Fast and Accurate Alignment with BatAlign

Chandana Tennakoon[1, 2,3], Jing-Quan Lim[1], Wing-Kin Sung[1, 2,3]
[1] Department of Computer Science, National University of Singapore, Singapore 117417
[2] NUS Graduate School for Integrative Sciences and Engineering, (CeLS), #05-01, 28 Medical Drive, Singapore 117456
[3]Genome Institute of Singapore, 60, Biopolis Street, #02-01 Genome, Singapore 138672

INTRODUCTION

In the last decade, rapid development of sequencing technologies has opened up unprecedented opportunities for the study of human genome. However, the large volume of the reads, the errors in sequencing and variations in the sampled genomes presents challenges that need to be addressed when aligning these reads to a reference genome. In this poster we introduce BatAlign, an algorithm that can align next generation sequencing reads rapidly and accurately taking into account both mismatches and indels.

METHODS

BatAlign is a Burrows-Wheeler transform (BWT) based aligner. It uses two strategies called reverse alignment and deep-scan for accurate alignment. For rapid alignment of reads, BatAlign utilises the fast k-mismatch mapping algorithm BatMis, a novel BWT-based data structure for handling indels and a fast SIMD based Smith-Waterman alignment algorithm for seed extension.

Reverse Alignment

Seed-based aligners search for candidate hits of its seeds; then, these hits are extended and the best alignment is selected based on a set of pre-defined criterion. In contrast, *Reverse-alignment* does the opposite by searching for the best possible hits in the reference first. Given a read R, *Reverse-alignment* incrementally finds the hits of R with the most likely combination of mismatches and indels. We define a function F such that F(i) = (p_i, q_i), where p_i and q_i are non-negative integers representing the number of mismatches and indels in an alignment respectively. If F(a) < F(b), then the probability of the correct alignment of a read having (p_a, q_a) mismatch/indel combination is higher than that of (p_b, q_b) mismatch/indel combination. *Reverse-alignment* incrementally tries to map R allowing F(i) mismatch/indel combinations for i = 1,..,9.

Determining F

In real-life, the likelihood of an indel in a genome is an order of magnitude less than that of a SNV. The likelihood of finding multiple indels within a read becomes small if the length of R is shortened. Empirical studies show that, for Illumina and SOLiD, the majority of mismatch errors are due to sequencing errors. A general heuristic for such platforms is to set the mismatches in a read due to sequencing errors and/or SNVs to be about ~5% of the read length. Furthermore, indels occur at a rate of ~0.02%. Based on these statistics, for a read of length around 75 bp, we can set one indel and four mismatches as a reasonable upper bound for the number of indels and mismatches to be allowed in a mapping. For the default mode of enumerating candidate hits, we have 9 levels where F(1)=(0,0), F(2)=(1,0), F(3)=(2,0), F(4)=(3,0), F(5)=(4,0), F(6)=(5,0), F(7)=(0,1), F(8)=(1,1) and F(9)=(2, 1).

Deep-scan

The best-scoring alignment according to the function F need not be the correct alignment. It is best if we can get the set of next-best alignments too. With these additional hits and using the quality information of the mapping, we might be able to find the correct alignment. *Deep-scan* enumerates

hits according to F. If F(k) is the first successful mismatch/indel combination found during *Reverse-alignment*, and there are multiple hits, we return all these hits. Otherwise, if there is a unique hit and k < 9, we return all hits having the mismatch/indel combinations F(k) and F(k + 1).

Bat-Align algorithm

We will first describe the Bat-Align algorithm for a short read (75 bp). It consists of three steps. First it will perform a *Deep-scan* for each read R to build a set of candidate alignments. Next, if a set of candidate alignments can be found, each hit is assigned an alignment score based on quality information and the unique, highest scoring hit is reported (if it exists). Finally, a quality score is assigned to each reported hit.

If the read length is less than 150bp, the first 75bp is used as a seed and aligned as described above. Then this alignment is extended to the full read. For reads longer than or equal to 150 bp, we will split the read into non-overlapping 75 bp reads. Each of the 75 bp segments will be aligned as described above. If the best hit from each segment is non-repetitive and fall within the locality of each other, we will try to align the original read onto this region of the reference. Otherwise, Bat-Align will examine and align the whole read onto the putative locations reported by each of the segments. Among these alignments, the best-scoring hit is reported.

RESULTS

We compared the results of mapping the 101bp real life paired end dataset SRR315803 with a set of popular aligners. The reverse and forward reads were mapped independently with each aligner, and a mapping was marked as concordant if its mate falls within 1000bp apart from each other with the correct orientation. Otherwise the pair is marked as concordant. Figure 1 shows the ROC curves for all the aligners, drawn using the number of concordant and discordant alignments stratified by the mapping quality of the forward read.

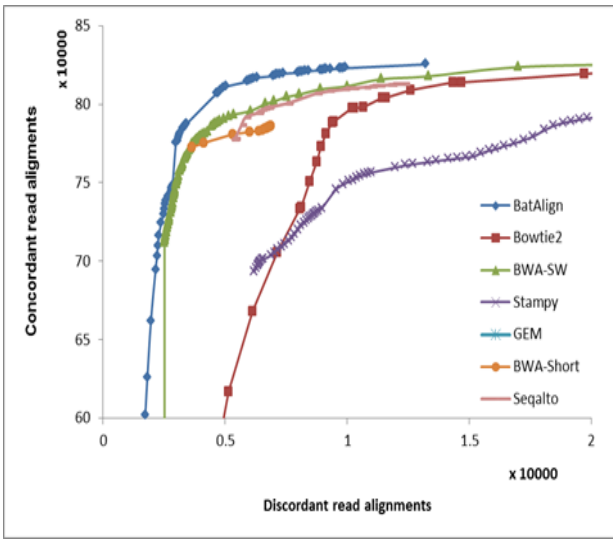

Figure 1

Program	Runtime (seconds)	Speedup factor
Bat-Align – Default	583	6.3x
Bat-Align – Fast	481	7.7x
Bat-Align – Turbo	331	11.2x
Bowtie2	459	8.0x
BWA-Short	598	6.2x
BWA-SW	639	5.8x
GEM	214	**17.3x**
SeqAlto	677	5.5x
Stampy	3694	**1.0x**

Table 1

BatAlign can run in several modes, trading accuracy for speed. Among the aligners compared, Stampy was the slowest and GEM was the fastest. If we take Stampy as the baseline for measuring speed, BatAlign can run upto 11times faster than Stampy. In its other modes of operation BatAlign can work at speeds comparable to the popular aligners BWA and Bowtie.