MRSFAST-ULTRA: A COMPACT, SNP-AWARE MAPPER FOR HIGH PERFORMANCE SEQUENCING APPLICATIONS

by

Iman Sarrafi
B.Sc., Shahid Beheshti University, 2010

a Project submitted in partial fulfillment
of the requirements for the degree of

Master of Science

in the
School of Computing Science
Faculty of Applied Sciences

APPROVAL

Name:	Iman Sarrafi
Degree:	Master of Science
Title of Project:	mrsFAST-Ultra: a compact, SNP-aware mapper for high performance sequencing applications

Examining Committee: Dr. Qianping Gu
Chair

Dr. Suleyman Cenk Sahinalp, Senior Supervisor

Dr. Martin Ester, Supervisor

Dr. Arrvindh Shriraman, SFU Examiner

Date Approved: June 24, 2013

Abstract

The high throughput sequencing (HTS) platforms generate unprecedented amounts of data that introduce challenges for processing and downstream analysis. While tools that report the "best" mapping location of each read provide a fast way to process HTS data, they are not suitable for many types of downstream analysis such as structural variation detection where it is important to report multiple mapping loci for each read. In fact, multi-mapping is the main bottleneck in genomic structural variation detection, RNA-Seq data analysis, etc. For this purpose we introduce mrsFAST-Ultra, a fast, cache oblivious, SNP-aware aligner that can handle the multi-mapping of HTS reads very efficiently. mrsFAST-Ultra improves mrsFAST, the first cache oblivious read aligner capable of handling multi-mappings, through new and compact index structures that reduce both the memory usage and number of CPU operations per alignment. The size of the index generated by mrsFAST-Ultra is 10 times smaller than that of mrsFAST. As importantly, mrsFAST-Ultra introduces new features such as being able to (1) obtain the best mapping loci for each read, and (2) return mapping locations for all reads that have at most k mapping loci (within an error threshold), for any user specified k. Furthermore mrsFAST-Ultra is SNP-aware, i.e., it can map reads to reference genome while discounting the mismatches that occur at common SNP locations provided by db-SNP; this significantly increases the number of reads that can be mapped to the reference genome. Finally mrsFAST-Ultra utilizes the presence of multiple cores and can be tuned for different memory settings.

Our results show that mrsFAST-Ultra is up to 4.5-times faster than its predecessor mrsFAST. In comparison to newly enhanced popular tools such as BWA and Bowtie2, it is more sensitive (it can report 60 times or more mappings per read) and much faster (5 times or more) in the multi-mapping mode.

To my beloved mother and father.

"Ignorance more frequently begets confidence than does knowledge: it is those who know little, and not those who know much, who so positively assert that this or that problem will never be solved by science."

— Charles Darwin

Acknowledgments

First of all, I would like to thank my senior supervisor, Dr. S. Cenk Sahinalp, for his extensive support, patience and guidance during my studies, and for the opportunity to work in his exceptional laboratory.

I would specially like to thank Faraz Hach for all his guidance and support during my studies, and for conceiving and leading the project which was used as a basis of this thesis, since this work would not be possible without his help.

I would like to thank the fellows of the SFU Computational Biology lab who I had the great pleasure of working with them. I am also thankful to my friends Ali, Narek, Ibrahim, Nataliya, Yasaman and all those who were my best friends and companions in this journey.

Last but not the least, I am thankful to my wonderful family for their unlimited support during my studies. I specially owe a great debt to my mother and father without whom none of this would be possible. That is why I dedicate this thesis to them.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

High Throughput Sequencing (HTS) technologies have changed the way genomics research is conducted since their inauguration in 2005 [21], and they continue to evolve with the introduction of single molecule sequencing and more recently the nanopore sequencing. Although the HTS technologies have proven their power in cataloging normal human genome variation [1, 2], finding disease causing mutations [23], and building *de novo* genome assemblies [9], the computational analysis of the data they generate is far from being perfect.

Aside from *de novo* sequencing projects, all HTS-based studies start with mapping the reads to a reference genome (of the same or closely related species). The importance of read mapping was recognized by the field, and various computational tools have been developed for different purposes and constraints. Aligners based on the Burrows Wheeler Transform (BWT) [6] and the Ferragina Manzini (FM) index [7], such as BWA [18], Bowtie [17], Bowtie2 [16], SOAP2 [19] and GEM [20] try to achieve high mapping speed with some reduction in mapping sensitivity, where hash based mappers such as mrFAST [5], mrsFAST [11], drFAST [14], RazerS [26], RazerS3 [27], SHRiMP [24], SRmapper [10] and GSNAP [28] aim full sensitivity at the cost of run time. Typically, the advantages and disadvantages of a read mapper is determined by the project needs, and the "best" aligner should be selected depending on the biological question in hand [8]. For example, in order to accurately detect structural variants in HTS data, especially in the repeat regions of a genome, one needs to obtain all mapping loci for each of the reads - which provides the main computational bottleneck [12].

Note that, even when fully sensitive mappers are used, many reads remain unmapped, primarily due to the sequencing errors associated with HTS platforms. In addition to these errors, some reads also involve real sequence variants such as SNPs [25], indels [22], and splits [15] or mapping ambiguity [4] due to structural variation. Available mapping tools require a (typically user defined) upper bound on the number of "errors" it can tolerate per read mapping, and treat real variants and sequencing errors identically - this reduces the mappability of a significant number of reads substantially. A mapper capable of distinguishing real variants from sequencing errors, will be able to map more reads to the reference, effectively providing an increased accuracy and sensitivity. Unfortunately, as each read is mapped independently from the others, and the genomic variants are detected only after the mapping process is complete, real variants cannot be known *a priori*. However, many of the 3 to 4.5 million SNPs in a human genome (in comparison to a reference genome) are shared among individual genomes [2], and have been collected and indexed in the dbSNP database. Therefore, a read mapper which utilizes the common SNP information in dbSNP (or any other genomic variation databases) can improve the signal-to-noise ratio in alignments.

In this project we introduce a new SNP-aware read mapper developed for the Illumina platform, that we call mrsFAST-Ultra, which improves the (1) mappability, (2) mapping accuracy, and (3) sensitivity by tolerating common, previously reported sequence variants and distinguishing them from likely sequencing errors. Given a user defined error threshold, mrsFAST-Ultra reduces the number of reads that could NOT be mapped by any available mapper by 19%. mrsFAST-Ultra achieves this while providing full sensitivity, i.e. it guarantees to find all mapping loci of each read within a user defined error threshold, as per its predecessors [5, 11, 14]. As mentioned earlier, this feature is essential for accurate structural variant detection techniques (e.g. VariationHunter [12, 13]). As a result, mrsFAST-Ultra has a significantly higher sensitivity compared to Bowtie 2 and BWA (the latest version) in the "all mapping mode" where mrsFAST-Ultra reports at least 30 times more mappings per read.

mrsFAST-Ultra introduces several additional improvements over its predecessors, such as (i) requiring a substantially smaller reference genome index file (which also improves its cache performance), (ii) introducing new filters to improve search space, and (iii) supporting multithreading. More specifically, mrsFAST-Ultra improves on the storage requirement of the original mrsFAST, the first cache-oblivious HTS read mapper, by a factor of 10. The index size was one of the limiting factors of the original mrsFAST in large scale sequencing projects. As mentioned above, the compactness of mrsFAST-Ultra's index structure also reduces the overall number of CPU operations and the I/O needs, resulting in a factor of

4.5 improvement in the running time in comparison to the original mrsFAST.

Finally, mrsFAST-Ultra introduces new features such as (1) the ability to retrieve the single best mapping loci, (2) the ability to retrieve all reads which map to at most (a user defined) k unique loci (within a user defined number of mismatches), and (3) automatic parallelization if multiple cores are available in the computing environment.

Chapter 2

Background

To get familiar with the working mechanism of mrsFAST-Ultra and the set of options and improvements it provides, we first introduce mrsFAST, the cache-oblivious seed and extend mapping tool which has been the basis of development for mrsFAST-Ultra. mrsFAST is a hash based multi-mapper that guarantees to find all mapping locations of a set of short reads from a donor genome in the reference genome within a user-specified number of mismatches. mrsFAST performs alignments by indexing both the reference genome and the input reads, and executing a cache-oblivious all-to-all comparison algorithm on the reference and read indexes.

2.1 Indexing the Reference Genome

mrsFAST creates an index array (figure 2.1) of the reference genome in a way that for any unique subsequence of size $k = l/(e+1)$ in the reference genome (where l is read length and e is the user defined error threshold e), there is an entry in the index. For each entry of this table, the list of all starting positions of each occurrence of this k-mer in the genome is stored. The items of this table are stored lexicographically with respect to these subsequences of size k.

2.2 Indexing the Donor Genome

To create an index of the donor genome, mrsFAST partitions reads of size l into $e+1$ non-overlapping blocks of size k. The pigeonhole principle guarantees that if a read is mapped

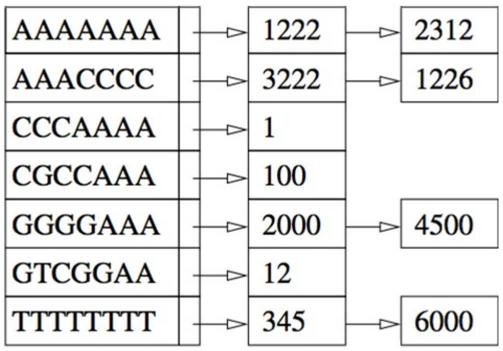

Figure 2.1: Sample snapshot of a hash index for the reference genome [11].

to a location in the reference genome within the error threshold e, then at least one of these blocks should map to the reference genome location with no mismatches.

mrsFAST creates an index array for the reads in a similar fashion to the reference index as follows. First mrsFAST partitions each read to non-overlapping blocks of size k and for each such block, then it creates an index entry where it stores every occurrence of this block in the reads by storing the read number as well as the block position. Figure 2.2 depicts a snapshot of such a read index.

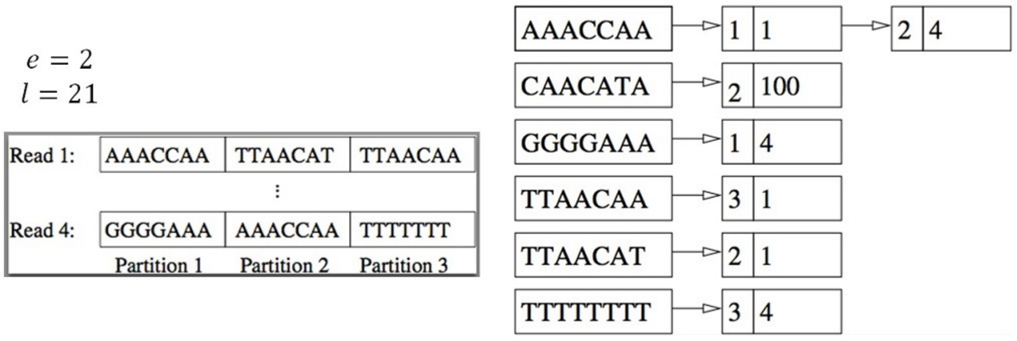

Figure 2.2: Sample snapshot of a read index for reads of length 21, and the error threshold 2 [11].

2.3 Search

To find all locations in the reference genome where each read can be mapped to within error threshold e, mrsFAST compares the entries of the reference index and the read index. For each block of a read, it finds all locations in the reference genome that have a matching subsequence. According to the position of the matching block within each corresponding read, mrsFAST extends that matching subsequence in the genome to the left and to the right so as to establish a full alignment between the read and the extended subsequence. If this alignment involves fewer than e mismatches (Hamming errors), then that particular location is returned as a match.

To perform a complete search for each read in the entire genome, mrsFAST needs to do an all-to-all comparison between the list of genome locations and the list of reads corresponding to each block. This all-to-all search is done with a recursive divide and conquer strategy that guarantees cache-obliviousness, meaning that the number of cache misses during this costly all-to-all comparison is minimum within a factor 2 approximation. As a result, mrsFAST guarantees to find all mapping locations of the input reads in the reference genome with high efficiency in terms of both cache performance and the CPU usage.

Chapter 3

Methods

mrsFAST-Ultra is a seed and extend aligner in the sense that it works in two main stages: (i) it builds an index from the reference genome for exact "anchor" matching and (ii) it computes all anchor matchings for each of the reads in the reference genome through the index, extends each match to both left and right and checks if the overall alignment is within the user defined error threshold.

3.1 Indexing

In the indexing step, mrsFAST-Ultra slides a window of size $k = l/(e+1)$ (where l is read length and e is the user defined error threshold e) through the reference genome and identifies all occurences of each k-mer present in the genome. For small values of k, mrsFAST-Ultra's genome index is an array of all possible k-mers in lexicographic order. For each k-mer, the index keeps an array of all locations the k-mer is observed in the reference genome. In case the value of k is prohibitively large, only a prefix of user defined size ℓ (for each k-mer) is used for indexing. For each such ℓ-mer, its locations on the reference genome are then sorted with respect to the $k-\ell$-mers following it. (In fact, for most applications, even keeping track of all $k-\ell$-mers following a particular ℓ-mer is not necessary: we just hash these $k-\ell$-mers via a simple checksum scheme.)

For further compacting the index, the reference genome itself is first converted to a 3 bit per base encoding. The genome sequence is stored in 8 byte long machine words implying that each machine word contains 21 bases. In addition, the index of the reference genome actually does not keep every occurence of each k-mer, but rather keeps how many occurences of each k-mer is present in the genome. Everytime we perform a search in the reference genome, the locations are computed on the fly. This reduces the I/O requirements of mrsFAST-Ultra significantly. One may think that such a set up would increase the overall running time of the search step but the savings from I/O reduction significantly offsets the cost of recalculating the k-mer locations on the fly. Overall, the storage requirement of the index we construct for the reference genome is 2GB, including the reference genome sequence itself. This represents a 10 fold improvement in the index storage requirement of the original mrsFAST.

3.2 Search

In this step, mrsFAST-Ultra processes the reads from an input HTS data set and computes "all" locations on the reference genome that can be aligned to each read within the user-defined error threshold e. mrsFAST-Ultra is a fully sensitive aligner meaning that it guarantees to find and report all mapping locations of a given read within e mismatches. mrsFAST-Ultra achieves this by partitioning the read into $e + 1$ non-overlapping fragments of length k for a given error threshold e. Due to the pigeonhole principle, at least one of these fragments should have an exactly matching k-mer of the reference genome in each location the read can be mapped to. The search step then validates whether each location of the reference genome with an exact k-mer match of the read is indeed a mapping location.

In order to perform the search step as fast as possible, mrsFAST-Ultra loads the genome index (see above) to the main memory and computes the locations of each k-mer on-the fly - for significant savings in I/O. For each k-mer, the number of locations in the reference genome is already stored in the index, thus we can preallocate the required memory for each array that keeps the locations of a given k-mer. Once this extended reference genome index is set up in the main memory, the remaining memory is allocated for the reads. At each subsequent stage, mrsFAST-Ultra retrieves sufficiently many (unprocessed) reads that can fit in the main memory and searches them in the reference genome simultaneously. (Alternatively, the user can specify an upper bound on the memory usage.) These reads are also indexed with respect to the $e + 1$ non-overlapping fragments of size k it extracts from each read. Basically, for each possible fragment of length k, the read index keeps the read ID, the fragment number and the direction the fragment is observed in the read. Once the read index is set, it is compared to the reference genome index, in a divide and conquer fashion as per mrsFAST, in order to achieve cache obliviousness. In other words, for each possible k-mer, the list of its occurences in the reference genome is compared against the list of its occurecens among the reads in a divide-and-conquer fashion (rather than linear fashion) to ensure an optimal cache performance at any level of the cache structure, within a factor 2 [11].

Because mrsFAST-Ultra aims to be fully sensitive, it needs to verify whether each reference genome location and each corresponding read that have the same k-mer have indeed an alignment within the user defined error tolerance. Note that, the value of k, set to $l/(e+1)$ can be too big for creating an index that has an entry for every possible k-mer from the 4 letter DNA alphabet. Thus the primary indexing is performed on a prefix of length $\ell = 12$ for each k-mer and all locations/reads that share this prefix are further sorted according to the $k-\ell$-mer succeeding this prefix. This is achieved by hashing the $k-\ell$-mer through a simple checksum scheme. As a result, the divide-and-conquer comparison of reference genome locations and reads is performed on those entries that have the same ℓ-mer and the same checksum value for the succeeding $k-\ell$-mer. The comparison for each genomic location and a read involves the calculation of the Hamming distance between the read and the k-mer location in the genome, extended by the appropriate length towards left and right. Before calculating the Hamming distance, mrsFAST-Ultra applies another filter that compares the number of As, Cs, Gs and Ts in the read and the genomic locus; if the total number of symbol differences is more than $2e$, then we do not need to compute the Hamming distance explicitly as it will be at least $e+1$ - above the error threshold. In comparison to the original mrsFAST, our new search strategy reduces the number of Hamming distance calculations, the main bottleneck for the search step, by a factor of 5. When combined with reduced I/O (due to compact index representation), a two stage index (for both the reference genome and the reads), and the introduction of new filters, this implies a 4.5 factor reduction in the overall running time of search.

3.3 SNP awareness

The user has the option of setting mrsFAST-Ultra to tolerate known SNP locations in the mappings: i.e. in this mode, SNPs in an alignment location simply do not contribute to the error count in the Hamming distance computation. For that, mrsFAST-Ultra parses dbSNP and generates a compact structure that it uses for mapping. Although conceptually simple, this feature is highly desired by users as it significantly reduces the number of reads that can not be mapped to anywhere in the reference genome. In this mode mrsFAST-Ultra reports the number of SNPs in addition to the number of mismatches per each mapping location.

3.4 Limited mapping

mrsFAST-Ultra provides the user the option of returning a single best mapping locus per read - which it performs much faster than computing all mapping loci. As per BWA, Bowtie2, SRmapper and others, a best mapping location (on the reference genome) is considered to be one which has the smallest number of differences with the read. In addition, mrsFAST-Ultra has the option to return only mapping loci of reads which map to at most n locations within the user defined error threshold. These features help the users to control the mapping multiplicity - which can grow prohibitively for further downstream analysis.

3.5 Parallelization

mrsFAST-Ultra is designed to utilize the parallelism offered by contemporary multicore architectures. The mapping task is simply partitioned into independent threads each of which is executed by a single core. For efficiency purposes, the only locks used by the threads are for allocating memory and I/O.

Chapter 4

Results

We report on experiments we performed on a single PC, equipped with an Intel(R) Xeon(R) CPU with 4 cores and 12GB of RAM.

4.1 Comparison with mrsFAST

The following experiments show how mrsFAST-Ultra has improved its predecessor mrsFAST in terms of speed, and that how different parameters of the input data affect the running times of these two versions.

First we investigate whether the number of input reads influences the time improvements provided by mrsFAST-Ultra. Table 4.1 shows the running times of mrsFAST and mrsFAST-Ultra on two datasets with different number of reads. As observed, by increasing the number of reads from 1 million to 2 million, the running times of both versions are approximately increased by a factor of 2, and in both cases mrsFAST-Ultra is almost 4.5 times faster than mrsFAST. Therefore the number of input reads is not an important factor regarding mrsFAST-Ultra's performance.

In the second experiment we show how the given error threshold changes the running times. Table 4.2 shows that on a set of 100 base pair reads, with error threshold 3, mrsFAST-Ultra is almost 10 times faster than mrsFAST, while the time improvement ratio is only 4.5 when error threshold is set to 6. This is due to the fact that by setting the error to 3, the size of read blocks $k = l/(e + 1)$ is longer, and therefore the checksum filter works far more effectively on these blocks.

Table 4.1: Comparing running times on 1M and 2M reads of length 100, with error threshold 6.

Software	1M reads	2M reads
mrsFAST	180m 34s	362m 36s
mrsFAST-Ultra	41m 58s	78m 57s

Table 4.2: Comparing running times on reads of length 100, with error thresholds 3 and 6.

Software	$e = 3$	$e = 6$
mrsFAST	153m 10s	362m 36s
mrsFAST-Ultra	15m 30s	78m 57s

To observe how the read lengths affect the running time improvements of mrsFAST-Ultra, in the third experiment two datasets with different read lengths (50 and 100 base pairs) are mapped to the reference genome. In both cases the dataset contains 2 million reads and error threshold is set to 3. As seen in table 4.3, when the read length is shorter the size of the read partitions $k = l/(e + 1)$ gets smaller and therefore the checksum filter looses its effectiveness. With read length 50, mrsFAST-Ultra performs only 2 times faster than mrsFAST, while it is about 10 times faster when the read length is 100.

4.2 Comparison with Other Mapping Tools

We benchmarked a number of read mapping software with parameters set as below, unless otherwise stated.

- mrsFAST v2.5.0.4 (-e 6, for error threshold)
- mrsFAST-Ultra v3.0.0 (-e 6, for error threshold, –threads 1 for using a single CPU)
- BWA v0.6.2 (-n 6 for error threshold; -N for disabling iterative search and reporting all mapping locations where required)
- Bowtie2 v2.0.2 (-k 100 for reporting up to 100 mappings for each read, -a for reporting all mapping locations where required [1])

[1]It was impractical to run Bowtie2 for all mapping location

Table 4.3: Comparing running times on reads of length 50 and 100, and error thresholds 3.

Software	$l = 50$	$l = 100$
mrsFAST	216m 6s	153m 10s
mrsFAST-Ultra	109m 31s	15m 30s

- GEM v1.367.beta (-m 6 for error threshold 6; -d for reporting all mapping locations where required)
- RazerS3 v3.1.1 (-i 94, provides %94 similarity for allowing 6 errors in reads of length 100, -rr 100 for full sensitivity)
- GSNAP 2013-01-23 release (-m 6 for error threshold 6)
- SRmapper v0.1.5 (-m 6 for error threshold 6)

Our results below are based on mapping 2 million Illumina reads of length 100bp from NA18507 (SRA ID: SRR034939) individual genome to the “hg19” version of the human reference genome. We carried out a few experiments to evaluate the performance of the above software. In the first experiment, we mapped the reads with an error threshold of 6% (i.e. 6bp). Table 4.4 depicts the results of this experiment. As can be seen, mrsFAST-Ultra reports about 308M mapping locations for these reads, which is at least 30-times more than the number of mapping locations reported by any of the competing methods. In the SNP-aware mode where we provided mrsFAST-Ultra with dbSNP32, the percentage of reads which could not be mapped drops by 19% (roughly 3% of all reads).

In the second experiment, we set the appropriate parameters in each method to report 100, 1000 and all mappings locations per read. Table 4.5 shows the running time of all four methods. Although the running time for GEM is better than the other methods, it misses many mapping locations as per BWA and Bowtie2.

In the third experiment, we ran all the tools in the ”best mapping” mode with various error thresholds. Although mrsFAST-Ultra is not as fast as some of the other tools, it has higher sensitivity than the others as shown in Table 4.6.

In the final experiment, we compare mrsFAST-Ultra and GSNAP in their SNP-tolerant best mapping mode. The results are given in Table 4.7.

Table 4.8 demonstrates the memory footprint of all tools we benchmarked on 2M reads. mrsFAST-Ultra occupies 2.5 GB by default settings. Though it could be set to work with

Table 4.4: Mapping 2M reads from NA18507 to hg 19 with e ≤ 6.

Software	Time	# mappings (millions)	% of reads mapped
mrsFAST	6h 2m	308.302	90.55
mrsFAST-Ultra	1h 18m	308.302	90.55
mrsFAST-Ultra (SNP-aware)[a]	1h 52m	354.691	93.17
BWA	7h 16m	4.133	91.57
Bowtie2[b]	2h 52m	5.130	91.52
GEM	16m	9.078	90.78
RazerS3[c]	10h 17m	10.631	92.00
GSNAP	3h 11m	5.388	86.37
SRmapper[d]	7m	0.32	10.46

BWA and GEM are set to report all mapping locations. Bowtie2 is impractical to run if it is set to report all mappings; other mappers do not provide such option. [a] The SNP-aware mrsFAST-Ultra employs dbSNP32 for this task; [b] For Bowtie2, we report the time when it is set to return at most 100 mappings per read; without this bound it does not complete the task in 24 hours. [c] RazerS3 cannot finish the task within 12hrs and its sensitivity is reduced to 99% (i.e. setting parameter -rr to 99); [d] SRmapper crashes on the full human genome. Results are shown only for mapping the reads to chr1.

a lower user-specified memory limit, which may lead to slower mapping time.

Finally, we show the effectiveness of mrsFAST-Ultra filters. In Figure 4.1, we calculated the expected number of the locations that should be verified given varying k-mer values. As expected, when the length of k-mer increases the expected number of locations that should be verified descreases. We also plot the average number of locations verified by mrsFAST-Ultra for various k-mer + checksum length values. As shown in the figure, using checksum filtration mimics the use of longer k-mer. In the figure we demonstrate the average number of the locations verified after we incorporated the 1-gram filtration method.

Table 4.5: Running time for reporting n mapping locations per read.

Software	n=100	n=1000	n=∞
mrsFAST-Ultra	53m	57m	78m
BWA	438m	436m	436m
Bowtie2[a]	172m	NA	NA
GEM	14m	13m	16m
RazerS3	538m	604m	617m
GSNAP	196m	191m	191m
SRmapper[b]	7m	7m	7m

[a] Bowtie2 can not complete the task in 24 hours for n≥1000. [b] SRmapper crashes on full human genome. Results are shown only for mapping the reads to chr1.

Table 4.6: Mapping of 2M reads in the best mapping mode, with an error threshold of 2, 4 and 6. No indels/gaps allowed in any method. We report on both the running time and the percentage of reads mapped.

Software	$e \leq 2$		$e \leq 4$		$e \leq 6$	
	Time	% of reads mapped	Time	% of reads mapped	Time	% of reads mapped
mrsFAST-Ultra	12m	80.97	21m	87.63	61m	90.55
BWA	4m	80.97	11m	87.52	18m	90.22
Bowtie2	10m	80.97	10m	87.52	10m	89.77
GEM	4m	80.97	6m	87.18	13m	89.33
RazerS3	13m	80.97	59m	87.63	325m	90.55
GSNAP	156m	77.78	180m	83.70	184m	86.16
SRmapper*	3m	7.29	5m	8.89	7m	10.46

* SRmapper only running on chr1

Table 4.7: Comparing mrsFAST-Ultra and GSNAP in SNP-tolerant best mapping mode.

Software	time	% of reads
mrsFAST-Ultra	82m	93.17
GSNAP	207m	85.30

Table 4.8: Memory footprint of the tools on 2M reads.

Software	Memory Footprint (GB)
mrsFAST-Ultra	2.5
BWA	3.2
Bowtie2	3.2
GEM	4.1
razerS	1.4
GSNAP	4.6
SRmapper*	0.2

* SRmapper footprint only on chr1

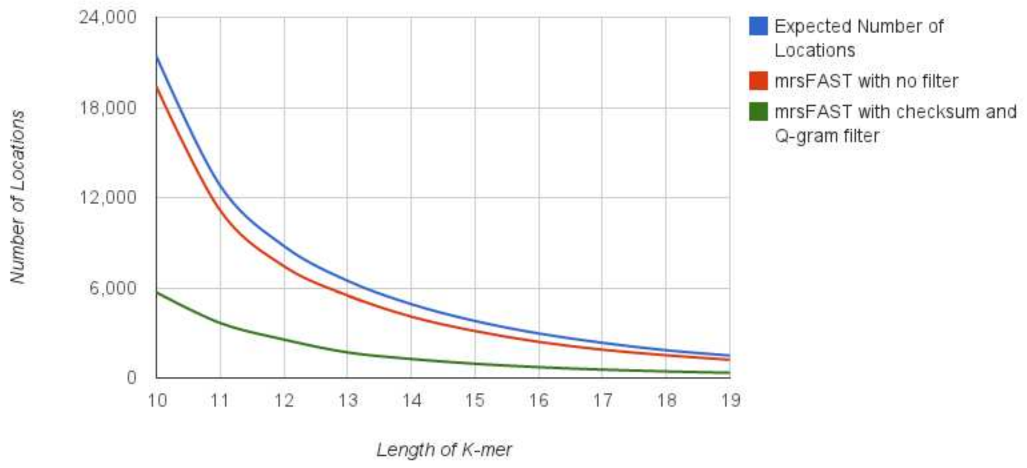

Figure 4.1: Average Number of locations verified per k-mer extracted from each read, as a function of k-mer length.

Chapter 5

Conclusion

As the first step of almost all High Throughput Sequencing applications (such as structure variation detection) reads from a donor genome need to be mapped to a reference genome. In addition to the human genome diversity caused by single-nucleotide variations and larger-sized structural variations such as insertions, deletions, inversions and segmental duplications, presence of sequencing errors associated with HTS platforms requires the alignment techniques to allow a user specified error threshold in the mappings. Since mapping is a time consuming step, and because the accuracy of many structural detection discovery tools largely depends on finding all mapping locations of the input reads, efficient and accurate multi-mapping of reads is of great importance in these applications.

In this report we introduced mrsFAST-Ultra, a new SNP-aware read mapper which improves accuracy and sensitivity by tolerating single nucleotide polymorphisms (SNPs). While maintaining full sensitivity, mrsFAST-Ultra significantly improves its predecessor in terms of running time, memory, disk usage, CPU and I/O operations. It also introduces several new features such as multithreading, limited mapping, best mapping, and adjustment to user-specified memory limits.

5.1 Future Work

mrsFAST-Ultra is specifically designed for Illumina generated reads and requires all the reads in an input set to have exactly the same length. Supporting reads with arbitrary lengths in a single data set could improve the flexibility of the software. In addition, mrsFAST-Ultra is designed for mapping short reads (up to approximately 500 bp). Running the current version of mrsFAST-Ultra for mapping reads of considerably higher lengths (up to 1Mbp) will be inefficient due to the proportionally small size of hash seeds in the search step. Accordingly, mrsFAST-Ultra can be extended to support such longer reads. For this purpose, "split read mapping" is required. In the split read mapping problem, long reads should broken into smaller segments which need to be separately mapped to the reference genome. Then these smaller segments are compared and combined in order to obtain possible mappings for the original long reads.

Another possible addition to mrsFAST-Ultra is the option of supporting chimeric alignments. Chimeric alignment allows opposite sides of long reads to be mapped to far apart locations in the genome which could even fall in different chromosomes. Note that such mappings imply occurrence of structural variation events in the donor genome. Again, split mapping is the key to implementation of chimeric alignments.

Bibliography

[1] 1000 Genomes Project Consortium. A map of human genome variation from population-scale sequencing. *Nature*, 467(7319):1061–1073, Oct 2010.

[2] 1000 Genomes Project Consortium. An integrated map of genetic variation from 1,092 human genomes. *Nature*, 491(7422):56–65, Nov 2012.

[3] A. Ahmadi, A. Behm, N. Honnalli, C. Li, L. Weng, and X. Xie. Hobbes: optimized gram-based methods for efficient read alignment. *Nucleic Acids Res.*, 40(6):e41, Mar 2012.

[4] C. Alkan, B. P. Coe, and E. E. Eichler. Genome structural variation discovery and genotyping. *Nat. Rev. Genet.*, 12(5):363–376, May 2011.

[5] Can Alkan, Jeffrey M Kidd, Tomas Marques-Bonet, Gozde Aksay, Francesca Antonacci, Fereydoun Hormozdiari, Jacob O Kitzman, Carl Baker, Maika Malig, Onur Mutlu, S Cenk Sahinalp, Richard A Gibbs, and Evan E Eichler. Personalized copy number and segmental duplication maps using next-generation sequencing. *Nature Genetics*, 41(10):1061–1067, 2009.

[6] M. Burrows and D. J. Wheeler. A block-sorting lossless data compression algorithm. Technical report, DEC Labs, 1994.

[7] P. Ferragina and G. Manzini. Opportunistic data structures with applications. In *FOCS*, pages 390–398, 2000.

[8] N. A. Fonseca, J. Rung, A. Brazma, and J. C. Marioni. Tools for mapping high-throughput sequencing data. *Bioinformatics*, 28(24):3169–3177, Dec 2012.

[9] S. Gnerre, I. Maccallum, D. Przybylski, F. J. Ribeiro, J. N. Burton, B. J. Walker, T. Sharpe, G. Hall, T. P. Shea, S. Sykes, A. M. Berlin, D. Aird, M. Costello, R. Daza, L. Williams, R. Nicol, A. Gnirke, C. Nusbaum, E. S. Lander, and D. B. Jaffe. High-quality draft assemblies of mammalian genomes from massively parallel sequence data. *Proc. Natl. Acad. Sci. U.S.A.*, 108(4):1513–1518, Jan 2011.

[10] P. M. Gontarz, J. Berger, and C. F. Wong. SRmapper: a fast and sensitive genome-hashing alignment tool. *Bioinformatics*, 29(3):316–321, Feb 2013.

[11] Faraz Hach, Fereydoun Hormozdiari, Can Alkan, Farhad Hormozdiari, Inanc Birol, Evan E Eichler, and S Cenk Sahinalp. mrsFAST: a cache-oblivious algorithm for short-read mapping. *Nature Methods*, 7(8):576–577, 2010.

[12] F. Hormozdiari, C. Alkan, E. E. Eichler, and S. C. Sahinalp. Combinatorial algorithms for structural variation detection in high-throughput sequenced genomes. *Genome Res.*, 19(7):1270–1278, Jul 2009.

[13] F. Hormozdiari, I. Hajirasouliha, P. Dao, F. Hach, D. Yorukoglu, C. Alkan, E. E. Eichler, and S. C. Sahinalp. Next-generation VariationHunter: combinatorial algorithms for transposon insertion discovery. *Bioinformatics*, 26(12):i350–357, Jun 2010.

[14] Farhad Hormozdiari, Faraz Hach, S Cenk Sahinalp, and Can Alkan. Sensitive and fast mapping of di-base encoded reads. *Bioinformatics*, 27(14):1915–1921, 2011.

[15] E. Karakoc, C. Alkan, B. J. O'Roak, M. Y. Dennis, L. Vives, K. Mark, M. J. Rieder, D. A. Nickerson, and E. E. Eichler. Detection of structural variants and indels within exome data. *Nat. Methods*, 9(2):176–178, Feb 2012.

[16] B. Langmead and S. L. Salzberg. Fast gapped-read alignment with Bowtie 2. *Nat. Methods*, 9(4):357–359, Apr 2012.

[17] B. Langmead, C. Trapnell, M. Pop, and S. L. Salzberg. Ultrafast and memory-efficient alignment of short DNA sequences to the human genome. *Genome Biol.*, 10(3):R25, 2009.

[18] Heng Li and Richard Durbin. Fast and accurate short read alignment with Burrows-Wheeler transform. *Bioinformatics*, 25(14):1754–1760, 2009.

[19] R. Li, C. Yu, Y. Li, T. W. Lam, S. M. Yiu, K. Kristiansen, and J. Wang. SOAP2: an improved ultrafast tool for short read alignment. *Bioinformatics*, 25(15):1966–1967, Aug 2009.

[20] S. Marco-Sola, M. Sammeth, R. Guigo, and P. Ribeca. The GEM mapper: fast, accurate and versatile alignment by filtration. *Nat. Methods*, 9(12):1185–1188, Dec 2012.

[21] M. Margulies, M. Egholm, W. E. Altman, S. Attiya, J. S. Bader, L. A. Bemben, J. Berka, M. S. Braverman, Y. J. Chen, Z. Chen, S. B. Dewell, L. Du, J. M. Fierro, X. V. Gomes, B. C. Godwin, W. He, S. Helgesen, C. H. Ho, C. H. Ho, G. P. Irzyk, S. C. Jando, M. L. Alenquer, T. P. Jarvie, K. B. Jirage, J. B. Kim, J. R. Knight, J. R. Lanza, J. H. Leamon, S. M. Lefkowitz, M. Lei, J. Li, K. L. Lohman, H. Lu, V. B. Makhijani, K. E. McDade, M. P. McKenna, E. W. Myers, E. Nickerson, J. R. Nobile, R. Plant, B. P. Puc, M. T. Ronan, G. T. Roth, G. J. Sarkis, J. F. Simons, J. W. Simpson, M. Srinivasan, K. R. Tartaro, A. Tomasz, K. A. Vogt, G. A. Volkmer, S. H. Wang, Y. Wang, M. P. Weiner, P. Yu, R. F. Begley, and J. M. Rothberg. Genome sequencing in microfabricated high-density picolitre reactors. *Nature*, 437(7057):376–380, Sep 2005.

[22] R. E. Mills, W. S. Pittard, J. M. Mullaney, U. Farooq, T. H. Creasy, A. A. Mahurkar, D. M. Kemeza, D. S. Strassler, C. P. Ponting, C. Webber, and S. E. Devine. Natural genetic variation caused by small insertions and deletions in the human genome. *Genome Res.*, 21(6):830–839, Jun 2011.

[23] B. J. O'Roak, P. Deriziotis, C. Lee, L. Vives, J. J. Schwartz, S. Girirajan, E. Karakoc, A. P. Mackenzie, S. B. Ng, C. Baker, M. J. Rieder, D. A. Nickerson, R. Bernier, S. E. Fisher, J. Shendure, and E. E. Eichler. Exome sequencing in sporadic autism spectrum disorders identifies severe de novo mutations. *Nat. Genet.*, 43(6):585–589, Jun 2011.

[24] Stephen M Rumble, Phil Lacroute, Adrian V Dalca, Marc Fiume, Arend Sidow, and Michael Brudno. SHRiMP: Accurate Mapping of Short Color-space Reads. *PLoS Computational Biology*, 5(5):11, 2009.

[25] M. Stoneking. Single nucleotide polymorphisms. From the evolutionary past.. *Nature*, 409(6822):821–822, Feb 2001.

[26] D. Weese, A. K. Emde, T. Rausch, A. Doring, and K. Reinert. RazerS–fast read mapping with sensitivity control. *Genome Res.*, 19(9):1646–1654, Sep 2009.

[27] David Weese, Manuel Holtgrewe, and Knut Reinert. Razers 3: Faster, fully sensitive read mapping. *Bioinformatics*, 28(20):2592–2599, 2012.

[28] T. D. Wu and S. Nacu. Fast and SNP-tolerant detection of complex variants and splicing in short reads. *Bioinformatics*, 26(7):873–881, Apr 2010.