMENTATION

We have extended the implementation of SHREC (Schröder *et al.*, 2009) with the ideas presented in the previous section. We provide two versions of the program. One can correct sets with only base space reads as the transformation to color space is an unnecessary

complication in this case. The other version corrects read sets that may include both color space and base space reads but can naturally be limited to only color space reads.

5 RESULTS AND DISCUSSION

We tested our implementation both on simulated reads and real reads from the *Escherichia coli* genome. The simulated reads are generated from the K-12 substrain MG1655 (NC_000913). The length of the simulated base space reads varies uniformly at random from 75 to 125 bp and the length of the simulated color space reads varies from 40 to 50 bp. The base space reads were generated with coverages of 6, 12 and 24, and error rates 1.5% and 3.0%. The color space reads were generated with coverages of 15 and 30, and error rates 1.4% and 2.9%. In the base space reads, each type of error is equally probable but in the color space reads only 5% of the errors are insertions or deletions as substitution is the dominant error type in reads produced by the SOLiD sequencing platform.

We downloaded real reads from the *E.coli* strain UTI89 produced by the Roche/454 sequencing platform from NCBI short read archive (accession number SRA000156) and real SOLiD reads from the *E.coli* strain DH10B from Applied Biosystems (http://solidsoftwaretools.com/gf/project/ecoli2x50/). The SOLiD reads were filtered to exclude low-quality reads. The coverage of these sets is typical for a project involving only one kind of reads. We used a subset of these reads so that the coverage would be closer to a project that uses a hybrid approach to genome assembly. The coverage of our subset of the 454 reads is 6 and the coverage of the subset of SOLiD reads is 30. The length of the 454 reads varies from 42 to 285 and the length of the SOLiD reads varies from 39 to 50. Unfortunately, we could not find base space and color space reads from the same strain of *E.coli* but nevertheless our experiments on these read sets show that there are benefits to be gained from a mixed set of real reads. Table 1 shows a summary of the read sets. To test the hybrid correction of base space and color space reads, we used a combination of the base space and color space read sets.

All the experiments were run on an Intel Xeon computer with eight cores operating at 3.66 GHz and 32 GB of memory. The algorithms are implemented in Java and compiled and run with JVM 1.6. We tried the algorithm with several values of the strictness parameter α producing different thresholds at the analyzed level of the trie and show the results for the best observed value. For the simulated read sets, the best value was determined by minimizing the number of false positives (FPs) and false negatives (FNs) when detecting erroneous reads, and for the real reads, we chose the value that allowed aligning the corrected reads best to the reference genome. Table 2 shows the runtime and memory usage of the algorithm, the best strictness value and the corresponding threshold for detecting low weight nodes at the analyzed level of the trie for the various read sets. The memory usage is measured as the peak resident set size of the program. The implementation of the algorithm uses threads. Each thread builds and analyzes a subtrie of the suffix trie where all suffixes start with a prefix specific to that thread. The memory usage of the algorithm can be decreased by reducing the number of threads or increasing the length of the prefix. However, the runtime will then increase. The strictness value must be chosen so that the threshold for detecting low weight nodes at analyzed level is greater than one. The best strictness value varied from 4 to 7 except for the lowest coverage read sets for which we were forced to choose a smaller value to keep the threshold at analyzed level greater than one.

To evaluate the performance of the algorithm, we measured its ability to detect erroneous reads and the error rate of the reads before and after correction. The detection of erroneous reads is a binary classification test. Table 3 shows the definitions for true positive (TP), true negative (TN), false negative (FN) and false positive (FP). The sensitivity is defined as TP/(TP+FN) and specificity

Table 1. Summary of read sets used in the experiments, last two read sets are real reads and the rest of the sets are simulated

ID	Base/ color space	Coverage	Error rate (%)	Number of reads (M)
B6x1.5	Base space	6	1.5	0.28
B6x3.0	Base space	6	3.0	0.28
B12x1.5	Base space	12	1.5	0.56
B12x3.0	Base space	12	3.0	0.56
B24x1.5	Base space	24	1.5	1.12
B24x3.0	Base space	24	3.0	1.12
C15x1.5	Color space	15	1.4	1.56
C15x3.0	Color space	15	2.9	1.56
C30x1.5	Color space	30	1.4	3.13
C30x3.0	Color space	30	2.9	3.13
RB6x	Base space	6		0.12
RC30x	Color space	30		2.83

Table 2. Runtime and memory usage of correction, the best value for the strictness parameter α and the corresponding threshold for detecting low weight nodes at the analyzed level of the trie for the various read sets

IDs	Runtime (s)	Memory (MB)	α	Threshold at analyzed level
B6x1.5	670	3087	2.7	1.5
B6x3.0	708	3106	2.7	1.5
B12x1.5	1391	3160	4.0	2.5
B12x3.0	1395	3379	4.0	2.5
B24x1.5	2765	3664	5.5	5.5
B24x3.0	2856	3905	5.5	5.5
C15x1.5	1245	3323	4.0	2.5
C15x3.0	1268	3457	4.0	2.5
C30x1.5	2517	3937	5.9	3.5
C30x3.0	2677	4124	5.8	4.5
B6x1.5 + C15x1.5	1788	3677	5.1	2.5
B6x3.0 + C15x3.0	1955	3826	5.1	2.5
B12x1.5 + C30x1.5	4014	4430	6.8	8.5
B12x3.0 + C30x3.0	4136	4740	7.0	7.5
RB6x	734	3002	2.9	1.5
RC30x	2467	3710	5.8	5.5
RB6x + RC30x	3296	3829	6.7	5.5

Table 3. Classifications of identified and erroneous reads

	Erroneous read	Error-free read
Identified as erroneous	TP	FP
Identified as error free	FN	TN

Table 4. The classification test for simulated read sets

ID	TP	FP	FN	TN	Sensitivity	Specificity
B6x1.5	213 075	3755	2682	60 272	0.988	0.941
B6x3.0	262 944	1584	1748	13 507	0.993	0.895
B12x1.5	429 389	861	2333	126 868	0.995	0.993
B12x3.0	527 718	483	1426	30 010	0.997	0.984
B24x1.5	859 386	59	3823	255 939	0.996	1.000
B24x3.0	1 056 625	132	2109	60 386	0.998	0.998
C15x1.5	747 968	5523	4113	806 307	0.995	0.993
C15x3.0	1 141 544	6012	4360	412 008	0.996	0.986
C30x1.5	1 497 762	18	8251	1 621 860	0.995	1.000
C30x3.0	2 285 417	347	8826	833 120	0.996	1.000
B6x1.5	214 590	17	1167	64 010	0.995	1.000
C15x1.5	747 911	91	4170	811 739	0.994	1.000
B6x3.0	264 083	12	609	15 079	0.998	0.999
C15x3.0	1 141 382	120	4522	417 900	0.996	1.000
B12x1.5	429 507	5	2215	127 724	0.995	1.000
C30x1.5	1 497 928	26	8085	1 621 852	0.995	1.000
B12x3.0	528 089	6	1055	30 487	0.998	1.000
C30x3.0	2 285 513	47	8730	833 420	0.996	1.000

Table 5. The error rates (%) of the simulated read sets before and after correction

ID	Original reads	Corrected	EULER-SR	SAET
B6x1.5	1.489	0.299	0.595	–
B6x3.0	2.960	0.882	1.984	–
B12x1.5	1.491	0.218	0.307	–
B12x3.0	2.960	0.661	1.094	–
B24x1.5	1.491	0.215	0.350	–
B24x3.0	2.960	0.643	1.000	–
C15x1.5	1.433	0.196	–	0.220
C15x3.0	2.864	0.707	–	0.663
C30x1.5	1.435	0.193	–	0.188
C30x3.0	2.870	0.691	–	0.527
B6x1.5	1.489	0.199	–	–
C15x1.5	1.433	0.193	–	–
B6x3.0	2.960	0.604	–	–
C15x3.0	2.864	0.695	–	–
B12x1.5	1.491	0.200	–	–
C30x1.5	1.433	0.195	–	–
B12x3.0	2.960	0.604	–	–
C30x3.0	2.870	0.697	–	–

as TN/(TN+FP). Table 4 shows the results of the classification test for the simulated read sets. For all datasets the sensitivity is >98 % meaning that only a few of the erroneous reads remain undetected. The specificity of the method is also >98 % except for the low-coverage base space read sets B6x1.5 and B6x3.0.

To measure the accuracy of correction, we computed the error rate of the read sets before and after correction by comparing them to the original reads before errors were introduced. We compared the accuracy of our correction method with EULER-SR (Chaisson *et al.*, 2004) in case of base space reads and to SAET by Applied Biosystems in case of color space reads. Table 5 shows that the error rate is significantly reduced by the correction. Performing hybrid correction with a mixed set of reads instead of correcting each set separately further increases the correctness of the reads for low-coverage read sets. Our method reduced the error rate much more than EULER-SR but SAET achieves in most cases better error rates than our method for color space reads.

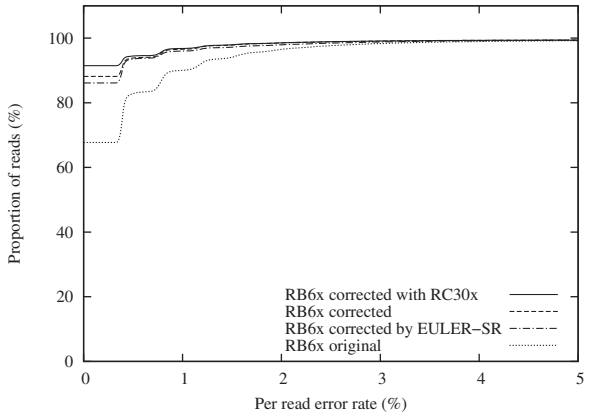


Fig. 7. The proportion of mapped reads as a function of the maximum allowed error rate when mapping the reads to the reference genome for the real 454 read set RB6x.

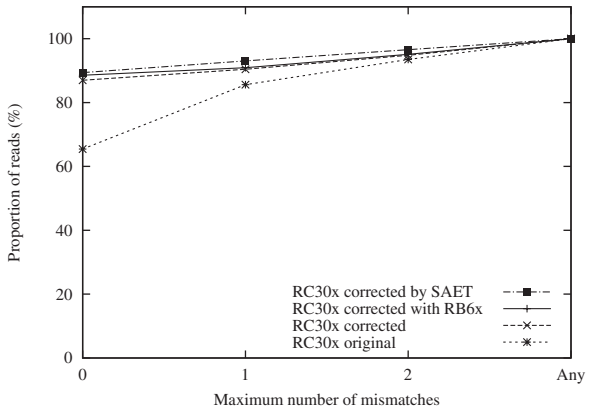


Fig. 8. The proportion of mapped reads as a function of the maximum number of mismatches allowed when mapping the reads to the reference genome for the real SOLiD read set RC30x. The coverage of RC30x is so much higher than the coverage of RB6x that the hybrid correction improves the correction result only slightly for RC30x.

For the real read sets, the accuracy of correction was measured by aligning the reads to the reference genome. We used BLAST (Altschul *et al.*, 1990) to align the base space reads and SOAP2 (Li *et al.*, 2009) to align the color space reads. Figures 7 and 8 show the proportion of reads that can be aligned to the reference genome as a function of allowed error rate. For example, 86% of the original color space reads can be aligned to the reference genome with at most one mismatch, whereas 91% of the corrected color space reads can be aligned with the same mismatch tolerance. Our new method performs better than EULER-SR but SAET achieves slightly better correction in color space than our method. The good performance of SAET might be due to its ability to correct more than one error in

Table 6. The impact of read correction on *de novo* assembly

ID	Number of contigs (≥100 bp)	N50	Mean contig length (bp)	Maximum contig length (bp)
RB6x	2587	2933	1728	13 555
RB6x corrected	1731	4318	2568	18 692
RC30x	10 868	438	348	2398
RC30x corrected	7027	590	455	3081

an *l*-tuple, while the bases surrounding an error must align exactly with other reads for our method to correct the error.

To demonstrate the impact of error correction on *de novo* assembly we ran the Velvet (Zerbino and Birney, 2008) assembler on the real read sets before and after correction. Table 6 shows that assembly is greatly improved by the error correction.

6 CONCLUSION

We have presented a tool for the hybrid correction of a mixed set of reads produced by several sequencing platforms including the SOLiD sequencing technology that produces color space reads. We showed that our method can detect errors with high sensitivity and specificity and also the error rate of the reads is reduced. We also showed that low-coverage read sets clearly benefit from hybrid correction with other read sets.

ACKNOWLEDGEMENTS

We wish to thank Andreas Bremges for implementing part of the algorithms and Simon J. Puglisi for helpful comments regarding a draft of this manuscript. We also wish to thank Rainer Lehtonen, Virpi Ahola, Ilkka Hanski, Panu Somervuo, Lars Paulin, Petri Auvinen, Esko Ukkonen, Veli Mäkinen and Niko Välimäki for insightful discussions about error correction.

Funding: Academy of Finland [grant number 118653 (ALGODAN)].

Conflict of Interest: none declared.

REFERENCES

Altschul,S.F. *et al.* (1990) Basic alignment search tool. *J. Mol. Biol.*, **215**, 403–410.

Applied Biosystems Incorporated (2008a) *Principles of di-base sequencing and the advantages of color space analysis in the SOLiD system*. Available at http://marketing.appliedbiosystems.com/images/Product_Microsites/Solid_Knowledge_MS/pdf/SOLiD_Dibase_Sequencing_and_Color_Space_Analysis.pdf (last accessed date April 15, 2010).

Applied Biosystems Incorporated (2008b) *A theoretical understanding of 2 base color codes and its application to annotation, error detection, and error correction*. Available at http://www3.appliedbiosystems.com/cms/groups/mcb_marketing/documents/generaldocuments/cms_058265.pdf (last accessed date April 15, 2010)

Batzoglou,S. *et al.* (2002) ARACHNE: a whole-genome shotgun assembler. *Genome Res.*, **12**, 177–189.

Butler,J. *et al.* (2008) ALLPATHS: de novo assembly of whole-genome shotgun microreads. *Genome Res.*, **18**, 810–820.

Chaisson,M. *et al.* (2004) Fragment assembly with short reads. *Bioinformatics*, **20**, 2067–2074.

Li,R. *et al.* (2009) SOAP2: an improved ultrafast tool for short read alignment. *Bioinformatics*, **25**, 1966–1967.

Li,R. *et al.* (2010) De novo assembly of human genomes with massively parallel short read sequencing. *Genome Res.*, **20**, 265–272.

Pevzner,P.A. *et al.* (2001) An Eulerian path approach to DNA fragment assembly. *Proc. Natl Acad. Sci. USA*, **98**, 9748–9753.

Sanger,F. *et al.* (1977) DNA sequencing with chain-terminating inhibitors. *Proc. Natl Acad. Sci. USA*, **74**, 5463–5467.

Schröder,J. *et al.* (2009) SHREC: a short-read error correction method. *Bioinformatics*, **25**, 2157–2163.

Shendure,J. and Ji,H. (2008) Next-generation DNA sequencing. *Nat. Biotechnol.*, **26**, 1135–1145.

Tammi,M.T. *et al.* (2003) Correcting errors in shotgun sequences. *Nucleic Acids Res.*, **31**, 4662–4672.

Zerbino,D.R. and Birney,E. (2008) Velvet: algorithms for de novo short read assembly using de Bruijn graphs. *Genome Res.*, **18**, 821–829.