American Journal of Bioinformatics Research 2013, 3(1): 1-9
DOI: 10.5923/j.bioinformatics.20130301.01

Review of Genome Sequence Short Read Error Correction Algorithms

M. Tahir[1,*], M. Sardaraz[1], Ataul Aziz Ikram[1], Hassan Bajwa[2]

[1]Department of Computing and Technology, Iqra University, Islamabad, Pakistan
[2]Department of Electrical Engineering, University of Bridgeport

Abstract Next-generation high throughput sequencing technologies have opened up a wide range of new genome research opportunities. High throughput sequencing technologies produces a massive amount of short reads data in a single run. The large dataset produced by short read sequencing technologies are highly error-prone as compared to traditional Sanger sequencing approaches. These errors are critical and removing them is challenging. Therefore, there are peremptory demands for statistical tools for bioinformatics to analyze such large amounts of data. In this paper, we present review of and measuring parameters associated with genome sequence short read errors correction tools and algorithms. We further present comprehensive detail of datasets and results obtained with defined parameters. The reviews present the current state of the art and future directions in the area as well.

Keywords Algorithms, Genome Sequence, Hadoop MapReduce, High Throughput, Next-Generation Sequencing, Short Reads Error Correction

1. Introduction

Biological sequences contain very important and useful information. The sequences are composed of A, T, G and C in a DNA structure. The compositions determine various traits such as personality, habits, and inheritance characteristics of species[1]. Next generation sequencing (NGS) technologies produce high throughput short read (HTSR) data at a lower cost[2]. High throughput platforms have open up a myriad of new and existing applications such as transcript-tome analysis, meta-genomics, single cell assembly and variation detection[3]. NGS technologies have become prominent tools in biological research[4, 5]. HTSR is changing the way genetic data are collected, stored and produced[4], and each aspect has its own challenges that need to be overcome.

As compared to traditional Sanger sequencing methods, next generation sequences have the advantages of deep and cheap coverage at the shorter read and high error rate[6, 7]. Shotgun fragment assembler is optimized for HTSR data such as, Altas, ARACHNE, Celera, and PCAP[8-11]. In de novo genome assembly, the goal is to build contiguous and unambiguous sequences known as contigs[12]. Due to different erroneous characteristics in assembling repeat regions, various contig extensions have been developed[2].

* Corresponding author:
tahir591@hotmail.com (M. Tahir)
Published online at http://journal.sapub.org/bioinformatics

Greedy short read algorithms and graph theory have been adopted and used in assembly programs such as SSAKE, SHARCGS, VCAKE and Taipan[13-16]. De Bruijin graphs approach has also actuated new quest in short read assembly[2, 17]. Various short read assemblers based on de Bruijin graphs have also been developed. Other visceral approaches for Sanger read such MisED[18-23] use alignment approach. Such alignments technologies are not feasible for millions of short reads alignment and does not adopt well at short reads[24, 25].

Short read error correction of different platforms is complex hence making error correction is difficult task[26]. A common approach for error correction of sequence reads is to determine a threshold and correct k-mers[3,17,18, 27]. Selection of correct thresholds is decisive task as low thresholds can result in too many uncorrected errors, while high thresholds will result in loss of correct k-mers[3]. Some researchers have developed and implemented SAP based read correction algorithms in assembly tools[19,28], other consider quality value of per-base and optimize the k-mer counting, but k-mer counting limits the performance of SAP-based error correction tools[6, 12, 29,30].

The NGS technologies generate short sequences that are prone to high error rates[31-32]. So, assemblies of long repeats and duplication suffer from short read length[33]. A Critical analysis on whole genome assembly algorithms, various types of errors and requirements for short read assemblies has been discussed in[34-40]. Sequencing technologies are bringing new possibilities for genome sequencing by generating read data with massive

throughputs. These reads are shorter and more prone to errors. This poses challenges for de novo fragment assembly algorithms such as accuracy and scalability[41-45].

2. Short Read Error Correction Tools and Algorithms

This section provides a review of short read error correction tools and algorithms. Due to unknown nature of the source genome, the reads from the same genomic location are inferred by relying on the assumption that they typically share sub-reads of fixed length, such as k-mers. Other techniques use multiple sequence alignments and some use suffix tree method. Therefore different researchers use different parameters for evaluation. This may lead to improper interpretation and cross comparison of all available methods. We classified them on the basis of methods and parameters that can be correctly understood.

2.1. K-Spectrum Based

In[3] authors present an algorithm *"Hammer"* developed for error correction in illumina/Solexa reads without uniformity assumptions. Hammer is based on hamming graph and a simple probabilistic model to correct sequencing errors. Hammer makes use of a sort technique to identify the set k of distinct k-mers in the reads as well as a table for sorting the multiplicity k-mers. It finds the connected components of hamming graph. Given $s^1,...s^{\tau+1}$ be a disjoint partition of $\{1,...,k\}$. These partitions represent spaced seeds into k-mers and also denote the restriction of k-mer ν to the seed s_j as $x(s_j)$. Then k-mers are denoted as $d(x,y)^{\tau}$ for x and y, there exists j such as $x(s_j)=y(s_j)$. Based on observations it creates spaced seeds "$\tau+1$" denoted as $s_j=\{j, j+^{\tau}+1, j+2(^{\tau}+1),...\}$. It also makes "$\tau+1$" copies of set "$k$" k-mers with the j^{th} copy that have the restriction of k to the indices s_j, and sorts each copy lexicographically. In the next step it linearly scans each copy and identifies equivalence blocks. This is then used to compute hamming distance for each block corresponding to a full length k-mer. In case of k-mer pair that is within the hamming distance τ shows that their components are joined[50]. The algorithm finds connected components *"cluster"* that does not have size one and the consensus string is unique. It corrects every k-mer in the cluster to the consensus string. Singleton k-mer is trivially the consensus. It must be either the only element in the generating set of k-mer or simply contain more than τ errors. Multiple consensus string exists in small clusters of size two or three, which creates ambiguity. In each case hamming graph does not provide good information regarding the correctness of the k-mers. Therefore, it makes decision solely based on the multiplicity of each k-mer x. If it is greater than the threshold value it keeps x in the dataset, otherwise it removes x from the dataset. Computation of the consensus of the cluster performs in $O(n_k^*)$ time.

In[6] authors present a novel approach *"Reptile"* to cope with error correction problem of short read sequence in illumine/Solexa reads. Reptile performs the operation with k-mers spectrum and corrects errors using hamming distance. It also makes use of neighboring k-mers for correction possibilities for both potential as well as contextual information. This algorithm has two phases (a) Information extraction: in which k-spectrum R^k of R is extracted for given threshold d-extract hamming graph $G_H=(V_H,E_H)$ where $V_i \in V_h$ represents α_i R^k and $\exists e_{ij}=(V_i,V_j) \in E_h \Leftrightarrow h_d(V_i,V_j) \leq d$. This is used to compute tile occurrences. (b) Individual read correction: It points to a tile t at the beginning of read, identifying d-mutant tiles of t. The next step is to correct errors in t, wherever applicable. This is done by adjusting tile t and identifying d-mutant step repeatedly until tile placement choices are exhausted. This amounts to a total run time of $O(nLlog(nL))$ and space usage of $O(|R^k|+|R^{|t|}|)$. Reptile is faster, has low memory requirements, and provides high accuracy. However it is single threaded, and lacks the possibility to determine unambiguously all errors. It also does not show explicitly model repeats.

In[12] authors present an algorithm "*Quake*" for detection and correction of errors in Illumine/Solexa short read sequences. Quake is based on maximum likelihood paradigm which incorporates quality values and nucleotide specific miscall rates. Quake makes use of k-mers coverage for differentiation of trusted k-mers in genome sequence, and searches error correction. Let observed nucleotide be denoted by $O=O_1,O_2,...,O_n$, and actual nucleotide of sequence fragment be denoted by $A=A_1,A_2,...,A_n$. Conditional probability is applied using bayes' rule to the observed nucleotides as given in equation 1[12].

$$P(A=a|O=o) = \prod_{i=1 \text{ to } N} P(O_i=o_i|A=a_i)P(A_i=a_i)/P(O_i=o_i) \quad (1)$$

The probability of nucleotide to be at position i denoted by $P_i = 1-10^{-qi/10}$, q_i denotes quality value, and $Eq(x,y)$ is the probability that the base call y of nucleotide x at quality value q given that sequence error. Then $P(O_i=o_i|A_i=a_i)$ can be written as $P(O_i=o_i|A_i=a_i)=P_i$ if $o_i=a_i$ otherwise $(1-p_i)E_{qi}(a_i,o_i)$. If all resulting k-mers are trusted then the set of corrections are valid. However Quake shows high accuracy on realistically simulated reads. It however cannot find any cutoff on single cell data.

In[29] authors present the *"CUDA"* programming model. It is used to correct sequencing errors in high throughput short read data for Roche/454 platform. It is based on spectral alignment method. CUDA kernel is used to denote the sequential processing for correction of an individual read r_i. Kernel is invoked for each read r_i of R. Time depends on $\Delta(\Delta \geq 1)$ for the correction within a weak l-tuple. CUDA delta mutation consists of two phases, i.e delta (Δ) mutation voting and identifying all l-tuples. On successful completion of the first phase corresponding counters in voting matrix are incremented to point positions of mutation and errors are fixed, based on the highest scores in matrix. CUDA provides multithreading, parallelism, space efficiency and accuracy, but it does not support distributed memory operations.

Research in[18, 47] presents a short read error correction algorithm "*Eular SR*" for applied Biosciences and

Illimina/Solexa reads based on spectral alignment method. This method first establishes a spectrum of trusted k-mers from input data set and then correct each read in the sequence that it contains based on spectrum only. However this algorithm is faster and provides fixation of memory leak, but its accuracy drops significantly while choosing optimal parameters, it also require large memory.

Another related research presented in[46] depicts a computation method to detect and correct errors in the genomic sequence repeats for the Pacific Biosciences platform. Computation method is available in the form of software package REDEEM (Read error detection and correction via expectation maximization). To correct errors in read *r,* it considers each of the nucleotide to have appeared at least once upto k-mers. Let's suppose that the nucleotide at position $1 \leq i \leq L$ of the read appear at position $1 \leq t \leq k$ of k-mer x_l. The probability that the true nucleotide at position *t* was *b* prior to possible misread as given in equation 2[46].

$$\rho_{it}(b)=\frac{\sum \alpha_m \rho_e(x_m,x_l)}{\sum \alpha_m \rho_e(x_m,x_l)} \quad (2)$$

Here it estimates T_m as substitute for the unknown α_m. At position *i* multiple overlapping k-mers provide dependent information. Therefore, it takes average across available *t* to achieve distribution *i.e*, $p_i(b)$. It then declares nucleotide *r[i:i]* as misread and correct it to $argmax_b p_i(b)$, given $argmax_b p_i(b) \neq r[i:i]$. This is the best approach for repeat rich genomes, but suffers from large time complexity problem due to modeling repeats.

2.2. Suffix Tree Based

Authors of[48] presented short read error correction (SHREC) algorithm for short read error correction in Illuminia/Solexa based on generalized suffix trie method. It constructs trie from all reads and their complements. Trie is then traversed to point out and analyze the potential errors in associated reads. Depth first traversal of $ST(\Re)$ is performed for inspection of nodes from level *s* to level $t+q$. The algorithm identifies all nodes with at least two children in which one of the children *w* has smaller weight than the threshold weight. Subsequently it finds a set of reads $R(w)$. So correction is performed to a sibling of *w* that fits the suffix and examines each read R_i that belongs to $R(w)$. SHREC points error position in R_i and performs the correction of the associated sibling's nucleotide edge. It is robust, accurate and has the ability to achieve sensitivity and specificity. It has the drawbacks of large memory usage, does not explicitly model repeats, and difficult choice of optimal parameters.

Research in[49] depicts a modified version of SHREC algorithm aiming to correct SOLEXA/Illumina reads. The SHREC's statistical model is modified to put up reads of variable lengths. Given a set of *s* reads contains *r* reads of various lengths and are denoted by $l_1,l_2,\ldots,l_r$. It is assumed that the number of reads of length l_i is k_i and let *m* be large enough that each sequence having length *m* appears once. Weight W_m of a node at level *m* is the number of suffixes whose path in the trie passes through that node. Level *m's* node weight is denoted by $W_m=\sum_{i=1 \text{ to } r} W_{(m,i)}$, where $W_{(m,i)}$ represent the contribution of reads length l_i to the weight of node. If $l_i<m, W_{(m,i)}=0$, otherwise each read is a Bernoulli trail to get the path label. 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 success probability of Bernoulli trail is a_i/n. K_i is the number of trails and $W_{(m,i)}$ is distributed according to the binomial distribution $Bin(K_i,a_i/n)$. Therefore, expected value of $W_{(m,i)}$ is $E(W_{(m,i)}=k_i a_i/n$ for $l_i>=m$, and *0* otherwise. Variance is as given in equation 3[49].

$$\sigma^2(W_{(m,i)}=k_i(a_i/n-a_i^2/n^2) \text{ for } l_i>m \quad (3)$$

The linearity of expectation is applied to get $E(W_m)=E(\sum_{i=0 \text{ to } r} W_{(m,i)})=\sum_{i=0 \text{ to } r} E(W_{(m,i)})$. For $W_{(m,i)}$ with different *I,* that are independent variables, the variance is given by equation 4[49].

$$\sigma^2(W_m)=\sigma^2(\sum_{i=0 \text{ to } r} W_{(m,i)})=\sum_{i=0 \text{ to } r}\sigma^2(W_{(m,i)}) \quad (4)$$

The nodes with weight less than the value of equation 5[49] are corrected.

$$E(W_m)-\alpha.\sigma(W_m) \quad (5)$$

Here α is the parameter of strictness. These follow by correction of insertion, deletion and indeterminate bases. This hybrid version significantly reduces errors in mixed reads. It also finds out error rate of reads before and after correction. But its performance is not as good compared to substitution error-based error- correction methods.

Research work presented in[52] described *"HiTEC"* for Illumina Genome Analyzer, ABI SOLiD and Roche/454 platform based on suffix array constructed from reads and their complements. It uses three concepts such as *witness, support* and *cluster*. HiTEC correction processes consist of two steps *(i)* A comprehensive statistical analysis to determine thresholds *(ii)* Usage of suffix array to compute the length of every witness and its cluster for error correction. HiTEC also make use of libdivsufsort library of Yuta Mori[53]. It performs computation of LCP to get all witnesses that have larger length and cluster than thresholds. The computational time complexity is $O(nlogn)$. It provides speed computation, efficiency and accuracy. But it is a serial program, and also shows degradation in performance while constructing suffix arrays for large dataset.

Another research conducted in[30] presents an algorithm *"PSEAC"* for Illumina/ Solexa and ABI SOLiD platform based on partial suffix arrays for short read error correction. PSAEC is a scalable parallel algorithm that works on multi-core machines using Pthreads. The processes of presented algorithm consist of five phases i.e. pre-processing, classification, multi-key quick sorting, error correction and iteration. Pre-processing reads data set and complements of their reverse, it then store them into memory, and compute thresholds. Next, it pushes them into classification for optimization and implements ECSSORT function of multi-key quick sort. Following this task it identifies errors and records its positions for further processing. Perform correction of the erroneous base values and records into an array *L*. Following error correction, it performs iteration of

second to forth phase to achieve high accuracy. It provides scalability, parallelism and improved accuracy. However it does not support distributed memory parallel computation and thus collision of threads degrades performance.

Similar research in[2] presents *"DecGPU"* a parallel and distributed error correction algorithm for Roche/454, Illumina/Solexa, Pacific Biosciences and Helicos Biosciences reads. DecGPU is a hybrid combination of CUDA and the MPI programming model. DecGPU is based on SAP (Spectral Alignment Problem) approach and makes use of counting bloom filter[53]. Error correction algorithm consist of construction of the distributed k-mer spectrum, filtration of error free reads, fixation of erroneous reads, trimming the fixed reads and iteration to correct more than one base error reads. Each processing element denoted as P_i consists of hybrid CPU and GPU threads, and provides performance maximization via overlapping. It provides parallel and distributed error correction, high accuracy, feasibility and flexibility in terms of memory. It lacks runtime scalability of available computation resources.

Research contribution of[55] also presents a scalable tool *"SEAL"* for Illumina/Solexa short read pair mapping and duplicate removal. It is distributed alignment tool that combines Burrows-Wheeler Aligner (BWA)[56] and duplicate read detection and removal. To efficiently distribute input, output and computation across cluster nodes, SEAL harnesses the Hadoop Map-Reduce framework which gives guarantee of reliability by resisting node failure and transient events. SEAL specializes in the pair-end alignment of sequences read by Illumina sequencing machines. SEAL is currently structured into pairReadQseq and seqal applications. These applications implement a Map-Reduce algorithm[57] that runs on Hadoop framework. SEAL is able to achieve scalability rates and parallelization in alignments. SEAL suffer from optimal parameter selection problem and also has large space requirement.

2.3. Multiple Sequence Alignment Based

A similar work based on multiple sequence alignment is given in[26], called "*Coral*". Coral is adjustable to reads of Illumina/Solexa Genome Analyzer and Roche/454. Coral takes reads that share common k-mers and forms multiple alignment correction based on alignment and corresponding consensus sequences. The operation begins with indexing that occurs in the reads or in its components. Coral extracts all reads that form k-mers index sharing at least one k-mer with base read. Reads in k-mer neighbourhood are aligned and corrected several times. Indexing k-mer causes the reads to be aligned and also locates the probable position for sequence alignment. It tries to align the read without error in consensus. Therefore total computation is skipped to save time. Gap positions are ignored in fraction computation with quality value that is higher than *1-e*. For each alignment read the algorithm checks and compares it with the consensus sequence in each aligned position. If not agreed upon comparison, correction of reads at each position is done to agree with consensus sequence. The consensus value is t_s, where ($0.5 \leq ts < 1.0$). It performs the correction of base only if the quality threshold t_Q ($0.5 \ t_Q<1.0$) exceeds from value Lm_{max} to eLm_{max}. Time complexity of quality threshold t_Q is $O(L)$ and worst case time complexity is $O(Lm_{max})$ to detect potential errors. $O(eL^2m_{max})$ time is taken to correct them. Coral has less memory requirements compared to SHREC, is faster for short reads and is multithreaded. Coral also has some disadvantages i.e. large memory requirement than Reptile and Quake, and is not compatible with color space reads.

In[51] authors present a mapping algorithm*"SHRiMP2"* based on original short read mapping techniques. SHRiMP2 points to the mapping sensitivity. It also has the ability to achieve high rate of accuracy at significant speed with the use of caching heuristic over previous versions. SHRiMP2 supports input format of fasta and fastq, output format of SAM and mapping of Illumina/solexa, Roche/454, AB /Solid reads. Additionally it has paired mapping mode and parallel computation capability. It supports both letter space and color space reads, provides multithreaded and parallel computation and has better sensitivity. It is however slower and has overhead in scanning the whole genome.

In[54] authors present a novel algorithm *"ECHO"* for correcting base-call errors in the short reads without genome reference for Illumina/Solexa platform. ECHO is based on a probabilistic model and has the ability to assign quality score to each corrected base. There is no need to specify parameters or unknown values for optimization. ECHO sets the parameters automatically and estimates error characteristics specific in each sequence run. ECHO has the ability to improve the accuracy of previous error correction methods modified for specific sequence coverage depth and position in the read. ECHO performs error correction as a pre-processing step and considerably facilitates de novo assembly. ECHO presents improvement towards the end of the reads where previous methods are less effective. But it is slower and has high memory requirements.

3. Results and Discussion

Table 1. Data Set used for performance evaluation

Data Set	Archive Accession Number	Genome	Read Length	Coverage	Error Rate	Number of Reads
D1	SRX000429	E.Coli	36bp	160x	0.6%	20.8M
D2	SRR001665_1	E.Coli	36bp	80x	0.6%	10.4M
D3	SRR006332	A.sp	36bp	173x	1.5%	17.7M
D4	NC_005966	A.sp	36bp	40x	1.5%	4.0M

Table 2. Experimental Results

Data Set	Method	TP	FN	FP	TN	Sensitivity	Specificity	Gain
D1	SHREC	138000	36900	6700	1050000	0.7890223	0.99365951	0.75071469
	H-SHREC	1617685	3228	13998	2231089	0.99800853	0.99376505	0.98937266
	HiTEC	140215	3575	3200	7408408	0.97513735	0.99956824	0.95288268
	PSEAC	208485	7425	298	7413763	0.96561067	0.99995981	0.96423047
	DecGPU	1620660	25300	349908	1895179	0.98462903	0.84414502	0.77204306
	SEAL	241740	10038	1	6101444	0.96013154	0.99999984	0.96012757
	Coral	169000	13700	12400	1110000	0.92501368	0.98895225	0.85714286
	SHRiMP2	418351	6238	91205	1417738	0.98530815	0.93955703	0.77050041
	ECHO	488165	12801	129881	3238521	0.97444737	0.96144136	0.71518626
	Hammer	806972	15783	29473	8749471	0.98081689	0.99664276	0.94499456
	Reptile	942457	12585	987457	3709818	0.98682257	0.78978088	-0.04711835
	QUAKE	0	0	0	0	0	0	0
	CUDA	0	0	0	0	0	0	0
	Eular-SR	2311124	61545	1298891	6785320	0.97406086	0.8393299	0.42662209
	Redeem	3443262	130905	133558	5643901	0.96337468	0.97688292	0.92600709
D2	SHREC	1620660	349908	253	1895179	0.82243292	0.99986652	0.82230453
	H-SHREC	1617685	13998	3228	2231089	0.99142113	0.99855526	0.9894428
	HiTEC	2575411	660533	306	629750	0.79587626	0.99951433	0.79578169
	PSEAC	2571520	31367	4197	1258916	0.98794915	0.99667726	0.98633671
	DecGPU	4053688	23	1024	5611265	0.99999433	0.99981754	0.99974172
	SEAL	4053827	4990124	885	621164	0.44823629	0.99857728	0.44813843
	Coral	6435328	3481	1621	3225570	0.99945937	0.99949771	0.99920762
	SHRiMP2	6436305	3129803	644	99248	0.67282379	0.99355304	0.67275646
	ECHO	6406078	2	5395	3254525	0.99999969	0.99834505	0.99915752
	Hammer	6411346	3185858	127	68669	0.66804311	0.99815396	0.66802988
	Reptile	8578176	1	8651	1079172	0.99999988	0.99204742	0.99899139
	QUAKE	0	0	0	0	0	0	0
	CUDA	0	0	0	0	0	0	0
	Eular-SR	324310	61545	1298891	6785320	0.84049708	0.8393299	-2.52577004
	Redeem	213213	130905	133558	5643901	0.6195927	0.97688292	0.23147583
D3	SHREC	8586743	1056392	84	22781	0.89045139	0.99632626	0.89044268
	H-SHREC	0	0	0	0	0	0	0
	HiTEC	262944	541584	1748	13507	0.32683014	0.88541462	0.32465744
	PSEAC	429389	611875	2333	126868	0.41237285	0.98194286	0.4101323
	DecGPU	527718	432283	1426	30010	0.54970568	0.95463799	0.54822026
	SEAL	859386	59343	3823	255939	0.9354075	0.98528268	0.93124632
	Coral	1056625	234132	2109	60386	0.81860877	0.9662533	0.81697484
	SHRiMP2	747968	25523	4113	806307	0.96700285	0.99492485	0.9616854
	ECHO	1141544	456012	4360	412008	0.71455649	0.98952849	0.71182732
	Hammer	1497762	14538	8251	1621860	0.99038683	0.99493838	0.9849309
	Reptile	2285417	32147	8826	833120	0.98612897	0.98951714	0.98232066
	QUAKE	214590	15647	1167	64010	0.93203959	0.98209491	0.9269709
	CUDA	747911	32191	4170	811739	0.95873488	0.99488914	0.95338943
	Eular-SR	264083	65412	609	15079	0.80147802	0.96118052	0.79962974
	Redeem	1141382	768120	4522	417900	0.59773805	0.98929507	0.59536989
D4	SHREC	339700	23	701	813320	0.9999323	0.99913884	0.99786885
	H-SHREC	578295	32	2380	573056	0.99994467	0.99586401	0.99582935
	HiTEC	746399	23	4330	403052	0.99996919	0.98937116	0.99416818
	PSEAC	0	0	0	0	0	0	0
	DecGPU	1093198	56	4552	1084129	0.99994878	0.99581879	0.99578506
	SEAL	1410950	57	8489	762526	0.9999596	0.98898984	0.99394333
	Coral	1128525	48	2376	2698039	0.99995747	0.99912014	0.99785215
	SHRiMP2	1919892	47	8068	1901030	0.99997552	0.99577392	0.9957733
	ECHO	2476109	73	15128	1337898	0.99997052	0.98881914	0.99386111
	Hammer	2733995	170	5972	6539234	0.99993782	0.99908758	0.99775361
	Reptile	4651239	321	19852	4608147	0.99993099	0.99571046	0.99566318
	QUAKE	5997457	300	37388	3244725	0.99994998	0.98860856	0.99371632
	CUDA	944906	106	3821	2480352	0.99988783	0.99846186	0.9958445
	Eular-SR	2820956	0	162698	5223249	1	0.96979213	0.94232523
	Redeem	0	0	0	0	0	0	0

Table 3. Summary of comparative analysis of short reads error correction algorithms

Algorithm	Method	Strengths	Weaknesses
K-Spectrum Based			
Hammer[3]	Hamming graph and Probabilistic model	Finding any cutoff with use of non-uniformity, support to work in multicell data.	Initial sorting of k-mers assumptions of at most are correct k-mers in cluster
Reptile[6]	Hamming Distance and neighboring k-mers	Faster, low memory requirements, high accuracy, consistence for ambiguous base, sensitivity and gain values are slightly slower	Single threaded, impossibility of determining unambiguously all errors, do not explicitly model repeats
QAUKE[12]	Maximum Likelihood	High accuracy on realistically simulated reads	Cannot find any cutoff on single cell data
CUDA Delta-Mutation Algorithm[29]	Spectral alignment and CUDA program	Multithreaded, parallelization, space efficiency, high accuracy.	Does not support distributed memory operations
Eular SR[18, 47]	de Bruijn graph	Faster, fixation of memory leak,	Accuracy drops significantly choosing optimal parameters, more memory usage
Redeem[46]	Maximum Likelihood	Best for repeat rich genomes	Tales longer time due to complexity of modeling repeats.
Suffix Tree Based			
SHERC[48]	Generalized Suffix Trie	Robust, accurate, ability to achieve high identification sensitivity and specificity, multithreading	Parameters sensitivity, large memory usage, choosing optimal parameter, do not explicitly model repeats
Hybrid Shrec[49]	Based on SHREC Program	Significantly reduces errors in mixed reads. Ability to detect errors in reads and error rate of reads before and after correction	Performance is not as par with substitution error-based error-correction methods
HiTEC[52]	Suffix Array	Faster, Efficient, and Accurate	Serial program, degradation in performance while constructing suffix array for large data, intolerable time and space for large data.
PSAEC[30]	Partial Suffix Array	Multithreaded, scalable parallel, improved accuracy.	Does not support distributed memory parallel computation, collisions of threads degrade speed up
DecGPU[2]	SAP,CUDA,MPI	Parallel and distributed error correction, high accuracy, feasible and flexible regarding memory for large data sets	Does not show improved runtime scalability as per computation resources
SEAL[55]	Hybrid of BWA and duplicate read determination and removal	Support for distributed computation, high reliability and accuracy, faster, flexible, capability of utilizing resources efficiently, capability to parallelize steps in alignment	Large space complexity
Multiple Sequence Alignment Based			
Coral[26]	Multiple Alignment	Significantly less memory than shrec, faster for short reads, multithreaded	More memory than Reptile and Quake, distinguishing of run times is not easy, correlates quadratic ally in worst case for large reads, not compatable with color space reads.
SHRiMP2[51]	Hash-based	Support both letter space and color space reads, multithreaded, parallel computation, better sensitivity for various polymorphism classes, non-uniformity	A bit slower, overhead in scanning the whole genome.
ECHO[54]	Probabilistic model, Bayesian Framework	Quality score assignment, explicitely models hetrozygosity in diploid genomes, high accuracy, capable to cope with non uniform coverage, automatically chooses parameters, improve data quality	Slower, high memory requirements

Algorithms discussed in above section have been simulated and evaluated by sole authors in individual papers. We have selected benchmark datasets as shown in (Table 1) from all the experiments to comparatively evaluate and analyze the algorithms. The results of the corresponding dataset *D1* (Accession Number: SRX000429), *D2* (Accession Number: SRR001665_1), *D3* (Accession Number: SRR006332) and *D4* (Accession Number: NC_005966) are shown in (Table 2). The two algorithms Quake and CUDA were generated errors while simulating on datasets *D1* and *D2* due to cutoff value and not supporting proper distribution respectively. Simulation of Hybrid-SHREC on dataset *D3* produced garbage values due to large space complexity. Also PSEAC and Redeem cannot be properly run on *D4* due to collision of threads and modeling repeats. That is why these were excluded in comparative analysis. The remaining algorithms are analyzed as follows.

Analysis parameters are based on Time and space complexity and are, True Positive (TP): is any erroneous base that is changed to the true base, a False Positive (FP): is any true base changed wrongly, a True Negative (TN): is any true base left unchanged, and a False Negative (FN): is any erroneous base left unchanged. Sensitivity =TP/ (TP+FN) and Specificity =TN/ (TN+FP). Gain= (TP-FP)/ (TP+ FN) represents the total percentage of erroneous bases removed from the dataset post-correction. Clearly, best methods generate gain that approach to 1 but in some cases its value may be negative that actually introduces more errors than they correct[2, 6, 44, 46, 48].

Results analysis of SHREC[48] shows that its accuracy with low coverage is at least 80% and the accuracy rate of Eular-SR[18, 47] reduces significantly with low coverage. The analysis of Coral[26] shows that it has relatively high error rate. That provides whole read alignment with slight edge over k-mers based method. On the other hand due to significant lower TP rates Reptile[6] fall short perceptible of the performance of Coral. Analysis of SEAL[55] describes the capability of throughput levels comparable to single node operation, which minimizes the distribution overhead due to implementation on Hadoop framework. Despite increase in size of the cluster throughput consistency of each node remains the same. SEAL also has the ability of utilizing the available resources efficiently in presence of massive data thus achieves scalability. HiTEC[52] and ECHO[54] yield higher gain value comparable to other methods and also have automated parameter selection for performance optimization. For dataset *D1* the SHRiMP2[51] produces negative gain values, which increase errors instead of correcting them. The results of Hammer[3] are also comparable but it neither detects nor corrects errors in clusters without uniformity. The DecGPU[2] analysis shows that it is a superior method in terms of error correction and execution speed. In summary HiTEC, ECHO and DecGPU are comparatively more accurate and efficient. A theoretical comparison in terms of strengths and weaknesses of reviewed algorithms are also presented in (Table 3). Further investigation is necessary to develop specific tools and algorithms to achieve high throughput.

4. Conclusions and Future Directions

The aim of the reviewed algorithms is to compare error rate in next generation sequencing technologies. The growth of NGS technologies present significant bioinformatics challenges, specifically design of bioinformatics tools that handle the operation of massive amounts of data efficiently. In this paper, we have described genome sequence short read error correction algorithms in detail. We concluded that most of the illumine/ Solexa sequencer algorithms focused on substitution errors, while the algorithms build for Roche/454 sequencers have slightly focused on error correction due to longer reads and low error rates. The improvement in sequence platforms and introduction of high throughput sequencing technologies such as Ion Torrent has changed the traditional way of reads and of insertion and deletion errors. Therefore the current methods are unable to achieve good results and hence need improvement.

To achieve the potential of NGS, it is important to maximally construe and utilize these short reads. For successful NGS technology application it is essential to develop large amounts of data storage, management and build informatics tools that can efficiently analyze data. It is also required to develop high performance computing and intensive bioinformatics applications to achieve the benefits of NGS technologies. The most significant footprint of NGS is successful aligning and assembling short reads to the reference genome. Efficiently aligning short reads to reference genome is a challenging task, particularly in development of new algorithms to manage ambiguity and alignment errors. It is important to develop such algorithms that support distributed-memory parallel computation to speed up the processes. It is also essential to develop algorithms that simultaneously estimate error parameters from data in regard to fast computation and handle large datasets via better memory management. It is also essential for error correction algorithms to distinguish errors from polymorphism.

REFERENCES

[1] Mathkour H, Ahmad M. Genome sequence analysis: A Survey. J Computer Science 2009; 5(9): 651-660.

[2] Liu Y, Schmidt B, Maskell D L. DecGPU: distributed error correction on massively parallel graphics processing units using CUDA and MPI. BMC Bioinformatics 2011; 12:85

[3] Medvedev P, Scott E, Kakaradov B, *et al*. Error correction of high-throughput sequencing datasets with non-uniform coverage. Bioinformatics 2011; 27: 137-141.

[4] Shendure J, Ji H. Next-generation DNA sequencing. Nat Biotechnol 2008; 26: 1135-1145.

[5] Hawkins R, Hon G, Ren B. Next-generation genomics: an integrative approach. Nat Rev Genet 2010, 11: 476-486.

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

[7] Sanger F, Nicklen S, Coulson A R. DNA sequencing with chain-terminating inhibitors. Proc Natl Acad Sci 1977; 74: 5463–5467.

[8] Havlak P, Chen R, Durbin KJ, *et al.* The Atlas genome assembly system. Genome Res 2004; 14: 721-732.

[9] Batzoglou S, Jaffe DB, Stanley K, *et al.* ARACHNE: a whole-genome shotgun assembler. Genome Res 2002; 12: 177-189.

[10] Myers E W, Sutton G G, Delcher A L, *et al.* A whole-genome assembly of Drosophila. Science 2000; 287: 2196-2204.

[11] Huang X, Wang J, Aluru S, *et al.* PCAP: a whole-genome assembly program. Genome Res 2003; 13: 2164-2170.

[12] Kelley D, Schatz M, Salzberg S. Quake: quality-aware detection and correction of sequencing errors. Genome Biology 2010; 11: R116.

[13] Warren RL, Sutton GG, Jones SJ, *et al.* Assembling millions of short DNA sequences using SSAKE. Bioinformatics 2007; 23: 500-501.

[14] Dohm JC, Lottaz C, Borodina T, *et al.* SHARCGS: a fast and highly accurate short-read assembly algorithm for de novo genomic sequencing. Genome Res 2007; 17: 1697-1706.

[15] Jeck WR, Reinhardt JA, Baltrus DA, *et al.* Extending assembly of short DNA sequences to handle error. Bioinformatics 2007; 23: 2942-2944.

[16] Schmidt B, Sinha R, Beresford-Smith B, *et al.* A fast hybrid short read fragment assembly algorithm. Bioinformatics 2009; 25: 2279-2280.

[17] Pevzner P.A,Tang H,Waterman M S. An Eulerian path approach to DNA fragment assembly. Proc Natl Acad Sci 2001; 98: 9748–9753.

[18] Chaisson MJ, Pevzner PA. Short read fragment assembly of bacterial genomes. Genome Res 2008; 18: 324-330.

[19] Zerbino DR, Birney E. Velvet: algorithms for de novo short read assembly using de Bruijn graphs. Genome Res 2008; 18:821-829.

[20] Butler J, MacCallum I, Kleber M, *et al.* ALLPATHS: de novo assembly of whole-genome shotgun microreads. Genome Res 2008; 18: 810-820.

[21] Simpson JT, Wong K, Jackman SD, *et al.* ABySS: a parallel assembler for short read sequence data. Genome Res 2009; 19: 1117-1123.

[22] Li H, Homer N. A survey of sequence alignment algorithms for next-generation sequencing. Brief Bioinformatics 2010; 11: 473–483.

[23] Tammi M.T, Arner E, Kindlund E, *et al.* Correcting errors in shotgun sequences. Nucleic Acids Res 2003; 31: 4663-4672.

[24] David A, Wheeler, Srinivasan M, *et al.* The complete genome of an individual by massively parallel DNA sequencing. Nature 2008; 452: 872–876.

[25] Li H, Durbin R. Fast and accurate long-read alignment with Burrows-Wheeler transforms. Bioinformatics 2010; 5: 589-595.

[26] Salmela L, Schroder J. Correcting errord in short reads by multiple alignments. Genome analysis 2011; 27:1455-1461.

[27] Zhao X, Palmer LE, Bolanos R, *et al.* Edar: an efficient error detection and removal algorithm for next generation sequencing data. J Comput Biol 2010; 17: 1549–1560.

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

[29] Shi H, Schmidt B, Liu W, *et al.* A parallel algorithm for error correction in high-throughput short-read data on CUDA-enabled graphics hardware. J Computing Biology 2009; 17: 603–615.

[30] Zhao Z, Yin J, Zhan Y, *et al.* PSAEC: An improved algorithm for short read error correction using partial suffix arrays. LNCS 2011; 6681: 220-232.

[31] Li R, Zhu H, Ruan J, *et al.*. De novo assembly of human genomes with massively parallel short read sequencing. Genome Res 2010; 20: 265-272.

[32] Bentley D R, Balasubramanian S, Swerdlow H P, *et al.* Accurate whole human genome sequencing using reversible terminator chemistry. Nature 2008; 456:53–59.

[33] Alkan C, Sajjadian S b, Eichler E E. Limitation of next-generation genome sequence assembly. Nature 2011; 8: 61-65.

[34] Green P. Whole-genome disassembly. Proc Natl Acad Sci 2002; 99: 4143–4144.

[35] Schatz M C, Delcher A L, Salzberg S L. Assembly of large genomes using second-generation sequencing. Genome Res 2010; 20: 1165–1173.

[36] Meader S, Hillier L W, Locke D, *et al.* Genome assembly quality: assessment and improvement using the neutral indel model. Genome Res 2010; 20: 675–684.

[37] Li H, Ruan J, Durbin R. Mapping short DNA sequencing reads and calling variants using mapping quality scores. Genome Res 2008; 11: 1851-1858.

[38] Langmead B, Trapnell C, Pop M, *et al.* Ultrafast and memory-efficient alignment of short DNA sequences to the human genome. Genome Biol 2009; 3:R25.

[39] Ning Z, Cox A J, Mullikin J C. SSAHA: a fast search method for large DNA databases. Genome Res 2001; 11: 1725–1729.

[40] Li R, Yu C, Li Y, *et al.* SOAP2: an improved ultrafast tool for short read alignment. Bioinformatics 2009; 15: 1966-1967.

[41] Huang W, Marth G. EagleView: a genome assembly viewer for next-generation sequencing technologies. Genome Res 2008; 9: 1538-1543.

[42] Bao H, Guo H, Wang J. MapView: visualization of short reads alignment on a desktop computer. Bioinformatics 2009; 12: 1554-1555.

[43] Milne I, Bayer M, Cardle L, *et al.* Tablet next generation

sequence assembly visualization. Bioinformatics 2010; 3: 401-402.

[44] IGV Software Home Page. http://www.broadinstitute.org/igv (last accessed on 22 August 2011).

[45] Li H, Handsaker B, Wysoker A, et al. The Sequence Alignment/Map format and SAMtools. Bioinformatics 2009; 16: 2078-2079.

[46] Yang X, Aluru A, Dorman K S. Repeat-aware modeling and correction of short read errors. BMC Bioinformatics 2011; 12:S52.

[47] Chaisson M J, Pevzner P, Tang H. Fragment assembly with short reads. Bioinformatics 2004; 20: 2067–2074.

[48] Schroder J, Schroder H, Puglisi S J, *et al.* SHREC: a short-read error correction method. Bioinformatics 2009; 25: 2157–2163.

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

[50] Cormen T H, Charles E L, Rivest R L, *et al.* Introduction to Algorithms 2nd Edition. McGrawHill Book Company 2001; pp. 505–509.

[51] David M, Dzamba M, Lister D, *et al.* SHRiMP2: Sensitivity yet practical short read mapping. Bioinformatics 2011; 27: 1011-1012

[52] Ilie L, Fazayeli F, Ilie S. HiTEC: accurate error correction in high-throughput sequencing data. Bioinformatics 2011; 27: 295–302.

[53] Mori Y.: Short description of improved two-stage suffix sorting algorithm, http://homepage3.nifty.com/wpage/software/itssort.txt, (last accessed on 23 August 2011)

[54] Kao W C, Andrew H, Chan, Yun S S. ECHO: A reference-free short-read error correction algorithm. Genome Res 2011; 110:1181-1192.

[55] Pireddu, Leo S, Zanetti G. SEAL: a distributed short read mapping and duplicate removal tool. Sequence Analysis 2011; 27: 2159-2160.

[56] Li H, Durbin R. Fast and accurate short read alignment with Burrows-Wheeler transform. Bioinformatics 2009; 25: 1754–1760.

[57] Dean,J. and Ghemawat,S. MapReduce: simplified data processing on large clusters. In OSDI '04: 6th Symposium on Operating Systems Design and Impl., USENIX Association 2004.