GapMis-OMP: pairwise short-read alignment on multi-core architectures

Tomáš Flouri[1], Costas S. Iliopoulos[2,3], Kunsoo Park[4], and Solon P. Pissis[1,5] ⋆

[1] Heidelberg Institute for Theoretical Studies, Germany
[2] Dept. of Informatics, King's College London, UK
[3] Dept. of Mathematics & Statistics, University of Western Australia, Australia
[4] School of Computer Science and Engineering, Seoul National University, South Korea
[5] Florida Museum of Natural History, University of Florida, USA

Abstract. Pairwise sequence alignment has received a new motivation due to the advent of next-generation sequencing technologies, particularly so for the application of *re-sequencing*—the assembly of a genome directed by a reference sequence. After the fast alignment between a factor of the reference sequence and a high-quality fragment of a short read by a short-read alignment programme, an important problem is to find the alignment between a relatively short succeeding factor of the reference sequence and the remaining low-quality fragment of the read allowing a number of mismatches and the insertion of a single gap in the alignment. In this article, we present `GapMis-OMP`, a tool for pairwise short-read alignment that works on multi-core architectures. It is designed to compute the alignments between all the sequences in a first set of sequences and all those from a second one in parallel. The presented experimental results demonstrate that `GapMis-OMP` is more efficient than most popular tools.

Availability: `http://www.exelixis-lab.org/gapmis`

1 Introduction

A *gap* is a consecutive sequence of holes in an alignment. When applying alignment algorithms to biological sequence data, it is sometimes desirable to globally penalise the formation of long gaps instead of locally penalising the individual deletion or insertion of letters. A gap in a biological sequence can be regarded as the absence (resp. presence) of a region, which is (resp. is not) present in another sequence, because of the natural diversity among individuals. The gap concept in sequence alignments is therefore important in many biological applications because the

⋆ This work was supported in part by the NSF-funded iPlant Collaborative (NSF grant #DBI-0735191)

insertion or deletion of an entire region (particularly in DNA) often occurs as a single event. Many of these single mutational events can create gaps of varying sizes with almost equal likelihood—within a wide, but bounded, range of sizes.

Our work is motivated by the well-known and challenging problem of *re-sequencing*—the assembly of a genome that is directed by using a reference genome. Continuous advances in sequencing technology are turning whole-genome sequencing into a routine procedure, resulting in massive amounts of sequence data that need to be processed. Tens of gigabytes of data, in the form of short sequences (reads), need to be mapped (aligned) back to reference sequences, a few gigabases long, to infer the read from which the genomic location derived. This is a challenging task because of the high data volume and the large genome sizes. In addition, the performance, in terms of speed, sensitivity, and accuracy, deteriorates in the presence of inherent genomic variability and sequencing errors, particularly so, for relatively short consecutive sequences of deletions or insertions in the reference sequence or in the reads.

A broad variety of short-read alignment programmes, (e.g. Bowtie [1], SOAP2 [2], REAL [3]) has been released recently to address the task of mapping millions of short reads to a genome, placing emphasis on various aspects of the challenge (e.g. time and memory efficiency, sensitivity, and accuracy). Although all programmes allow for a small number of mismatches in the alignment, some of them either perform poorly when allowing the insertion of gaps or do not allow it at all.

The *seed-and-extend* strategy is applied in most current short-read alignment programmes [4]. After a fast alignment between a fragment of the reference sequence and a high-quality fragment of a short read (positions 1-3 in square brackets in Figure 1) by a short-read alignment programme, an important problem is to find the alignment between a relatively short succeeding fragment of the reference sequence and the remaining low-quality fragment of the read (positions 4-9 in Figure 1); allowing a number of mismatches (position 8 in Figure 1) and the insertion of a single gap (positions 5-6 in Figure 1) either in the fragment of the reference sequence or in the read.

One main observation from Figure 1 is that a gap might need to be inserted in the leftmost position of the alignment (position 4 in Figure 1), and that we are unable to know the exact length of the succeeding fragment of the reference sequence to be aligned a priori. Hence, we need an intermediate approach between the global (e.g. Needleman-Wunsch algorithm [5]) and the local alignment (e.g. Smith-Waterman algorithm [6]),

```
          [1 2 3] 4 5 6 7 8 9 10 11 12
Reference G C G A C G T C C G  A  A
          | | | |     | . |
          G C G A ⋆ ⋆ T A C
```

Fig. 1. Alignment between the fragment of the reference sequence, starting at position 1 and ending at position 9, and the read with one mismatch at position 8 and a single gap of length two inserted in the read after position 4

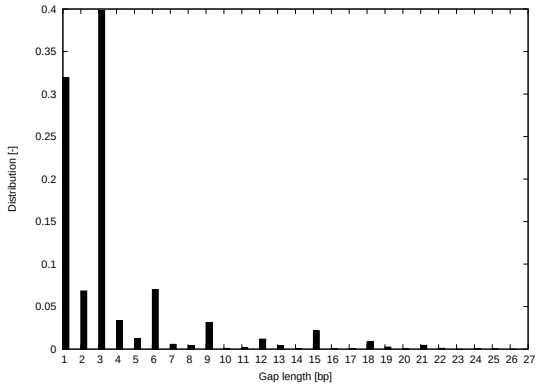

Fig. 2. Distribution of gap lengths in exome sequencing

known as *semi-global* alignment, that penalises the insertion of a gap in the leftmost position of the alignment, but ignores the insertion of a gap at the end.

Concerning the length of the gaps, a broad range of lengths is possible. In practice, however, the length of reads is too small to confidently and directly detect a large gap. In Figure 2, the distribution of lengths of gaps in exome sequencing is demonstrated[1]. The shape of the gap length distribution is consistent with other studies (cf. [7]). The presented data reflect a gap occurrence frequency of approximately 5.7×10^{-6} across the *exome*—the part of the genome formed by exons that codes portions of genes in the genome which are expressed.

The main observation from Figure 2 is the exponential decrease of gaps with increasing length, and a preference for lengths which are multiples of 3 base pairs (bp)—loss or gain of codons. For short reads in the range of 25-150 bp, the presence of multiple gaps is unlikely given the

[1] Data generated by the Exome Sequencing Programme at the NIHR Biomedical Research Centre at Guy's and St Thomas' NHS Foundation Trust in partnership with King's College London.

above gap occurrence frequency, and can greatly reduce the mapping confidence (accuracy) of those reads. Hence, applying a traditional dynamic programming approach, which allows for multiple mismatches, deletions, and/or insertions in the alignment, can deteriorate mapping confidence.

In this article, motivated by the aforementioned observations, we consider the problem of approximate string matching with k-mismatches and a single gap. We present `GapMis-OMP`, a tool for pairwise short-read alignment that works on multi-core architectures. It is based on algorithm GapMis, introduced in [8], for computing a modified version of the traditional dynamic programming matrix for sequence alignment to solve the above problem. `GapMis-OMP` is designed to compute the alignments between all the sequences in a first set of sequences and all those from a second one in parallel.

2 Definitions and notation

An *alphabet* Σ is a finite non-empty set whose elements are called *letters*. A *string* on an alphabet Σ is a finite, possibly empty, sequence of elements of Σ. The zero-letter sequence is called the *empty string*, and is denoted by ε. The *length* of a string x is defined as the length of the sequence associated with the string x, and is denoted by $|x|$. We denote by $x[i]$, for all $1 \leq i \leq |x|$, the letter at index i of x. Each index i, for all $1 \leq i \leq |x|$, is a position in x when $x \neq \varepsilon$. It follows that the ith letter of x is the letter at position i in x, and that

$$x = x[1\,..\,|x|]$$

A string x is a *factor* of a string y if there exist two strings u and v, such that $y = uxv$. Let x, y, u, and v be strings, such that $y = uxv$ holds. If $u = \varepsilon$, then x is a *prefix* of y. If $v = \varepsilon$, then x is a *suffix* of y.

Let x be a non-empty string and y be a string. We say that there exists an *occurrence* of x in y, or, more simply, that x *occurs in* y, when x is a factor of y. Every occurrence of x can be characterised by a position in y. Thus we say that x occurs at the *starting position* i in y when $y[i\,..\,i+|x|-1] = x$. It is sometimes more suitable to consider the *ending position* $i+|x|-1$. The *Hamming distance* δ_H for two strings of the same length, is defined as the number of positions where the two strings possess different letters.

A *don't care* letter is a special letter that does not belong to alphabet Σ, and matches with itself as well as with any letter of Σ. It is denoted by $\star$. A *gap* is a finite sequence of such *don't care* letters. A *gap string* is a

finite, possibly empty, sequence of elements of the alphabet $\Sigma \cup \{\star\}$. Two letters a and b of alphabet $\Sigma \cup \{\star\}$ are said to *correspond*, denoted by $a \approx b$, if they are equal, or, if at least one of them is the don't care letter. The *G-distance*, denoted by δ_G, for two gap strings of the same length is defined as the number of positions in which the two strings possess letters that do not correspond. A gap string x is called *single-gap string* if there exist two strings u and v on alphabet Σ and a gap g, such that $x = ugv$. Let $\mathsf{conc}(y')$ be an operation that, given a gap string

$$y' = y_0 g_0 y_1 g_1 \dots y_{n-2} g_{n-2} y_{n-1}$$

where $y_i \in \Sigma^*$, for all $0 \leq i < n$, and $g_j \in \{\star\}^*$, for all $0 \leq j < n-1$, returns the string $y = y_0 y_1 \dots y_{n-1}$, such that $y \in \Sigma^*$.

The *approximate string-matching with k-mismatches and a single gap* problem can now be formally defined as follows.

Problem 1. Given a text t of length n, a pattern x of length $m \leq n$, an integer k, such that $0 \leq k < m$, and integers α and β, such that $0 \leq \alpha \leq \beta$ and $\beta < n$, find all prefixes of t, such that for each prefix y

- either there exists a single-gap string y', with a gap g, such that $y = \mathsf{conc}(y')$, $\delta_G(x, y') \leq k$, and $\alpha \leq |g| \leq \beta$
- or there exists a single-gap string x', with a gap g, such that $x = \mathsf{conc}(x')$, $\delta_G(x', y) \leq k$, and $\alpha \leq |g| \leq \beta$
- or $\delta_H(x, y) \leq k$ and $\alpha = 0$

Example 1. Let $t = \mathtt{AGCAGAGGAGCAGGCGTTCCGTGGT}$, $x = \mathtt{ACCGT}$, $k = 2$, $\alpha = 6$, and $\beta = 7$. The solution to this problem instance is the set $\{11, 17, 22\}$ of ending positions. For example, the solution contains 11 since there exists a single-gap string $x' = \mathtt{ACC}\star\star\star\star\star\star\mathtt{GT}$, with a gap $g = \star\star\star\star\star\star$, such that $x = \mathsf{conc}(\mathtt{ACC}\star\star\star\star\star\star\mathtt{GT})$, $\delta_G(x', t[1\,..\,11]) = 2$, and $|g| = 6$.

Let $\mathsf{G}[0\,..\,n, 0\,..\,m]$ be a matrix, where $\mathsf{G}[i, j]$ contains the minimum number of mismatches of the alignment between factor $t[1\,..\,i]$ of t and factor $x[1\,..\,j]$ of x allowing the insertion of a single gap either in $t[1\,..\,i]$ or in $x[1\,..\,j]$, for all $1 \leq i \leq n$, $1 \leq j \leq m$. We say that $x[1\,..\,j]$ matches $t[1\,..\,i]$ with at most k-mismatches and a single gap *if and only if* $\mathsf{G}[i, j] \leq k$, for all $1 \leq j \leq m$, $1 \leq i \leq n$.

In order to compute the exact location of the inserted gap, either in the text or in the pattern, we also need to compute a matrix $\mathsf{H}[0\,..\,n, 0\,..\,m]$,

such that

$$\mathsf{H}[i,j] = \begin{cases} j-i & \text{if } \mathsf{G}[i,j] = \mathsf{G}[i,i] \text{ and } i \leq j \\ i-j & \text{if } \mathsf{G}[i,j] = \mathsf{G}[j,j] \text{ and } i > j \\ 0 & \text{otherwise} \end{cases}$$

Example 2. Let t = AGGTCAT, x = GGGTA, and $\beta = 2$. Table 1a and Table 1b illustrate matrix G and matrix H, respectively.

		0	1	2	3	4	5
		ϵ	G	G	G	T	A
0	ϵ	0	0	0			
1	A	0	1	1	1		
2	G	0	0	1	1	1	
3	G		0	0	1	1	1
4	T			1	1	1	1
5	C				1	1	2
6	A					1	1
7	T						2

(a) Matrix G

		0	1	2	3	4	5
		ϵ	G	G	G	T	A
0	ϵ	0	1	2			
1	A	1	0	0	0		
2	G	2	0	0	0	2	
3	G		0	0	0	1	2
4	T			0	0	0	1
5	C				2	1	0
6	A					2	0
7	T						0

(b) Matrix H

Table 1. Matrix G and matrix H for t = AGGTCAT, x = GGGTA, and $\beta = 2$

3 Algorithm GapMis

Algorithm GapMis, introduced in [8], computes matrices G and H. It takes as input the text t of length n, the pattern x of length m, and the threshold β.

Proposition 1 ([8]). *There exist at most $2\beta + 1$ cells of matrix G that yield a solution for Problem 1.*

Proposition 2 ([8]). *Problem 1 can be solved by algorithm* GapMis *in time $\mathcal{O}(m\beta)$.*

As soon as we have computed matrices G and H, we can apply a simple *alignment scoring* scheme, depending on the application of the algorithm, to compute the maximum score among all possible alignments of t and x in time $\Theta(\beta)$ by Proposition 1.

ALGORITHM GapMis(t, n, x, m, β)

```
    {Initialise matrices G and H }
 1: for i ← 0 to n do
 2:     G[i, 0] ← 0;
 3:     H[i, 0] ← i;
 4: for j ← 0 to m do
 5:     G[0, j] ← 0;
 6:     H[0, j] ← j;
    {Computing matrices G and H}
 7: for i ← 1 to min{n, m + β} do
 8:     for j ← max{1, i − β} to min{m, i + β} do
 9:         if i < j then
10:             u ← G[i − 1, j − 1] + δ_H(t[i], x[j]);
11:             v ← G[i, i];
12:             G[i, j] ← min{u, v};
13:             if v < u then
14:                 H[i, j] ← j − i;
15:             else
16:                 H[i, j] ← 0;
17:         if i > j then
18:             u ← G[i − 1, j − 1] + δ_H(t[i], x[j]);
19:             v ← G[j, j];
20:             G[i, j] ← min{u, v};
21:             if v < u then
22:                 H[i, j] ← i − j;
23:             else
24:                 H[i, j] ← 0;
25:         if i = j then
26:             G[i, j] ← G[i − 1, j − 1] + δ_H(t[i], x[j]);
27:             H[i, j] ← 0;
28: return G and H;
```

4 Implementation

GapMis-OMP was implemented in the C programming language, and was developed under the GNU/Linux operating system. We used the Open Multi-Processing (OpenMP) application programming interface that supports multi-platform shared-memory multiprocessing programming.

We implemented algorithm GapMis as a function to compute the optimal semi-global alignment between two sequences with a single gap. We applied a simple alignment score scheme for DNA (resp. protein) sequences, that uses the scoring matrix EDNAFULL (resp. EBLOSUM62) [9] to assign scores for every possible nucleotide (resp. residue) match or mismatch; and affine gap penalty to score the insertion of a single gap. The

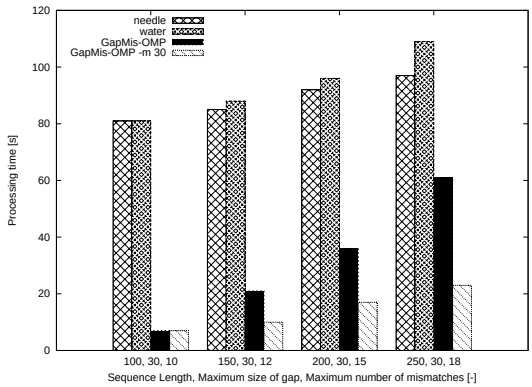

Fig. 3. Processing times of `needle`, `water`, and `GapMis-OMP` for aligning 10,000 pairs of sequences

penalty for a gap of $n > 0$ positions is computed as

$$gap\ opening\ penalty + (n-1) * gap\ extension\ penalty$$

The total score of each alignment is obtained by adding these two scores, and the optimal alignment is the one with the maximum total score. The same alignment score scheme is applied in package EMBOSS [10].

The programme takes as input arguments two files with the two sets of sequences (DNA or protein) in FASTA format. It consists of an outer for loop, going through the sequences of the second set (target sequences), and an inner for loop, going through the sequences of the first set (query sequences), to compute the optimal semi-global alignment for each pair. We used the `omp parallel for` directive to effectively define the inner for loop as a parallel region, as there exists no dependency between those tasks (pairs). The number of threads to be used can be set by the user. The programme produces an EMBOSS-like text file with all the optimal semi-global alignments as output.

5 Experimental results

To the best of our knowledge, `GapMis-OMP` is the first tool for pairwise short-read alignment between two sets of sequences. In [11], in order to highlight the suitability of algorithm GapMis, we compared it against EMBOSS `needle`, which implements the Needleman-Wunsch algorithm for global alignment, and EMBOSS `water`, which implements the

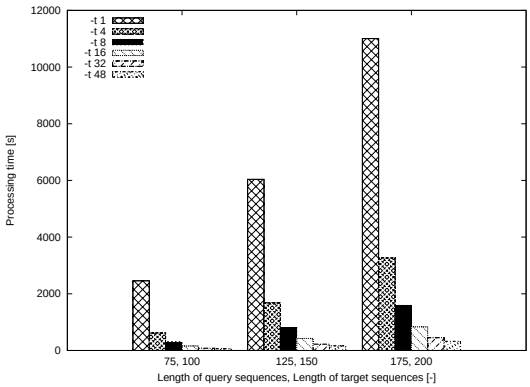

Fig. 4. Processing times of `GapMis-OMP` for aligning 100,000 query sequences and 100 target sequences

Smith-Waterman algorithm for local alignment. At present, these two programmes are two of the most widely used pairwise sequence alignment programmes. We demonstrated that `needle` and `water` can, by design, not guarantee the insertion of at most one gap in the alignment.

In this article, in order to evaluate the efficiency of `GapMis-OMP`, we compared its performance to the respective performance of `needle` and `water`. We simulated 10,000 pairs of 100 bp-long DNA sequences, such that each pair consists of two identical sequences; and then we inserted, in one of the two sequences, a single gap of random length ranging from 1 to 30 and a random number of mismatches ranging from 1 to 10. We repeated the same experiment by simulating 150, 200, and 250 bp-long sequences. Two versions of `GapMis-OMP` were used; one with the modifier `-m 30` to set $\beta = 30$ and use the $\Theta(m\beta)$ algorithm, and one without it to use the $\Theta(mn)$ algorithm.

As it is demonstrated by the results in Figure 3 `GapMis-OMP` was able to complete the assignment much faster than `needle` and `water`. The version with the modifier `-m 30` was always the fastest confirming our theoretical results.

In order to evaluate the parallel efficiency of `GapMis-OMP`, we simulated a set of 100,000 75 bp-long query sequences and 100 100 bp-long target sequences. Six versions of `GapMis-OMP` were used: with 1 thread (`-t 1`); with 4 threads (`-t 4`); with 8 threads (`-t 8`); with 16 threads (`-t 16`); with 32 threads (`-t 32`); and with 48 threads (`-t 48`). We repeated the same experiment with 150 and 200 bp-long target sequences. As depicted in Figure 4, the `-t 4` version is able to complete the assignment

up to 4× faster, the `-t 8` version up to 7.8× faster, the `-t 16` version up to 15.5× faster, the `-t 32` version up to 30.6× faster, and the `-t 48` version up to 41.7× faster than the `-t 1` version.

The experiments were conducted on 1 node of a cluster architecture using 1 to 48 2.4 GHz AMD 6136 processors and 125 GB of main memory, and running the GNU/Linux operating system. `GapMis-OMP` is distributed under the GNU General Public License (GPL).

References

1. Ben Langmead, Cole Trapnell, Mihai Pop, and Steven L. Salzberg. Ultrafast and memory-efficient alignment of short DNA sequences to the human genome. *Genome biology*, 10(3):R25+, 2009.
2. Ruiqiang Li, Chang Yu, Yingrui Li, Tak-Wah Lam, Siu-Ming Yiu, Karsten Kristiansen, and Jun Wang. SOAP2: an improved ultrafast tool for short read alignment. *Bioinformatics*, 25(16):1966–1967, 2009.
3. Kimon Frousios, Costas S. Iliopoulos, Laurent Mouchard, Solon P. Pissis, and German Tischler. REAL: an efficient REad ALigner for next generation sequencing reads. In Aidong Zhang, Mark Borodovsky, Gultekin Özsoyoglu, and Armin R. Mikler, editors, *Proceedings of the first ACM International Conference on Bioinformatics and Computational Biology (BCB 2011)*, pages 154–159, USA, 2010. ACM.
4. Stephen Altschul, Warren Gish, Webb Miller, Eugene Myers, and David J. Lipman. Basic Local Alignment Search Tool. *Journal of Molecular Biology*, 215(3):403–410, 1990.
5. Saul B. Needleman and Christian D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*, 48(3):443–453, 1970.
6. Temple F. Smith and Michael S. Waterman. Identification of common molecular subsequences. *Journal of Molecular Biology*, 147(1):195–197, 1981.
7. Sarah B. Ng, Emily H. Turner, Peggy D. Robertson, Steven D. Flygare, Abigail W. Bigham, Choli Lee, Tristan Shaffer, Michelle Wong, Arindam Bhattacharjee, Evan E. Eichler, Michael Bamshad, Deborah A. Nickerson, and Jay Shendure. Targeted capture and massively parallel sequencing of 12 human exomes. *Nature*, 461(7261):272–276, 2009.
8. Tomáš Flouri, Kimon Frousios, Costas S. Iliopoulos, Kunsoo Park, Solon P. Pissis, and German Tischler. Approximate string-matching with a single gap for sequence alignment. In ACM, editor, *Proceedings of the second ACM International Conference on Bioinformatics and Computational Biology (BCB 2011)*, pages 490–492, USA, 2011. ACM.
9. National Center for Biotechnology Information (NCBI). `ftp://ftp.ncbi.nih.gov/blast/matrices/`, June 2012.
10. Peter Rice, Ian Longden, and Alan Bleasby. EMBOSS: The European Molecular Biology Open Software Suite. *Trends in Genetics*, 16(6):276–277, 2000.
11. Tomáš Flouri, Kimon Frousios, Costas S. Iliopoulos, Kunsoo Park, Solon P. Pissis, and German Tischler. GapMis: a tool for pairwise sequence alignment with a single gap. *Recent Patents on DNA and Gene Sequence*, 2012. (accepted).