SIMD Bit-Parallel Tandem Alignment

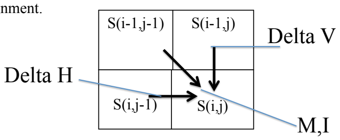
John Paul Scaduto[1,2], Josh Loving[1,3], Gary Benson[1,3]
[1]Boston University BRITE Bioinformatics REU Program Summer 2016; [2]Boston College;
[3]Boston University Bioinformatics Graduate Program

Abstract

DNA sequence analysis to determine variable number of tandem repeat (VNTR) genetic variation can give us information on the function of specific regions of DNA based on VNTR copy number and conservation, including insight into gene regulation, disease mechanisms and forensics. The need to quickly analyze high volume sequencing data motivated the search for a fast, efficient sequence alignment algorithm for tandem repeats. Tandem alignment can be accomplished using a wraparound dynamic programming method that aligns a single copy of the pattern to an unknown number of tandem copies in a second sequence. The goal of this project was to increase the efficiency of tandem alignment by applying bit-parallel techniques and using Single Instruction, Multiple Data (SIMD) computer instructions implemented in C. The bit-parallel approach was previously shown to be faster than iterative alignment. Our SIMD wraparound tandem alignment method also repeatedly loops through a single pattern copy instead of using multiple pattern copies. The SIMD instructions perform multiple operations in a single time step. The SIMD data types and intrinsics store data in 128 bit computer words split into 16 sections of 8-bits, or 1 byte. Operations are performed on all of the bytes simultaneously. A simple scoring method was used to assign integer weights and calculate a score based on the number of matches, mismatches, and insertion/deletions between the sequences to yield a final alignment score for the alignment. Benchmarking was performed on 250,000 DNA sequence pairs, each pair consisting of a pattern (chosen from lengths 15, 30, 60, 120, 240 base pairs) and a sequence 960 base pairs long. Our SIMD tandem alignment runtime averaged over 3 times faster than standard wraparound dynamic programming.

Background

Sequencing data is becoming more widely available as the cost of genome sequencing is becoming less expensive and the time it takes to sequence a genome becomes faster. As this sequencing data becomes available, it is useful to analyze and compare it using sequencing alignment.

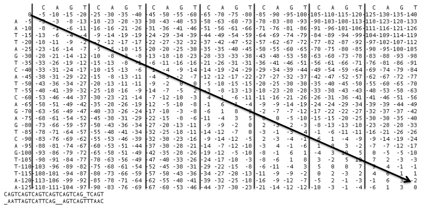


- **Delta H**= Horizontal change in score
- **Delta V**= Vertical change in score
- **M,I**= Match or mismatch value

$$S(i,j) = \max\left\{ \begin{array}{ll} s(i-1,j-1) & + M(match) \text{ or } I(mismatch), \\ s(i-1,j) & + G(gap), \\ s(i,j-1) & + G(gap) \end{array} \right\}$$

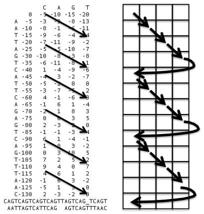
Sequencing Alignment uses a scoring matrix to compute a similarity between 2 sequences, or a pattern and a sequence. The scoring matrix looks at each position S(i,j) which holds the current score in a given position based on the cells above, to the left, and diagonally from the top-left. A gap value is used if it comes from above or the left, and a match or mismatch value is used if it comes diagonally. The match and mismatch value are dependent on whether the current letter of the 2 sequences being compared are the same Above is an example of a four cell demonstration of how the scoring matrix works.

Multiple copy global alignment



The multiple pattern copy alignment method of the tandem repeat uses multiple copies of the pattern duplicated along the top and compares it to the other DNA sequence as the pattern in view moves from one copy of the pattern to the next.

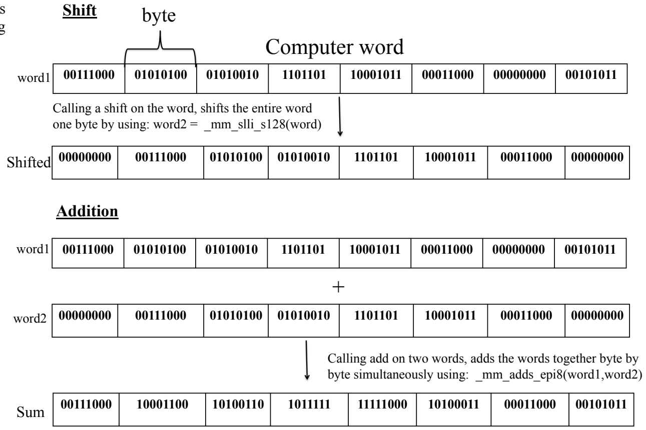
Single pattern copy



A single copy of the same pattern is used and when the score in use reaches the end of a pattern copy, it wraps around to return to the beginning of the string and carries with it the score from the ending of the previous row as it wraps. This yields the same score and trace back of the alignment using less information than the global alignment.

Methods

Single Instruction Multiple Data



Shift

word1	00111000	01010100	01010010	1101101	10001011	00011000	00000000	00101011

Calling a shift on the word, shifts the entire word one byte by using: word2 = _mm_slli_s128(word)

Shifted	00000000	00111000	01010100	01010010	1101101	10001011	00011000	00000000

Addition

word1	00111000	01010100	01010010	1101101	10001011	00011000	00000000	00101011
+								
word2	00000000	00111000	01010100	01010010	1101101	10001011	00011000	00000000

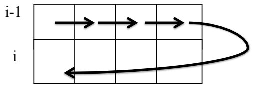
Calling add on two words, adds the words together byte by byte simultaneously using: _mm_adds_epi8(word1,word2)

Sum	00111000	10001100	10100110	1011111	11111000	10100011	00011000	00101011

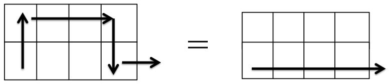
Single instruction multiple data processing uses a single computer word, that is then divided into smaller bytes. When an operation is preformed on a computer word it applies the operation to each of the bytes simultaneously, thus increasing efficiency since you only have to apply one operation for 8 bytes, rather than 8 operations.

Algorithm

Step 1: Calculate Initial Delta V values based off the previously set Delta H values of the previous row. To begin the algorithm, this is initialized as a row of all gaps. A horizontal sum of the delta H values is added to the wraparound match, or mismatch, value and added to the initial delta V of the current row to set the first DV value of row i.



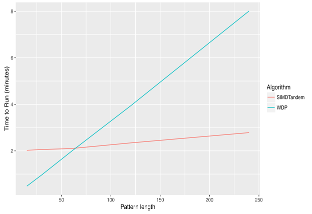
Step 2: After setting initial Delta V values, for the first pass of the pattern, the horizontal sum of the current row is calculated and used to determine if a horizontal wraparound is preformed on the second pass of the pattern. If the sum of the Delta H values of the current row is greater than zero, the calculated score value a pattern length to the left is better and the horizontal wraparound is performed.



Step 3: Now that correct Delta V values are calculated, the Delta H values of row i are calculated based on the Delta V values just calculated and the Delta H values of the previous row.

This process is iterated through each row until the final row of Delta H values are calculated. To calculate the score all of the Delta V values in the last position are stored and added and then added to the gap penalty times length of the pattern to yield a final score.

Results



The comparison of run times between traditional dynamic wraparound programing solution and the SIMD tandem program was done for 250,000 alignments with averages over 3 trials shown. Pattern lengths of 15, 30, 60, 120, and 240 searching in texts of length 960.

Discussion

The SIMD tandem alignment increases the efficiency of aligning VNTR patterns to full text sequences. The SIMD tandem alignment algorithm outlined here uses a series of shifts that are very costly in computational power and require a complexity of O(n * log(W)/W), in which W is the number of values held in the word. To increase the efficiency of the process, we can replace some of the shifts to parallel scan method, a newly developed method which can accomplish the same task, but in a linear complexity of O(n). This will increase the efficiency of the overall algorithm.

- G. Benson. Tandem repeats finder: a program to analyze DNA sequences. Nucleic Acids Research, 27:573–580, 1999. (doi:10.1093/nar/27.2.573).
- J. Loving, Y. Hernandez, and G. Benson. Bitpal: A bit-parallel, general integer-scoring sequence alignment algorithm. Bioinformatics, 30(22):3166–3173, 2014.
- J. Loving, E. Becker and G. Benson, Bit-Parallel Alignment with Substitution Scoring, Proceedings of the 8th International Conference on Bioinformatics and Computational Biology (BICoB-2016), Las Vegas, Nevada, April 2016.

Acknowledgements

This work was funded, in part, by NSF grant IIS-1423022 and the Boston University Bioinformatics BRITE REU program.