technische universität dortmund
fakultät für informatik

Variant-tolerant read mapping with locality-sensitive hashing

Johannes Fischer,
Sven Rahmann,
Dominik Köppl,
Dominik Kopczynski,
Henning Timm

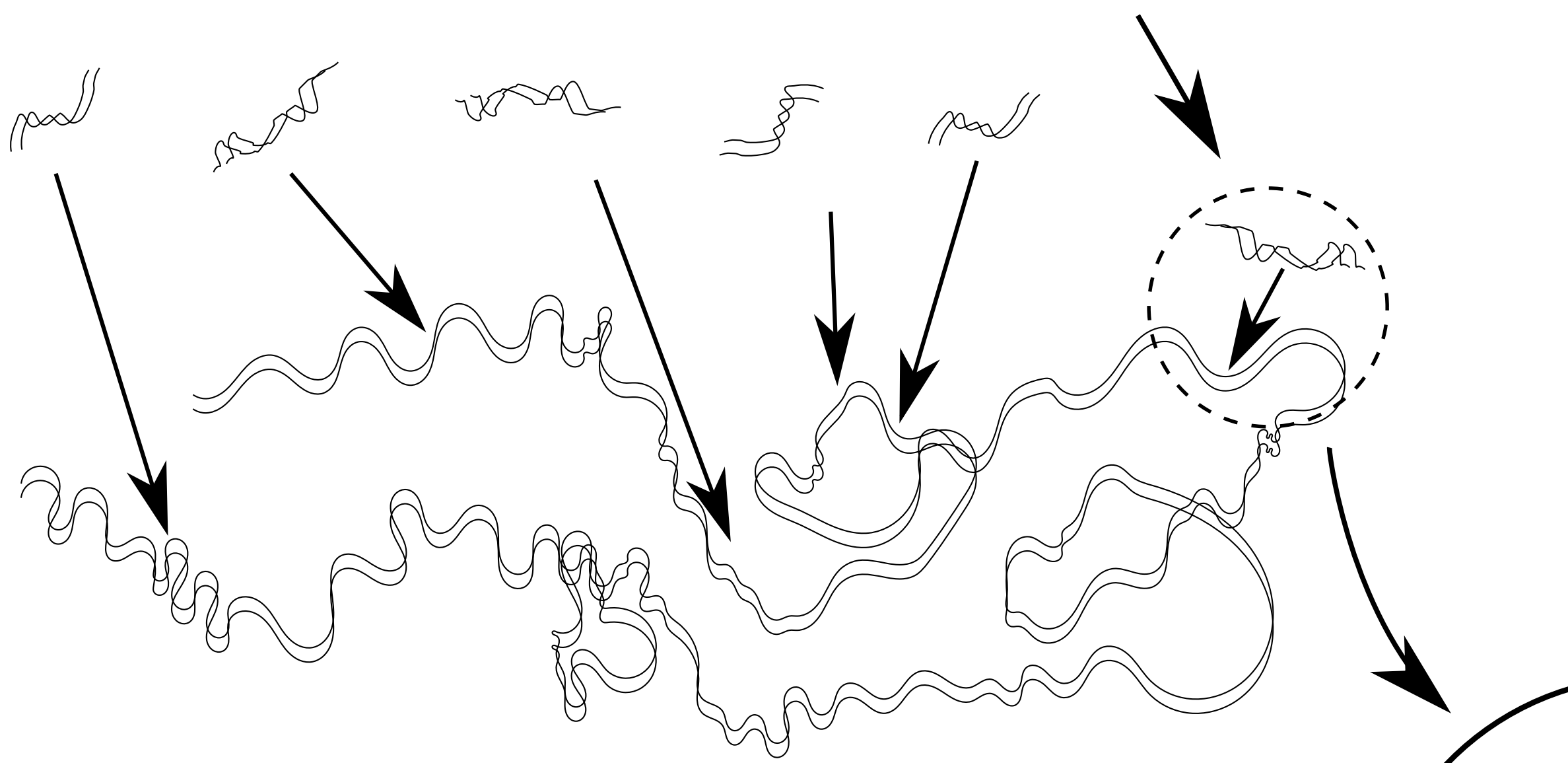
Project group 583

Marcel Bargull,
Kada Benadjema,
Benjamin Kramer,
David Losch,
Jens Quedenfeld,
Sven Schrinner,
Jan Stricker

Abstract

The rapid development of genomic sequencing technologies in the past decades has outgrown the advances of computing power and therefore requires efficient read mapping algorithms.[2] Read mappers align sequenced reads to a reference genome, where a set of reads aligned to the same position can hint at possible mutations. Even though many fast read mappers have been published in the recent years, most of them do not consider common variants of the reference genome. Variant tolerance highly increases accuracy of read mappers when aligning reads against a species' pangenome.

We have developed a new read mapper for variant tolerant alignment by usage of hash based filtering in combination with an alignment algorithm based on dynamic programming. In the first step, we use locality-sensitive hashing (LSH), initially designed for finding similarities in documents, for candidate filtering.[1] We treat reads and windows of the reference genome as documents, which are compared by LSH. As a result, we obtain an approximative mapping to the reference regions. This leads to a dramatic reduction of the reference length and therefore semi-global alignment becomes feasible. The aligner handles variants like SNPs, insertions and deletions and decides which variants lead to the best alignment. New genetic variants and gene mutations can be found by observing the mismatches from the alignments.

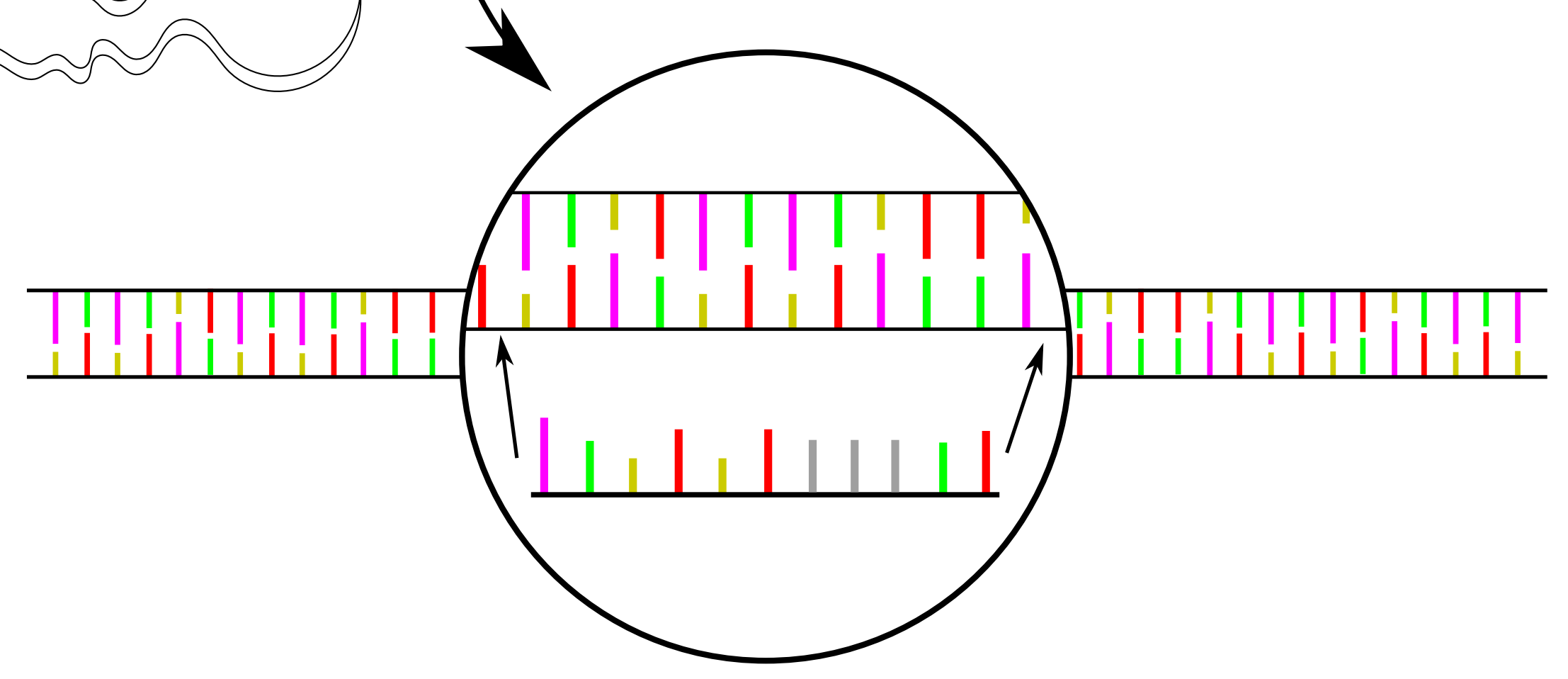


1. Mapping

The LSH algorithm maps all reads to candidate positions in the reference genome.

2. Alignment

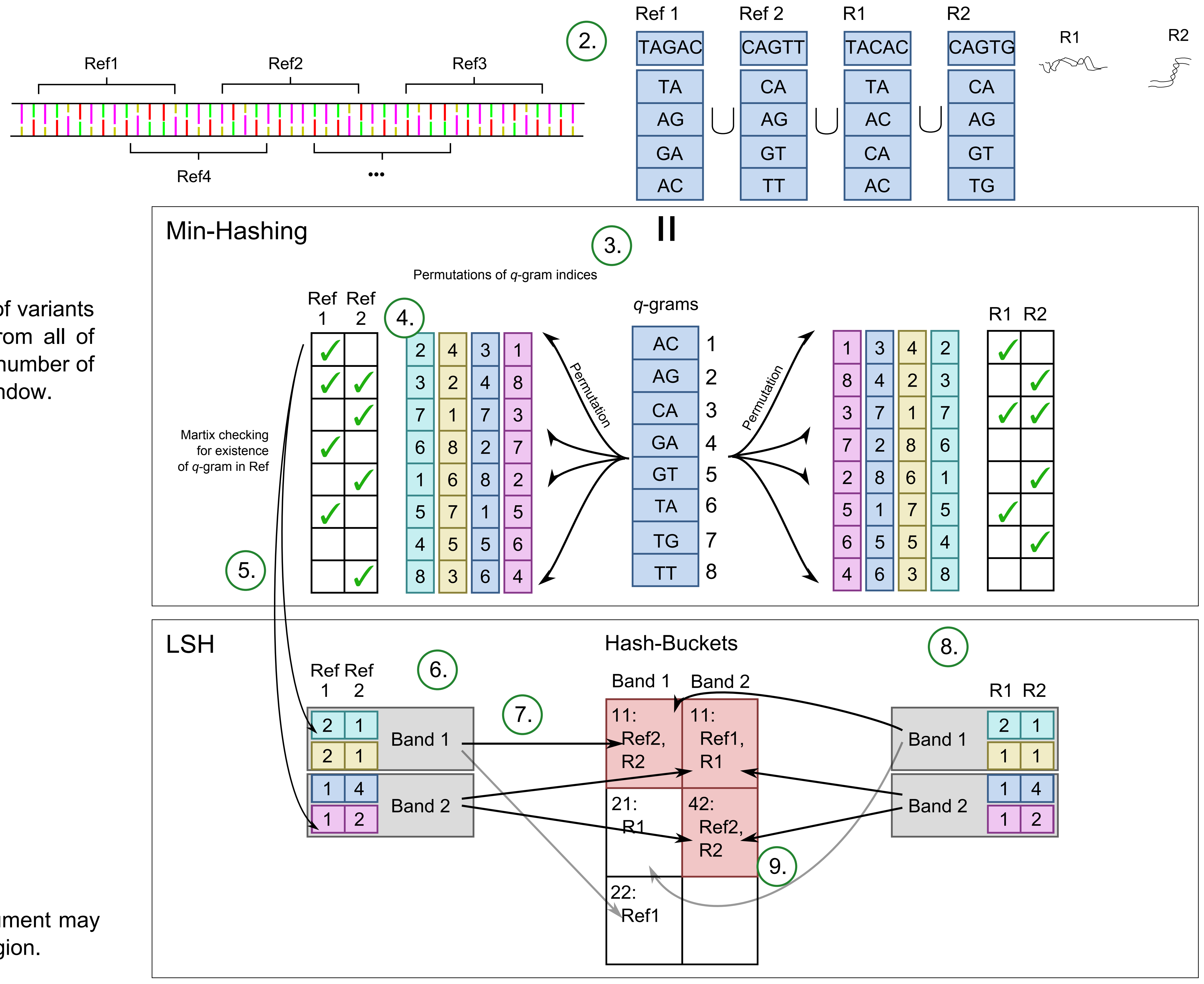
The algorithm for exact alignment is based on the semi-global pattern matching algorithm by Ukkonen.[3] Reduced data size leads to reasonable run time.



Mapping - LSH algorithm

Utilization for read mapping:

1. Split references into half-overlapping windows. These will provide our documents for locality-sensitive hashing.
2. Calculate set of *q*-grams for every document.
   In case of genomic variants, the algorithm computes all combinations of variants inside a window of *q* characters. The *q*-grams, which are derived from all of these combinations, are added for the corresponding document. If the number of combinations exceeds a certain limit, variants are ignored inside the window.
3. Calculate union of the *q*-gram sets.
4. Permute *q*-gram indices.
5. Save index of the first existing *q*-gram in the current document according to the generated permutation.
6. Create signatures by splitting the signature matrix into bands.
7. Signatures within a band represent the keys for the hash buckets.
8. Create signature matrix for reads analogously.
9. Reads whose signature collide with a signature from a reference document may be interpreted as putatively mapping to the corresponding reference region.



Semi-global Alignment

The semi-global alignment algorithm is based on dynamic programming.

Here is an example for a nucleotide string aligned against a read.

```
Alignment: AATAGACCT
              T GAC
```

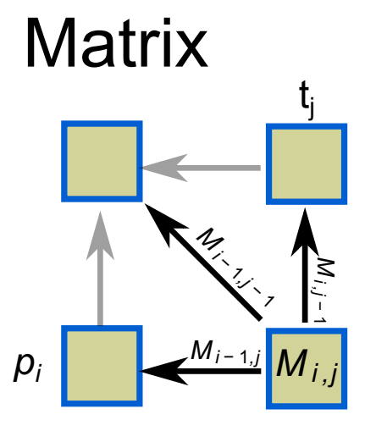
Our improvement to the algorithm is that it aligns a read against multiple variants of a reference string.

Variants

AGAACTAATG	AGAACTAATG
AGCACTAATG	SNP(2, "C")
AGAACTTG	Deletion(6, 2)
AGAACGCTAATG	Insertion(4, "CG")

Since the variants are stored as a difference to the reference, only one (modified) alignment process suffices.

Matrix



The matrix is calculated column by column. Each entry is calculated recursively with:

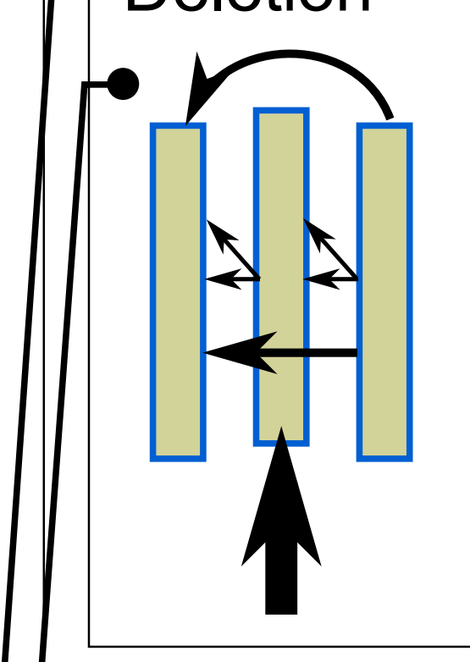
$$M_{i,j} = \min \begin{cases} M_{i-1,j-1} & + \operatorname{dist}(p_i, t_j) \\ M_{i-1,j} & + 1 \\ M_{i,j-1} & + 1 \end{cases}$$

SNP

A	C	G	T	
0	1	0	0	C
1	0	0	1	A/T
⋮				⋮

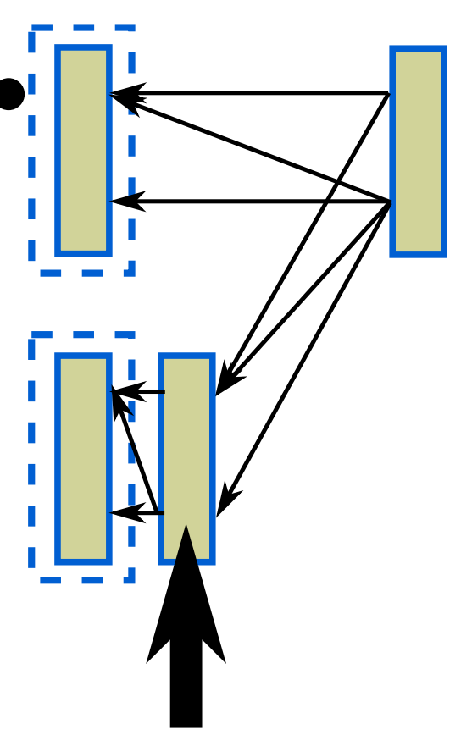
The algorithm uses the IUPAC alphabet for the reference string, which allows it to encode arbitrary SNPs within the reference sequence. The semantic of the IUPAC chars is considered when comparing reference bases with read bases.

Deletion



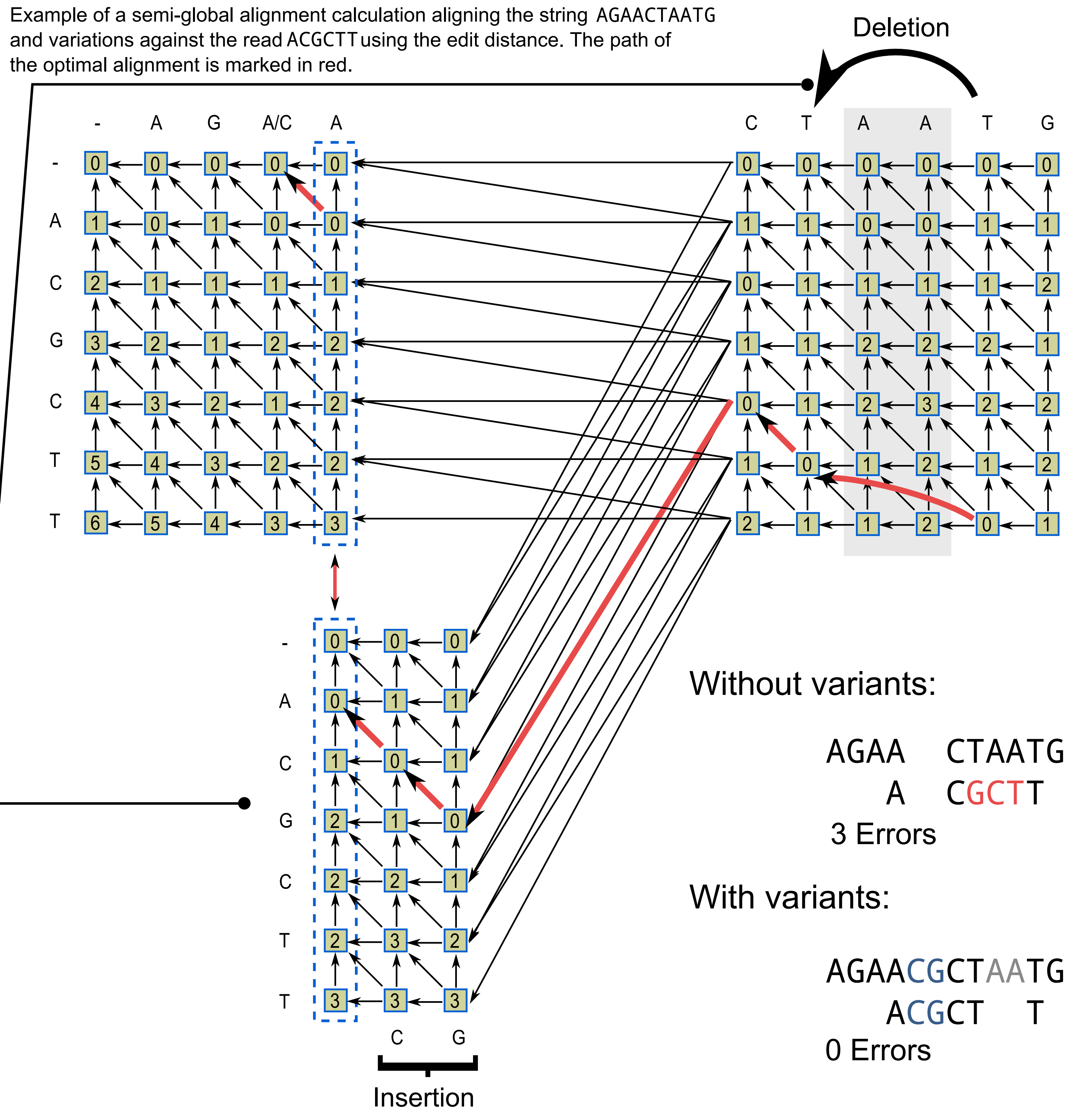
Deletion variants can be handled by skipping the affected columns for the computation. For each entry the algorithm decides, whether it uses the previous reference base or the reference base before the deletion variant. For the sake of simplicity some arrows are left out for this variant and only indicated by the large arrow above the matrix.

Insertion



For insertion variants the algorithm takes the last filled column before the insertion's position and speculatively continues the computation of the alignment on the insertion string. For the reference base after the insertion's position the algorithm decides, whether it uses the insertion string for optimization or just the reference string. The relevant predecessors of each entry are indicated by arrows.

Example of a semi-global alignment calculation aligning the string AGAACTAATG and variations against the read ACGCTT using the edit distance. The path of the optimal alignment is marked in red.



Without variants:

```
AGAA  CTAATG
   A  CGCTT
```

3 Errors

With variants:

```
AGAACGCTAATG
   ACGCT  T
```

0 Errors

References

[1] Alexandr Andoni and Piotr Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. In Foundations of Computer Science, 2006. FOCS'06. 47th Annual IEEE Symposiumon, pages 459-468.IEEE, 2006.

[2] Po-Ru Loh, Michael Baym, and Bonnie Berger. Compressive genomics. Nature biotechnology, 30(7):627-630, 2012.

[3] E. Ukkonen. Finding approximate patterns in strings. Journal of Algorithms,6(1):132 - 137, 1985.