

PerM: Efficient Mapping of Short Sequencing Reads with Periodic Full Sensitive Spaced Seeds

Yangho Chen, Tade Souaiaia, and Ting Chen*

Program in Computational Biology and Bioinformatics, University of Southern California, 1050 Childs Way, Los Angeles, CA 90089-2910, USA

Associate Editor: Dr. Joaquin Dopazo

ABSTRACT

Motivation: The explosion of next generation sequencing data has spawned the design of new algorithms and software tools to provide efficient mapping for different read lengths and sequencing technologies. In particular, ABI's sequencer (SOLiD system) poses a big computational challenge with its capacity to produce very large amounts of data, and its unique strategy of encoding sequence data into color signals.

Results: We present the mapping software, named PerM (Periodic Seed Mapping) that uses periodic spaced seeds to significantly improve mapping efficiency for large reference genomes when compared to state-of-the-art programs. The data structure in PerM requires only 4.5 bytes per base to index the human genome, allowing entire genomes to be loaded to memory while multiple processors simultaneously map reads to the reference. Weight maximized periodic seeds offer full sensitivity for up to three mismatches and high sensitivity for four and five mismatches while minimizing the number random hits per query, significantly speeding up running time. Such sensitivity makes PerM a valuable mapping tool for SOLiD and Solexa reads.

Availability: <http://code.google.com/p/perm/>

Contact: tingchen@usc.edu

1 INTRODUCTION

Next-generation sequencing technology has created the need for highly efficient methods to align short DNA reads (1). Current technologies (2), e.g. the Illumina and SOLiD platforms, are capable of generating hundreds of millions of short reads in a single run (3) (4). These breakthroughs have led to many important biological applications, including the identification of transcription factor binding sites (Chip-Seq) (5), estimation of RNA expression levels (RNA-Seq) (6) and SNP calling (7).

Applied Biosystems ligation mediated SOLiD sequencer has the unique property of collecting color signals for the transitions between nucleotides. Each single nucleotide polymorphism changes two transitions (i.e. consecutive colors), providing a large advantage in the detection of SNPs because only a fraction of color changes represent possible SNPs (4) and single color discrepancies can be regarded as sequencing errors. While this unique encoding results in base accuracy as high as 99.94% (4) it also requires algorithms

capable of finding alignments of less similarity than those designed for sequencers such as Solexa which output signals directly from nucleotides.

Most read mapping programs are designed to be full sensitive to 'k' mismatches, meaning all alignments within k mismatches will be reported. Original alignment tools such as Blast (8) and Blat (9), are capable of finding highly sensitive alignments for long reads, but do not provide full sensitivity to specific numbers of mismatches and are incapable of efficiently mapping the amount of reads currently produced by short read sequencing machines. Many mapping programs have been designed to handle large amounts of short reads, including ELAND (Cox, 2006, unpublished), MAQ (10), ZOOM (11), RMAP (12) and SeqMap (13), which preprocess and index read sets and then scan the reference for potential matches. Programs including SOAP (14), Pass (15), MOM (16), ProbMatch (17), SXOligoSearch (18), Mosaik (19), BWA (20) and Bowtie (21) preprocess and index the reference then search for potential matches among the reads.

In many SOLiD data sets (22) (Results 3.3), more than half of the sequenced reads will not align to the reference with fewer than three mismatches. For this reason, programs designed specifically for SOLiD data, including the Corona Lite pipeline (23) and SOCS (22) provide full sensitivity to alignments including three or more mismatches. Such sensitivity usually comes at a cost; the shorter subsequences used to provide full sensitivity to high numbers of mismatches also slow mapping such that genome scale mapping may not be possible.

PerM ameliorates these difficulties through the design of periodic spaced seeds to maximize efficiency for many distance measures, including those specific to the location of polymorphism in color space. For example, rather than accepting the slower performance associated with sensitivity to multiple mismatches, we introduce a faster method which provides full sensitivity to one potential SNP locus (consecutive mismatches) and a free color error. Periodic seeds allow PerM to maximize efficiency while maintaining full sensitivity for up to three mismatches with about 88% and 67% partial sensitivity for four and five mismatches. (See Supplement section 3)

Most read-mapping programs use the idea of "seeds" for preprocessing and matching (24). A seed is a set of selected positions within a window which generates fixed length subsequences when slid along a string. When a seed is aligned to a read or a genomic sequence, selected positions are concatenated to form a

* to whom correspondence should be addressed

Fig. 1. Conventional seeds used by ELAND, SOAP and MAQ divide a 32-bp read into four substrings. For any alignment within two mismatches, at least one of six pairs of substrings will match exactly. This method requires three hash tables and six lookups for each read and direction (forward or reverse complement).

fixed length subsequence which can be used to extract similar reads or genomic substrings. These subsequences serve as a filter, only when a read and a genomic substring share the same subsequence will a "hit" be declared and further examination be carried out to determine the actual similarity level. When subsequences are short, the seed is said to have "low weight" and the probability of subsequences matching by chance becomes greater. For large genomes and data sets these "random hits" will often be the bottleneck of the running time.

The most widely used short-read mapping algorithm, implemented in many programs including ELAND, MAQ (10), SOAP (14), Corona Lite (23), and SOCS (22) divides each read into $k+m$ fragments to provide full sensitivity to k mismatches. Then $\binom{k+m}{m}$ hashing steps are used to check for exact matches in the different combinations of m fragments. If a read aligns to the reference with k or fewer mismatches then one of the $\binom{k+m}{m}$ subsequences will match exactly. The larger the value chosen for m , the greater the seed weight, but more index tables and hashing steps are required. For example, Corona Lite (23) chooses $m = 3$ while SOCS (22) chooses $m = 1$. However, the most common choice is $m = 2$, used by ELAND, MAQ (10), and SOAP (14). This method is displayed in (Figure 1). The read is divided into four equal size fragments, resulting in the hashing of six substring pairs for each read. These six pairs of substrings are encompassed into three seeds which can be used to preprocess the genome or the set of reads into three index tables.

Instead of using seeds composed of equal sized substrings, our method is based on idea of the spaced seed proposed by Burkhardt et al. (25), Ma et al. in PatternHunter (26) and Kucherov et al. (27). A spaced seed is a set of "care" and "don't care" positions, annotated as "1"s and "*"s respectively. For example, PatternHunter suggested 111*1**1*1**11*111 as a good spaced seed with weight (number of "care" positions) eleven. Mathematical results (28) (29) have shown that spaced seeds are more sensitive than consecutive seeds (those without "don't care" positions) in finding local similarities between two strings. Kucherov et al. (27) attempted to minimize the number of multiple spaced seeds necessary to achieve different levels of sensitivity. Lin et al. (11) used multiple spaced seeds for short read mapping and provided bounds for the number of lookups necessary to achieve full sensitivity for varying seed lengths and weights. They showed that for a thirty-two base pair read, seeds with weight of sixteen require at least six lookups to obtain full

Fig. 2. The single periodic spaced seed full sensitive to two mismatches over a 32bp-read. For any alignment within two mismatches, at least one out of the seven subsequences will match exactly. This seed is composed of repeating the pattern (111*1**).

sensitivity for two mismatches. They implemented this idea into a program, called ZOOM (11).

Shrimp (30) also uses spaced seeds to find hits, however they find alignments with InDels as well as mismatches which requires significantly longer running time.

PerM uses single periodic weight-maximized spaced seeds. An example, shown in Figure 2, is composed of four repeating patterns of (111*1**) whose length is seven and is full sensitive to two mismatches. When this seed is applied to a 32bp read, it generates seven subsequences, one of which will match any string within two mismatches of the read. It should be noticed that the length of the repeating pattern (seven in the example) is equal to the number of subsequences generated.

In *Methods* we describe how PerM's periodic seeds allow increases in mapping efficiency and sensitivity when compared to the conventional consecutive seeds used by many other programs.

2 METHODS

2.1 Seed notation:

C_k The conventional seed family which divides reads into $k + 2$ fragments (used in ELAND, MAQ and SOAP) to provide full sensitivity to k mismatches.

F_k The maximum-weight periodic spaced seed family which is full sensitive to k mismatches.

$S_{x,k}$ The special weight maximized periodic seed family for mapping SOLiD reads, full sensitive to x SNP candidates (consecutive mismatches) and k free mismatches.

2.2 Motivation for Periodic Design

For any full sensitive seed family there exists two performance criteria, memory load and running time. Memory load is a directly related to the number of index tables required to be built during preprocessing. PerM's single periodic seeds store a single index for each locus, requiring only 4.5 bytes per base to index the entire genome. Running time is dependent linearly on the number of queries or slides required of a seed but can be influenced exponentially by the seed weight (number of "care" positions). Specifically, periodic seeds that are composed of repeating patterns of length $|P|$ will generate, for each read, $|P|$ subsequences of weight w for query, where w is the seed weight. An approximation for the expected number of random hits per read is $2|G||P|/4^w$,

where $|G|$ is the length of the genome and the number 2 is necessary for the forward and reverse compliment.

For large-scale genome projects, if the seed weight is not sufficiently large the number of random hits will grow by a factor of four each time the weight is decreased by one. Thus, we make our goal the design of single high weight periodic seeds with moderate pattern lengths. Sections 2.3 - 2.4 describe how the use of periodic seeds provides a generalizable framework to quickly develop high performing maximum weight seeds for a variety of distance measures.

2.3 Periodic Seeds: Generalization, Indexing, and Extendability

Generalization for different read lengths That full sensitive periodic seeds generalize to all lengths is a function of their repeating pattern. For example, sliding the following seed (length 28) six times generates seven subsequences that provide full sensitivity to two mismatches for a 34bp read.

$$(111 * 1 **)(111 * 1 **)(111 * 1 **)(111 * 1 **)$$

By the definition of full sensitivity, all pairs of positions i and j will be covered pairwise with "don't care" (*) positions in at least one of the slides. Ignoring boundary effects, we can examine the internal read positions 8 – 14 in Table 1 when the above spaced seed is applied (Slide 0) and slid six times (Slides 1-6). In total, each of the $\binom{7}{2} = 21$ pairs of positions is covered pairwise with "*" exactly once. Therefore, this pattern is locally optimal, providing local full sensitivity to two mismatches.

Table 1. The periodic spaced seed, applied to a read and slid through positions 8-14 six times, covers every of the 21 pair of positions exactly once.

Positions	8	9	10	11	12	13	14	Covering 21 pairs of positions
Slide 0	1	1	1	*	1	*	*	(11,13) (11,14) (13,14)
Slide 1	*	1	1	1	*	1	*	(8,12) (8,14) (12,14)
Slide 2	*	*	1	1	1	*	1	(8,9) (8,13) (9,13)
Slide 3	1	*	*	1	1	1	*	(9,10) (9,14) (10,14)
Slide 4	*	1	*	*	1	1	1	(8,10) (8,11) (10,11)
Slide 5	1	*	1	*	*	1	1	(9,11) (9,12) (11,12)
Slide 6	1	1	*	1	*	*	1	(10,12) (10,13) (12,13)

In fact, local full sensitivity of this pattern implies global full sensitivity of the periodic spaced seed because every position in the read shares a similar relationship with some position within 8-14. Formally, for any read position i (excluding the boundaries of the reads), there exists a position j , $8 \leq j \leq 14$, where $i \bmod 7 = j \bmod 7$, such that when j is aligned to "*", so is i . Thus, if every pair of positions within a pattern are covered pairwise with "*" in some slide then every pair of positions within the read is pairwise covered with "*" at some slide. Note that this property is also applied to the boundary positions which are more often aligned to "*" .

This pattern of length seven can be generalized to produce the periodic spaced seeds for any read length, ex: read lengths of 25 and 36 as follows:

$$111*1**(111*1**(111*1*)), \\ (111*1**(111*1**(111*1**(111*1**))11.$$

This generalizability of repeating patterns in periodic seeds holds not just for 2 mismatches but for all k mismatches, listed as a lemma in the work of Kucherov et al (27). Most importantly, generalizability drastically simplifies the seed-design algorithm and provides the framework for variable seed extension, which provides a significant increase in seed weight while maintaining a single index table.

Efficient indexing to allow seed extension To achieve full sensitivity a periodic seed will slide $|P| - 1$ times and generate $|P|$ subsequences on a read. As shown in Figure 2 this is accomplished by keeping open the last $|P| - 1$ positions at the end of the read. However, as periodic seeds maintain full sensitivity when generalized, these empty positions are not required. This is shown with a different example in Figure 4, each slide seeds can be extended to the end of the read so that all but the last are likely to have increased seed weight. In the example shown in Figure 4, seed extension results in weight increasing from the minimum of 14 to between 14 and 19 which corresponding to an approximately five times faster mapping performance for the whole human genome.

To take advantage of the extended seed weight without additional requirements to memory, PerM combines hashing with the binary search. Our special data structure consists of a hash table which stores pointers to the starting position of each bucket in an index table which stores the addresses of reads or the genome locations. In the example shown in Figure 4, we preprocess a genome using the maximum weight seed with $W = 19$ to generate one subsequence for every position in the genome. The prefix of each subsequence (e.g. the first 14 bases) is used to hashed the location into a bucket in the index table. Inside each bucket, multiple subsequences with the same prefix are sorted lexicographically and stored in the index table. Note that within each bucket, we can search for exact matches of variable-length strings using the binary search. Then, we apply the extended variable-weight seeds to each read to generate variable-length subsequences, and use the extended bases after 14 to find the exact matches using the binary search in the index table. This method results in seed weight which varies from 14 to 19 in the example shown in Figure 4.

2.4 Seed-search algorithm

The weight of periodic spaced seeds can be optimized by calculating the maximum-weight patterns of length $|P|$ which achieve full sensitivity to some distance measure. The small search space allows us to easily enumerate all reasonable size patterns of length $|P|$ to find which has maximum weight. Table 2 lists the maximum weights of the optimal patterns found for various numbers of consecutive color mismatches (x) and single mismatches (k) for various pattern lengths $|P| = 6, \dots, 15$.

For any fixed values for k and x , the design of periodic space seeds has to consider which pattern length provides the best seed. Notice that a longer pattern provides a greater weight/length ratio asymptotically, but the relationship is not monotonic, as shown in the curve for $k = 2$ in Figure 3. On the other hand, longer pattern lengths will increase the number of required queries, so our seed design considers the shortest pattern whose weight to length ratio is large enough to provide a tolerable number of random hits. In the case of two mismatches, local optima occur at $|P| = 7$ and $|P| = 13$, which provide ratios of $4/7$ and $9/13$ respectively. Here

Fig. 3. This figure shows the optimal weight-length ratios for different pattern lengths.

we choose the pattern with $|P| = 7$ because it requires six fewer queries per read. This pattern is used to design F_2 , which is shown in Figure 2, generating seven queries when applied to a 32bp read. We prove that the F_2 seed is the maximum weight spaced seed for seven lookups in the supplement. This proof agrees with the results shown by Kucherov et al (27), that for moderate read lengths, optimal seeds are usually periodic. This method is also used to design F_3 , F_4 and the SOLiD specific family of seeds $S_{x,k}$.

Table 2. The maximum weights of patterns that are full sensitivity to x SNPs and k free mismatches.

Sensitivity Threshold	Periodic Pattern Length $ P $									
	6	7	8	9	10	11	12	13	14	15
$k = 2$	3	4	4	5	6	7	8	9	9	10
$x = 1, k = 1$	2	2	3	4	5	5	6	7	8	8
$k = 3$	2	2	3	3	4	5	5	6	6	7
$x = 2, k = 0$	1	2	2	3	4	5	5	6	7	8
$k = 4$	1	1	1	2	3	3	3	4	4	5

2.5 Implementation and Bitwise Encoding

Quality scores and Paired End Reads If reads include a corresponding quality score file PerM will evaluate mappings by their alignment scores according to the quality score of each base. PerM can also filter out poor quality reads before mapping. If paired end reads and bounds for their separation distance are given as input, PerM will output both the best alignments within and beyond the separation bounds. Each alignment will be stored in a separate file.

Base to color encoding PerM encodes each read into two 64-bit words with two bits for each base, that is A = (0,0), C = (0,1), G = (1,0) and T = (1,1). We adopt RMAP’s method (12) to encode these two bits separately in two words. Similarly, each base on the reference genome is encoded three bits, with the third bit indicating whether the locus is masked as character N . This encoding method enables us to quickly check the number of mismatches between two bit-strings, using two ‘XOR’ and one ‘OR’ bitwise-operations.

Fig. 4. This figure shows the extension of periodic spaced seed $S_{1,1}$, composed of the repeating pattern (1111 * *1 * **), to multiple variable-weight spaced seeds that are applied to a 34-color SOLiD reads where 0, 1, 2 and 3 represent the four colors. The original seed is shown in the black boxes, and the extended variable-weight seeds are highlighted at the tail by the dashed boxes. The weight w is shown at the beginning for each extended seeds

Fig. 5. Dinucleotide Colors Signals Encoded from the Base Encoding.

For SOLiD reads, the color signals are also encoded in a similar way, with Blue = (0,0), Green = (0,1), Yellow = (1,0) and Red = (1,1). This encoding enables a quick translation of encoded base to encoded color signals by using bitwise operations of ‘SHIFT’ and ‘XOR’. In the example shown in Figure 4, a five-base read ‘ATGGA’ is encoded as two binary strings, U = (01110) and V = (01001). The color signals of this read, ‘Red Green Blue Yellow’ encoded as U’ = (1001) and V’ = (1100), can be obtained by U’ = U XOR (U SHIFT 1) and V’ = V XOR (V SHIFT 1). This allows the reference genome to be saved as an encoded string of bases while the corresponding color spaced encoding can be read out in few bitwise operations.

Color mutations for SNPs Applied Biosystems lists rules (31) to check whether two consecutive mismatches are valid in their indication a base substitution (SNP). Only one third of possible combinations of consecutive color mismatches are valid for a base substitution. For example, if ‘AATAA’ is encoded into the colors ‘BRRB’ and mapped to ‘BBBB’ the reference may have resulted from ‘AAAA’ meaning a SNP from ‘A’ \leftrightarrow ‘T’ causing two identical color mismatches, Blue \leftrightarrow Red. However, if ‘BRRB’ is mapped to ‘BBGB’, a SNP cannot be present because the nucleotide sequence which corresponds to ‘BBGB’ is ‘AAACC’. Instead, this is likely the result of two color sequencing errors. For ease of explanation, the combinations of consecutive mismatches are classified into three types:

- Type I Blue \Leftrightarrow Red or Green \Leftrightarrow Yellow
- Type II Blue \Leftrightarrow Green or Red \Leftrightarrow Yellow
- Type III Blue \Leftrightarrow Yellow or Green \Leftrightarrow Red

Only when both color mismatches are of the same type does it indicate a valid SNP. These three types of consecutive mismatches correspond to the three classes of base substitutions, (1) Transversion I [A \Leftrightarrow T or G \Leftrightarrow C] (2) Transversion II [A \Leftrightarrow C or G \Leftrightarrow T] (3) Transition, respectively.

The validation of SNP candidates is simple given our encoding method. Given two colors, one encoded into two bits B1 and B2, and the other into two bits b1 and b2, the three types of SNPs can be determined by the following bitwise operations:

- Type I: iff (B1 XOR b1) AND (B2 XOR b2) = 1
- Type II: iff (NOT (B1 XOR b1)) AND (B2 XOR b2) = 1
- Type III: iff (B1 XOR b1) AND (NOT (B2 XOR b2)) = 1

3 RESULTS

3.1 Results of Seed Design

As shown in the experimental results, the periodic spaced seeds used in PerM outperform the seeds used in ELAND inspired seeds used in MAQ in terms of mapping speed and sensitivity for both Illumina and SOLiD data. Table 3 displays our fixed weight periodic spaced seeds generalized to 34-color SOLiD reads. F_k denotes a seed full sensitive to k mismatches, while $S_{x,k}$ denotes a SOLiD-specific seed full sensitive to x consecutive color mismatches (SNPs) and k free color mismatches.

Table 3. PerM’s single periodic spaced seeds for SOLiD 34-color reads.

Seed name	Seed patterns parenthesized according to their repeats	Seed weight
F_2	(111*1**)(111*1**)(111*1**)(111*1**)	16
$S_{1,1}$	(1111**1****)(1111**1****)(1111*)	14
F_3	(111*1**1****)(111*1**1****)(11)	12
$S_{2,0}$	(1111**1****)(1111**1****)(11)	12
F_4	(11***1****)(11***1****)(11***)	8

Table 3 groups F_3 and $S_{1,1}$ into one category because both are full sensitive to reads with one SNP and one color error. However, $S_{1,1}$ achieves higher weight than F_3 weight by taking advantage SNP’s signature in color space. The introduction of positional restriction at one mismatch at the SNP locus significantly reduces the number of combinations of three mismatches, leading to the higher seed weight. Similarly, $S_{2,0}$ and F_4 are both full sensitive to two SNPs, but $S_{2,0}$ provides an increase of four (12 to 8) in seed weight. Thus, the design of seeds specifically for the color space will provide a significant advantage in mapping speed.

3.2 Theoretical Performance of Periodic Spaced Seeds

Table 4 further compares the F -seed family, the S -seed family, and the conventional C_k seed method described in the introduction. Our comparison is based on the most common implementation where

the human genome is preprocessed and 34-color SOLiD reads are divided into $k + 2$ fragments for mapping.

Table 4. Three seed families are compared in their ability to map 34-color SOLiD reads to a preprocessed human genome

Seed name	# of Index Tables	# of Queries Per Read	Seed Weight	Extended Weights	E(Random Hits) Per Read
F_2	1	7	16	16 to 20	1.89
C_2	3	6	16		8.38
$S_{1,1}$	1	10	14	14 to 19	68.91
F_3	1	11	12	12 to 16	627.25
C_3	4	10	12		3576.28
$S_{2,0}$	1	11	12	12 to 16	534.42
C_4	5	15	10		85.830
F_4	1	10	8	8 to 11	216.007

An index table is, by definition, an array of N index for a genome of N base pairs. Thus, C_2 requires three index tables as shown in Figure 1.

Memory Requirements As shown in Table 4, PerM’s use of a single seed results in the requirement of a single index table to preprocess the human genome no matter the sensitivity requirement, compared to three to five index tables for the conventional method C_k . The use of single periodic spaced seeds allows us to preprocess the human genome efficiently into 4.5 bytes per base, and load it to 14 gigabytes of memory, without the swapping of index tables between disk and memory.

Running time The total running time of a mapping project can be divided into two major components:

1. Preprocessing: the time to pre-processing the reference genome (or the reads set) into one or more index tables.
2. Mapping: the total time to find matches in the index tables for all queried subsequences, and the time to examine all matches using the full read-genome substrings alignments.

PerM’s requirement of a single index table results in faster preprocessing time than methods which use the conventional multi-seed, multi-table approach. Mapping time consists of two parts: the time to query each seed-induced subsequence and to validate matches which result in true alignments, and the time to examine and ignore matches that result from random hits. The former is fixed as the number of true alignments is constant given a particular sensitivity level, while the latter is related directly to the seed weight. Ideally mapping time is largely spent on the matching and validation of true alignments, but if the seed weight is insufficient, the examination of random hits will dominate running time. As listed in last column of Table 4, the expected numbers of random hits per read can grow so large that most of the running time is wasted on filtering out the random hits. For example, using F_3 on the human genome will require the examination of approximately 627 random hits per read which will result in drastically slower performance than the F_2 which is expected to examine fewer than

two random hits per read. Thus, the number of random hits is the most important indicator of the actual running time. Table 4 shows that the weight increase associated with extended variable-weight periodic spaced seeds will result in a large reduction in random hits and significantly faster running times. As expected this increased efficiency is greatest for the $S_{x,k}$ family, when we compare $S_{2,0}$ with C_4 , we expect 161 times fewer random hits for the periodic seed.

3.3 Experimental Results

We performed genome scale comparison with two popular mapping programs, MAQ (version 0.6.6) (10) and Bowtie (version 0.10.0) (21). Both Illumina and SOLiD reads from The 1000 Genomes Project used for mapping to the human genome. We also compared PerM to SOCS, a mapping program designed specifically for ABI SOLiD reads.

Genome scale mapping with SOLiD reads We mapped five million 35 color SOLiD reads (the first five million reads in the NCBI data set ERR000455 available on our web site) to the whole human genome. Overall we were able to map 58% of the reads (2.94M) with five or fewer mismatches. Over 78% of the mapped reads included at least one mismatch and 22% of the mapped reads have four or five mismatches in their best alignment to the reference, indicative of a high machine error rate. Considering that each SNP causes consecutive mismatches, it's likely that the majority of reads which cover SNP loci will contain at least three color mismatches. Thus, the detection of genomic variation with SOLiD reads requires far greater sensitivity than necessary for Illumina data. For this reason, PerM offers the seed $S_{1,1}$ to maximizing seed weight while still maintaining full sensitivity to reads which contain one SNP and a color error. For each seed, Table 5 lists the discovery rate for alignments containing five or fewer mismatches. The small difference in the discovery rates of $S_{1,1}$ and F_3 provides further reason to use $S_{1,1}$ for SOLiD data.

Table 5. The results of mapping 5 Million 34 color SOLiD reads to the whole human genome.

Seed name	Mapped reads			Unique SNP-supporting reads	
	3 mis	4 mis	5 mis	Mis Threshold	Read count
F_2	298,898	167,048	117,964	≤ 3 colors	74,877
$S_{1,1}$	465,460	348,416	257,281	≤ 3 colors	98,325
F_3	496,401	379,936	283,971	≤ 3 colors	98,325

All PerM seeds provide a minimum of full sensitivity to two mismatches and report 637, 681 exact matches, and 583, 363 and 561, 029 reads with one and two mismatches respectively.

Although MAQ and Bowtie do not provide alignment options which adhere to the definition of full sensitivity to greater than two mismatches, the version of each which come closest to PerM's mapping sensitivity was used for comparison. MAQ's version for SOLiD data finds hits using the seed C_2 (shown in Figure 1), and checks each for the possibility of a feasible base-color alignment. Bowtie was run in the mode "-v k -a -best -strata", capable of reporting each of the "best" alignments (fewest mismatches) provided that they contain k or fewer color errors and do not require

excessive backtracking. Values of two and three were used for "k" and the backtracking limit was set at the default level, 125. Bowties fastest mode, which reports only a single low mismatch alignment if one exists, was not used for the SOLiD data because the consecutive mismatch SNP signature, the high error rate, and the large number of ambiguous reads in SOLiD data sets result in a significant loss of information when poor alignments and multiple mappings are not considered.

It should also be noted that SOLiD reads include thirty-five colors preceded by the base of the primer used to synthesize the read. Thus, the first color represents the transition from the primer (which is not part of the reference) to the first base on the reference, leaving only thirty four colors which represent transitions on the reference. MAQ's current implementation uses only these thirty four colors for alignment. Unfortunately, each thirty-four color read could be the result of four different base strings, depending on the first base synthesized in the read. PerM uses the primer-color transition to infer the identity of this base and includes an extra check to insure that the first base in the reference matches the first base on the read. Bowtie does not currently include an implementation for SOLiD data, for this comparison we provided an translation to color space similar to that used in MAQ's SOLiD mapping package.

Table 6. Running time comparison of mapping the 35bp SOLiD reads to the whole human genome.

Program	Seed / mode	weight	(Full) Sensitivity	speed
PerM	F_2	16-20	2 colors	3.53 M / hours
PerM	$S_{1,1}$	14-19	1 base + 1 color	1.17 M / hours
PerM	F_3	12-16	3 colors	0.75 M / hours
MAQ	-c	14	2 colors	0.56 M / hours
Bowtie	-a -best -strata		-v 2 *	0.32 M / hours
Bowtie	-a -best -strata		-v 3 *	0.22 M / hours

*Bowtie does not guarantee full sensitivity, instead it reports alignments within "k" mismatches which do not require excessive backtracking.

Table 6 compares the mapping efficiency of MAQ, Bowtie, and PerM for different sensitivity thresholds. In general PerM offers faster running time for SOLiD reads than both MAQ and Bowtie at different sensitivity thresholds. In addition, only PerM is able to offer full sensitivity to more than two mismatches, as full sensitive spaced seeds require no backtracking limit or possibility of missed alignments.

Genome scale mapping with Illumina reads We mapped 9.9 million Illumina reads from whole human genome shotgun fragments (NCBI data set SRR001154) to the whole human genome with full sensitivity to two and three mismatches. The reads, originally of length 47bp were also trimmed down to 40bp and 36bp to provide a better comparison to MAQ and Bowtie. After trimming, 71.5%, 62.7%, and 36.7% of the 36bp, 40bp and 47bp reads could be mapped within four mismatches to the genome with our F_3 seed. MAQ which only offers full sensitivity to two mismatches was compared to the seed F_2 . Bowtie was run in it's default mode which outputs the first alignment encountered with two or fewer mismatches as well as conditions which are most similar to offering full sensitivity to two and three mismatches, "-v 2 -a -best -strata"

and "-v 3 -a -best -strata". These modes were compared to the seeds F_2 and F_3 .

Table 7. Running time comparison of mapping the Illumina reads with different read lengths and seeds to the whole human genome.

Length	36bp		40bp		47bp				
	Seed	Weight	Reads/Hr	Weight	Reads/Hr	Weight	Reads/Hr		
F_2	18-21		5.92M	20-24		8.01M	24-28		20.1M
MAQ	14		0.49M	14		0.55M	14		0.67M
Bowtie -v2*			4.43M			3.87M			2.64M
F_3	13-18		1.69M	15-19		2.21M	18-23		3.27M
Bowtie -v3*			4.28M			3.38M			1.63M
Bowtie default			9.27M			7.95M			7.20M

The default mode of Bowtie is equivalent to -k 1. The -v k mode is set with -a -best -strata. The tests are performed on Sun, X4600, Opteron, 2.6GHz, using 15GB single node and thread.

Table 7 compares the performance for mapping different length Illumina reads with sensitivity to two or three mismatches. For the popular task of mapping 36bp reads with full sensitivity to two mismatches, PerM runs approximately twelve times faster than MAQ. Bowtie, which cannot offer full sensitivity is slower than PerM when two mismatches are reported but outperforms PerM when three mismatches are allowed for 36bp reads. However, as read lengths grow longer, PerM significantly outperforms Bowtie and MAQ for sensitivity to both two and three mismatches. It should be noted also that Bowtie's default mode is faster than PerM's F_2 seed for read length 36. This advantage results from Bowtie's requirement to find and output only a single alignment in the default mode.

Comparison: PerM and MAQ As shown in Table 6 and Table 7, PerM is able to map both SOLiD and Solexa reads significantly faster than MAQ while offering greater levels of full sensitivity. The differences in performance are a testament to the benefit of extendable periodic spaced seeds which provide greater seed weights than the fixed length consecutive seeds implemented in MAQ. This increase in seed weight allows PerM to avoid the bottleneck which results from the many random hits present on a large genome. It should also be noted that MAQ preprocesses reads, requiring it to build an index table for each mapping project while PerM can reuse the same index because it preprocesses the genome.

Comparison: PerM and Bowtie Although PerM and Bowtie both index the genome, PerM finds full sensitive alignments through seed subsequence matching while Bowtie uses a modified exact matching algorithm to report alignments that require fewer than a threshold number of backtracking steps. As shown in Table 7, Bowtie's performance slows down when aligning longer reads because of the increased number of backtracking steps required to find inexact alignments. PerM's performance is affected quite differently. PerM's performance is primarily a result of its seed weight, which is maximized for all sensitivity levels. Longer reads allow PerM's seeds to be extended to offer more weight and speed up mapping. As PerM's seed weight is constant across mapping projects, Table 6 shows PerM's performance to be less affected

by the higher number of mismatches prevalent in SOLiD data than Bowtie which performs significantly slower. PerM, developed with the SOLiD system in mind, provides a ranking and bitwise checking system for the identification of SNP candidates in color space as well as the capability to report alignments containing four and five mismatches. Bowtie, which cannot report alignments containing more than three mismatches, uses a "quality-aware" backtracking algorithm, favoring mismatches on low quality positions and increasing the probability that alignments which suggest true SNP loci will be ignored. These differences make PerM the more capable alignment tool for reads from the SOLiD sequencer.

Comparison: PerM with SOCS PerM was compared to a program dedicated to SOLiD reads, SOCS (version 1.2.1). Both PerM and SOCS provide full sensitivity to three mismatches, but SOCS does not provide sufficient seed weight to map reads to the entire genome. For this reason, we mapped the five million 35bp SOLiD reads used in 3.3.1 to chromosome X. Eight percent of the reads included a mapping to chromosome with three or fewer substitutions. Table 8 lists the mapping times for different sensitivity levels. PerM's faster running time in comparison to SOCS is primarily the result of much higher seed weight.

Table 8. Comparing of PerM and SOCS in chromosome level reference.

Full Sensitivity	PerM		SOCS	
	Running Time	Weight	Running Time	Weight
2 color mis	11 min 46 sec	16-20	14 min 30 sec	11
1 base + 1 color mis	23 min 0 sec	14-19		
3 color mis	32 min 41 sec	12-16	2 hrs 20 min	8

The running time includes preprocessing and I/O. The memory usage for both program are lower than 2G bytes. The tests are performed on Sun, X4600, Opteron, 2.6GHz, using single node and thread.

Genome preprocessing Genome preprocessing time is linear with respect to the size of the reference regardless of the seed used. PerM requires 3 hours 30 min to index the whole human genome to 14GB of memory. In comparison Bowtie requires 4 hours and 47 minutes to build a compressed 2.7GB human genome index. However, for large genome re-sequencing projects, preprocessing time is negligible in comparison to mapping time. Once the genome has been preprocessed, its index can be shared and reused by multiple processors to map different read sets. Both PerM and Bowtie use multiple cores (32) in one computer to map read sets in parallel by querying the same genome index table in the shared memory. Thus, on a server with shared memory architecture, PerM is more memory efficient in terms of "memory per CPU" compared to MAQ, despite a 14GB index table.

4 DISCUSSION

PerM provides highly efficient mapping solutions for genome scale mapping projects involving Illumina or SOLiD data. PerM owes its performance primarily to the use of single periodic spaced seeds which are capable of providing sufficient weight and sensitivity to significantly increase genome scale mapping performance in comparison to other mapping programs.

However for applications that require full sensitivity to many mismatches ($k \geq 4$) on a short read, single periodic seeds may prove incapable of providing efficient mapping performance. In this situation the costly step of hashing to multiple index tables may be necessary to increase seed weight and eliminate a bottleneck in the checking step. Already a topic of much interest (33) (34), (35) (36) (37), Ma et al. (38) showed the optimization of multiple seeds cannot be easier than the Golomb Ruler Design problem, considered likely to be NP-hard. Thus, although we cannot guarantee optimality over the entire search space, we propose three methods to design high weight multiple seeds: a constrained exhaustive search, a reduction to the integer programming problem, and a "tuples-grouping" algorithm. These methods and additional performance analysis and experiments are discussed at detail in the supplement.

ACKNOWLEDGEMENT

The authors would like to thank Dr. Zhenyu Xuan and Dr. Richard McCombie from the Genome Center of Cold Spring Harbor Laboratory and Dr. Andrew Smith at USC for providing us with data and helpful suggestions. We would also like to thank Zheng Zhang from ABI as well as graduate students Dazhe Meng and Zach Frazier from USC for help discussion.

Funding: This work was supported by the NIH Center of Excellence in Genomic Sciences.

REFERENCES

- [1] J Shendure and H Ji, "Next-generation dna sequencing", *Nat Biotechnol*, vol. 26, no. 10, pp. 1135–1145, Oct 2008.
- [2] M Ronaghi, M Uhlén, and P Nyrén, "A sequencing method based on real-time pyrophosphate", *Science*, vol. 281, no. 5375, pp. 363–363, Jul 1998.
- [3] S Bennett, "Solexa ltd", *Pharmacogenomics*, vol. 5, no. 4, pp. 433–438, Jun 2004.
- [4] Applied Biosystems, "Principles of di-base sequencing and the advantage of color space analysis in the solid system", World Wide Web electronic publication, 2008.
- [5] E R Mardis, "Chip-seq: welcome to the new frontier", *Nat Methods*, vol. 4, no. 8, pp. 613–614, Aug 2007.
- [6] J C Marioni, C E Mason, S M Mane, M Stephens, and Y Gilad, "Rna-seq: an assessment of technical reproducibility and comparison with gene expression arrays", *Genome Res*, vol. 18, no. 9, pp. 1509–1517, Sep 2008.
- [7] A R Quinlan, D A Stewart, M P Strömberg, and G T Marth, "Pyrobayes: an improved base caller for SNP discovery in pyrosequences", *Nat Methods*, vol. 5, no. 2, pp. 179–181, Feb 2008.
- [8] S F Altschul, W Gish, W Miller, E W Myers, and D J Lipman, "Basic local alignment search tool", *J Mol Biol*, vol. 215, no. 3, pp. 403–410, Oct 1990.
- [9] W J Kent, "Blat—the blast-like alignment tool", *Genome Res*, vol. 12, no. 4, pp. 656–664, Apr 2002.
- [10] H Li, J Ruan, and R Durbin, "Mapping short DNA sequencing reads and calling variants using mapping quality scores", *Genome Res*, Sep 2008.
- [11] H Lin, Z Zhang, M Q Zhang, B Ma, and M Li, "Zoom! zillions of oligos mapped", *Bioinformatics*, vol. 24, no. 21, pp. 2431–2437, Nov 2008.
- [12] A D Smith, Z Xuan, and M Q Zhang, "Using quality scores and longer reads improves accuracy of solexa read mapping", *BMC Bioinformatics*, vol. 9, pp. 128–128, 2008.
- [13] H Jiang and W H Wong, "Seqmap: mapping massive amount of oligonucleotides to the genome", *Bioinformatics*, vol. 24, no. 20, pp. 2395–2396, Oct 2008.
- [14] R Li, Y Li, K Kristiansen, and J Wang, "SOAP: short oligonucleotide alignment program", *Bioinformatics*, vol. 24, no. 5, pp. 713–714, Mar 2008.
- [15] D Campagna, A Albiero, A Bilardi, E Caniato, C Forcato, S Manavski, N Vitulo, and G Valle, "Pass: a program to align short sequences", *Bioinformatics*, vol. 25, no. 7, pp. 967–968, Apr 2009.
- [16] H L Eaves and Y Gao, "Mom: maximum oligonucleotide mapping", *Bioinformatics*, vol. 25, no. 7, pp. 969–970, Apr 2009.
- [17] Y J Kim, N Teletia, V Ruotti, C A Maher, A M Chinnaiyan, R Stewart, J A Thomson, and J M Patel, "Probematch: rapid alignment of oligonucleotides to genome allowing both gaps and mismatches", *Bioinformatics*, vol. 25, no. 11, pp. 1424–1425, Jun 2009.
- [18] Malaysian Genomics Resource Center, "Sxoligosearch", World Wide Web electronic publication, 2009.
- [19] The MarthLab, "Mosaik", World Wide Web electronic publication, 2009.
- [20] H Li and R Durbin, "Fast and accurate short read alignment with burrows-wheeler transform", *Bioinformatics*, vol. 25, no. 14, pp. 1754–1760, Jul 2009.
- [21] Mihai Pop Ben Langmead, Cole Trapnell and Steven L. Salzberg, "Ultrafast and memory-efficient alignment of short dna sequences to the human genome", World Wide Web electronic publication, 2008.
- [22] B D Ondov, A Varadarajan, K D Passalacqua, and N H Bergman, "Efficient mapping of applied biosystems solid sequence data to a reference genome for functional genomic applications", *Bioinformatics*, vol. 24, no. 23, pp. 2776–2777, Dec 2008.
- [23] Applied Biosystems, "Solidtm system application documentation: Ab resequencing analysis pipeline (corona lite)", World Wide Web electronic publication, 2008.
- [24] Gene Myers, "A fast bit-vector algorithm for approximate string matching based on dynamic programming", *J. ACM*, vol. 46, no. 3, pp. 395–415, 1999.
- [25] S Burkhardt and J Karkkainen, "Better filtering with gapped q-grams", *In Proceedings of the 12 th Symposium on Combinatorial Pattern Matching (CPM'01)*, July 2001.
- [26] B Ma, J Tromp, and M Li, "Patternhunter: faster and more sensitive homology search", *Bioinformatics*, vol. 18, no. 3, pp. 440–445, Mar 2002.
- [27] G Kucherov, L Noé, and M Roytberg, "Multiseed lossless filtration", *IEEE/ACM Trans Comput Biol Bioinform*, vol. 2, no. 1, pp. 51–61, Jan-Mar 2005.
- [28] Y. Sun J. Buhler, U. Keich, "Designing seeds for similarity search in genomic dna", *RECOMB*, Mar 2003.
- [29] J Xu, D Brown, M Li, and B Ma, "Optimizing multiple spaced seeds for homology search", *J Comput Biol*, vol. 13, no. 7, pp. 1355–1368, Sep 2006.
- [30] S M Rumble, P Lacroute, A V Dalca, M Fiume, A Sidow, and M Brudno, "Shrimp: accurate mapping of short color-space reads", *PLoS Comput Biol*, vol. 5, no. 5, May 2009.
- [31] Applied Biosystems, "Principles of di-base sequencing and the advantage of color space analysis in the solid system", World Wide Web electronic publication, 2008.
- [32] The OpenMP Architecture Review Board, "Open mp application program interface", World Wide Web electronic publication, 2009.
- [33] François Nicolas and Eric Rivals, "Hardness of optimal spaced seed design", *J. Comput. Syst. Sci.*, vol. 74, no. 5, pp. 831–849, 2008.
- [34] M Li, B Ma, D Kisman, and J Tromp, "Patternhunter II: highly sensitive and fast homology search", *J Bioinform Comput Biol*, vol. 2, no. 3, pp. 417–439, Sep 2004.
- [35] "Efficient methods for generating optimal single and multiple spaced seeds", in *BIBE '04: Proceedings of the 4th IEEE Symposium on Bioinformatics and Bioengineering*, Washington, DC, USA, 2004, p. 411, IEEE Computer Society.
- [36] L Noé and G Kucherov, "Improved hit criteria for dna local alignment", *BMC Bioinformatics*, vol. 5, pp. 149–149, Oct 2004.
- [37] Y Sun and J Buhler, "Designing multiple simultaneous seeds for dna similarity search", *J Comput Biol*, vol. 12, no. 6, pp. 847–861, Jul-Aug 2005.
- [38] Hongyi Yao Bin Ma, "Seed optimization is no easier than optimal golomb ruler design", *APBC*, 2008.