The implementation of bit-parallelism for DNA sequence alignment

Setyorini*, Kuspriyanto, D H Widyantoro and A Pancoro

School of Electrical & Informatic Engineering, Bandung Institute of Technology, Bandung, Indonesia, 4Schools of Life Science, TB Bandung Indonesia

E-mail:setyorini@students.itb.ac.id

Abstract. Dynamic Programming (DP) remain the central algorithm of biological sequence alignment. Matching score computation is the most time-consuming process. Bit-parallelism is one of approximate string matching techniques that transform DP matrix cell unit processing into word unit (groups of cell). Bit-parallelism compute the scores column-wise. Adopting from word processing in computer system work, this technique promise reducing time in score computing process in DP matrix. In this paper, we implement bit-parallelism technique for DNA sequence alignment. Our bit-parallelism implementation have less time for score computational process but still need improvement for the reconstruction process.

1. Introduction

Alignment is the important first step toward structural and functional analysis of newly determined sequences. Many new biological sequences are being generated and already existing in the databases. Functional and evolutionary inference of new biological sequence can be drawn by sequence comparison from that database. Sequence alignment does accomplish by searching for common character pattern and making residue-residue correspondence among related sequences. Pairwise sequence alignment is the process of making alignment between 2 protein sequences [1].

As a computational problem, sequence comparison is defined as approximate string matching problem. Approximate string matching is a technique finding pattern of string that match approximately. There are limited errors that allowed during matching process. There are several techniques to implement approximate string matching: Dynamic Programming (DP) [2], Automaton [3], Filtering [4] and Bit-Parallelism.

DP is one of the classical algorithms to solve approximate string matching problem. Classical DP computation process increases exponentially along with the increasing of sequence size. Several algorithms proposed in an effort to make the computational process more efficient, without sacrificing the accuracy [5]. Bit-parallelism is the one of that improvement, that parallelize the proses of score computation in DP matrix.

The authors implement bit-parallelism for aligning the DNA sequence, which is originally proposed by Myer [6]. Myer proposed bit-parallelism algorithm to speed-up DP score computation by column-wise computation. Alignment result obtained by reconstruction algorithm proposed by Hyrro [7] slightly modified to accelerate the computing process.

* Corresponding author
This paper is organized as follows: in Section 2 we describe sequence alignment as approximate string matching problem, that bit-parallelism DP matrix as one approach. Explanation of bit-parallelism and alignment construction mechanism will be obtained in Section 3. We present implementation and result discussion in Section 4, and then the conclusion in Section 5.

2. Sequence Alignment as a Problem of Approximate String Matching

Approximate string matching is defined formally by [5] as searching short pattern $P$ of length $m$, in a long text $T$ of length $n$, with up to $k$ eror. The output is all text position $j$, such that suffix $T ... j$ matches $P$ with at most $k$ errors. Error defined as the minimum cost of sequence operations to transform pattern $P$ to text $T$. The cost of sequence operations is sum of individual operations. The sequence operations are an insertion, deletion, and substitution. The error also is known as a distance function. Commonly used distance functions are: Levenshtein/ Edit Distance, Hamming Distance, Episode Distance and Longest Common of subsequences Distance.

Sequence alignment consists of two tasks: searching the best score of a common pattern of sequence residues and constructing residue correspondence among them. The common pattern is selected by similarity level. This needs a scoring mechanism to determine the quality of similarity. Scoring scheme defines usage rules of edit operations into three assessment criteria, the score for a match, mismatch, and insertion. Gap character is used to represent insertion and deletion. Inserting and removing naturally occur in evolution. As approximate string matching problem sequence alignment defined as searching optimal total score of edit operation from all possible sequence residues comparisons.

DP algorithm computed the edit distance. This algorithm is flexible to handle different distance function, but it is not very efficient. DP matrix increases exponentially as the sequence size increases. Comparing more than 2 sequences is feasible, but not reliable on its computation.

3. Sequence Alignment using Bit-parallelism

3.1. DP Score Computation

Some Dynamic Programming (DP) computation for solving pairwise sequence alignment problem will generate a matching score matrix. Score each cell matrix is decided by it adjacency. Each cell has 3 adjacency, upper side, diagonal side and left side.

Given seq$S_1$ (length $m$) and seq$S_2$(length $n$), DP algorithm will generate a matrix $C(m + 1, n + 1)$. The matrix initializes, and compute the score with this the well-known recurrence:

Recurrence 1.

$$
C_{0,j} = j
$$

$$
C_{i,j} = \begin{cases} 
S_1_i = S_2_j \text{ then } C_{i-1,j-1} \\
\text{else} \min(C_{i-1,j}, C_{i-1,j-1}, C_{i,j-1})
\end{cases}
$$

Each score represents the cumulative distance until the i-th character of $S_1$ and j-th character of $S_2$. There is illustration for DP matrix computation:
Initial cell matrix

|   |   |   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| A | 1 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| N | 2 | 1 | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
| N | 3 | 2 | 1 | 0 | 1 | 2 | 3 | 4 | 5 |
| U | 4 | 3 | 2 | 1 | 1 | 2 | 3 | 4 | 5 |
| A | 5 | 4 | 3 | 2 | 1 | 2 | 3 | 4 | 5 |
| L | 6 | 5 | 4 | 3 | 3 | 2 | 1 | 2 | 3 |

Figure 1. Example of dynamic programming matrix computation.

From matrix above, (Figure 1) the last line matrix can be seen as the final score. Scores for optimal full/global alignment is on the bottom right corner. Score represent the length of distance or differences between the seqS₁ and seqS₂. In this case, the best score is 1, because it does not belong to bottom right corner, it generates local alignment. All alignment result can be generated by tracing back from each last line matrix score.

The example of full alignment results shown by Figure 2. Alignment result gives 5 matches, a mismatch, and 3 gap insertion. Matches/mismatches condition occur when the arrow of trace-back goes diagonally, the gap for string “ANNUAL” when the arrow goes vertically and gap for string “ANNEALING” when the arrow goes horizontally.

Figure 2. Example of alignment reconstruction from Dynamic Programming Matrix.

The idea bit-parallelism aims to save computing operations by converting the computing unit matrix cells from each cell into a collection of cells. Set of cells treated as new data unit such as the concept of word in the computer system. With this concept, the computing score DP replaced by computing the DP matrix cell adjacency. This difference value is then formed into a boolean value vector.

The adjacency of C(i,j) cell matrix is shown in Figure 3.
There are 3 different kinds of adjacency, vertically, horizontally and diagonally Figure 3. With Recurrence 1 the range of the adjacency values are represented as follows:

\[
\begin{align*}
\Delta h_{i,j} &= C_{i,j} - C_{i,j-1} \in \{-1,0,+1\} \\
\Delta v_{i,j} &= C_{i,j} - C_{i-1,j} \in \{-1,0,+1\} \\
\Delta d_{i,j} &= C_{i,j} - C_{i-1,j-1} \in \{0,+1\}
\end{align*}
\]

Two vectors required representing three possibility values in horizontal/vertical difference into boolean form. So the vectors formed as follows:

- **VP** vector for \( \Delta V_{1,m,j} = +1 \)
  - VP\(\_\) vector for \( \Delta V_{1,m,j} = -1 \)
- **HP** vector for \( \Delta H_{1,m,j} = +1 \)
  - HP\(\_\) vector for \( \Delta H_{1,m,j} = -1 \)
- **DO** vector for \( \Delta V_{1,m,j} = +1 \)

The next vector can be operated with the concept of bits. The bit operator involve are “&” for and, operator “+” for add operator, “|” for or operator, and “^” for xor operator.

**Recurrence 2**:

\[
\begin{align*}
D_0 & \leftarrow \left( ((P M_j \& V P_{j-1}) + V P_{j-1}) \& V P_{j-1}) \& V N_{j-1} \right) \\
H P_j & \leftarrow V N_{j-1} (D_0 \mid V P_{j-1}) \\
H N_j & \leftarrow D_0 \mid V P_{j-1} \\
\text{if } & H P_j \& 10^{m-1} \neq 0 \text{ then } D[m,j] \leftarrow D[m,j] + 1 \\
\text{if } & H N_j \& 10^{m-1} \neq 0 \text{ then } D[m,j] \leftarrow D[m,j] - 1 \\
V P_j & \leftarrow (H N_j \ll 1) \mid (D_0 \mid ((H P_j \ll 1) \ll 1)) \\
V N_j & \leftarrow D_0 \& ((H P_j \ll 1) \ll 1)
\end{align*}
\]

In Figure 4 is the example of generating the 5 vectors. A vector can be represented as an array of boolean values or as one/more integers values.
3.2. Alignment Reconstruction

Alignment reconstruction is about recovering alignment result from bit-parallelism vectors. Alignment reconstruction of DP matrix produced by running backtracking procedure. Neighboring cell value is checked to know from where the value is obtained. Recovering simple edit distance for LCS proposed by Iliopoulos and Pinzon [8], which permits a single operation to insert or delete a character.

Algorithm that handling more edit operation, substitution of character, is proposed by Heikki Hyyro [7]. This algorithm using vertical delta vector VN and VP (Figure 5). As backtracking in classical DP, this algorithm also start from the most right bottom. First, it checks if VP set, means the value come from the upper side, alignment result will be inserted gap for the pattern. But, if the VP is not set then it check VN. VN set means the value must become from the left side.

```
Alignment Reconstruction (DO, VP, VN)
  i ← m, j ← n
  while i > 0 and j > 0 Do
    if VP(i,j) == 1 then
      output (S1(i), '-')
      i ← i - 1
    else
      if VN(i,j-1) == 1 then
        output ('-', S2(j))
      else
        output (S1(i), S2(j))
        i ← i - 1
        j ← j - 1
  while i > 0 do
    output (S1(i), '-'); i ← i - 1
  while j > 0 do
    output ('-', S2(j)); j ← j - 1
```

**Figure 4.** The example of constructing D0, HP, HN, VP and VN vectors.

**Figure 5.** Hyyro algorithm for constructing alignment using vector VP & VN [7]
Using this algorithm we made a reconstruction matrix, during each column score computation processes. Next, is the principal to:

**Table 1. Truth table of reconstruction matrix.**

| VPj | VNj-1 | source cell |
|-----|-------|-------------|
| 1   | φ     | 01          |
| 0   | 1     | 10          |
| 0   | 0     | 00          |

Use Table 1, truth table will produce output value for reconstruction matrix. There are three possibilities, 00 if the value comes from a diagonal direction (match/mismatch representation), 10 if come from above direction, and 11 if come from the left side (gap insertion). Thus, can be generated output function that produces output in relational logic from as follows:

\[
\text{Construction Matrix} = \text{not}(VP_j)VN_{j-1} \cdot 2^1 + VP \cdot 2^0 \tag{4}
\]

Reconstructions algorithm have to maintain all iterative update of matrix VP and VN.

4. **Sequence Alignment using Bit-parallelism**

This paper applies the algorithms bit-parallelism and classical DP for computing the edit distance and alignment reconstruction. Alignment of the global alignment is intended. The system will require input as a pair of DNA Sequence in several sizes. A pair of sequence alignment is made in the same size. The DNA Sequence only contains 4 characters, as {‘A’, ‘T’, ‘G’, ‘C’}.

In referring from all the pair sequence groups in the same size, Sequence alignment test result will be calculating from average of test parameters as follows: (1) identical pairwise residue, i.e. the percentage alignment results with the same character pairs, (2) pairwise mismatch residue, can be % of mismatching residue alignment and (3) the percentage of large number gap insertion. All test parameters are measured relative to initial sequence size.

Alignment computational time is measured from two main processes, namely the computational score and reconstruction alignment results based on matrix score. Computing time score is the time to calculate all possible distance.

Datasets used in this study refer to data sets on the Loving research [9], which consists of 200 sequences with the length of each sequence is 100 residues. The data is then processed for generating some data size as follows:

**Table 2. DNA data set.**

| sequence size | # of sequence |
|---------------|--------------|
| 10            | 2000         |
| 20            | 1000         |
| 25            | 800          |
| 50            | 400          |

For Classic DP algorithm implementations adopts Needleman Wunch[10], with modifications to the weighting used, 0 for the conditions of the match and 1 for mismatch conditions and gap insertion. The weights used are the same with weights on the bit-parallelism algorithms Hyrro. Reconstruction algorithm adopted from the previous explanations on the Hyrro algorithm[7].

Alignment results are presented in the form of 3 string rows. The first and third line is a sequence of characters/residue from the sequence or the notation gap. The second line is the status of the correspondence between the residues. The symbol ‘|’ to represent the match condition, the symbol ‘:’ for a mismatch, and the notation ‘-’ to represent the gap insertion exist.
Figure 6 the following show average value for identical residue-pair percentage, and gap insertion percentage.

![Alignment Result: % Identical Residu-Pair & % Insert-Gap](image)

**Figure 6.** Sequence Alignment Result: average of percentage of match found & gap insertion.

Bit-parallelism got the similar result in the percentage of identical residu-pair. This result show that the algorithm still maintain the accuracy.

The best percentage identical residu-pair for sequence size 50 is 62% (Figure 7). However the algorithm still insert more gap than classical DP. Most of them are consecutives gaps (Figure 8) that occurred at the beginning and the end of sequence.

**Figure 7.** Best sequence alignment result for sequence size = 50

Consecutives gap can be handled with the gap penalty technique, such as affine gap that combine contant and linear gap penalty. It uses 2 different weight for opening gap and extention gap. Bit-parallelism have no flexibility different weights is not directly applicable. It need different scoring technique. Loving [9] proposed general integer scoring for sequence alignment. Loving algorithm assume a integer score for match, mismatch and gap weight.
The different weights (Figure 9) also have impact on accuracy as the next case in DP technique:

```
match = 0, mismatch, gap = 1
TTGGTAGGA-G-A-T-GGTAC-TGGAGTTGTTTTCTC-CGC-GATATCAAGATTGTAC
---CACCCAGACAGCCGCGCAGTATGCTT-TTGGGACTCGGAGCTC-TCT-GGCCGA-
identik = 8
match = 0, mismatch = 1, gap = 2
TTGGTAGGAGA-TGGTACTGGAGTTTCCG-CGATACAGCATTGTAC
-CAACCAGACAGCCAGCGCTAGCTGTTTTGGGACTCGGAGCTC-TCT-GGCCGA
identik = 18
match = 0, mismatch = 1, gap = 5
TTGGTAGGAGA-TGGTACTGGAGTTTCCG-CGATACAGCATTGTAC
CAACCAGACAGCCAGCGCTAGCTGTTTTGGGACTCGGAGCTC-TCT-GGCCGA
identik = 12
```

**Figure 9.** Cases when different weights give impact on the sequence alignment accuracy

High accuracy occured when the composition of match, mismatch and gap weights give the minimum distance to the sequences.

**Figure 10.** Sequence Alignment computation time

For computational time in Figure 10, show that bit-parallelism algorithm outperform at score computation phase, but still underperform at reconstruction phase. In our implementation need, reconstruction process consist of 2 phase, filling the direction matrix for backtracking and mapping the result. Matrix filling phase use the formula (1) did not consume much time, but not for iterative process in mapping phase.

5. Conclusion
The implementation of DNA sequence alignment using Bit-Parallelism technique produced similar accuracy as DP technique. Bit-Parallelism also increase the efficiency in score computation phase, however reconstruction phase still need to be improved. Implementation on more flexible weights
needed as future work to improved the accuracy and do more complex sequence alignment problem such on protein. This also open the possibility to implement the computation on hardware level.

References

[1] Rosenberg M S 2009 Sequence alignment: methods, models, concepts, and strategies (California: Univ of California Press)
[2] Xiong J 2006 Essential bioinformatics (Cambridge: Cambridge University Press)
[3] Wu S dan Manber U 1992 Commun. ACM 35 83-91
[4] Wright A H 1994 Softw. Pract. Exp. 24 337-362
[5] Ukkonen E 1992 Theor. Comput. Sci. 92 191-211
[6] Smith T F and Waterman M S 1981 Adv. Appl. Math. 2 482-489
[7] Wright A H 1994 Softw. Pract. Exp. 24 337-362
[8] Sellers P H 1974 SIAM J. Appl. Math. 26 787-793
[9] Needleman S B and Wunsch C D 1970 J. Mol. Biol. 48 443-453
[10] Navarro G 2001 ACM Comput. Surv. (CSUR) 33 31-88
[11] Myers G 1999 Journal of the ACM (JACM) 46 395-415
[12] Masek E J and Paterson M S 1980 J. Mol. Evol. 20 18-31
[13] Smith T F, Waterman M S and Fitch W M 1981 J. Mol. Evol. 18 38-46
[14] Landau G and Vishkin U 1989 J. Algorithm. 10 157-169
[15] Iliopoulos C S and Pinzon Y J 2001 Proc of the 12th Australasian Workshop On Combinatorial Algorithms (Australia)
[16] Iliopoulos C S and Pinzon Y J 2001 Proc of the 12th Australasian Workshop On Combinatorial Algorithms (Australia)
[17] Hyyrø H, Fredriksson K dan Navarro G 2005 JEA 10 2-6
[18] Hyyrø H 2001 Explaining and extending the bit-parallel approximate string matching algorithm of Myers (Technical Report A-2001-10) (Tampere: University of Tampere)
[19] Hyyrø H dan Navarro G 2005 Algorithmica 41 203-231
[20] Hyyrø H 2004 Proc. 15th Australasian Workshop on Combinatorial Algorithms (Australia)
[21] Hyyrø H 2004 Proc. Prague Stringology Conference '04 (Prague : Vydavatelství CVUT) pp.79-87
[22] Hyyrø H 2003 Nord. J. Comput., 10 29-39
[23] Gotoh O 1982 J. Mol. Biol. 162 705-708
[24] Benson G, Hernandez Y dan Loving J 2014 Bioinformatics 30 3166-3173
[25] Benson G, Hernandez Y dan Loving J 2014 Bioinformatics 30 3166-3173
[26] Baiza Y R and Navarro G 1999 Algorithmica 23 127-158.
[27] Baiza Y R, Barbosa E F dan Ziviani N 1993 XIII Congress of the Brazilian Computer Society, (Florianopolis: Anais do SBC’93) pp 11-24