ITPP0590
ICT

An Enhanced Dynamic-programming Technique For Finding Approximate Overlaps

10.5339/qfarc.2014.ITPP0590

Maan Haj Rachid; Qutaibah Malluhi

Corresponding author :
mh1108047@qu.edu.qa
Qatar University, Doha, Qatar

Abstract

The next generation sequencing technology creates a huge number of sequences (reads), which constitute the input for genome assemblers. After prefiltering the sequences, it is required to detect exact overlaps between the reads to prepare the necessary ingredients to assemble the genome.
The standard method is to the find the maximum exact suffix-prefix match between each pair of reads after executing an error-detection technique. This is applied in most assemblers, however, a few studies worked on finding the approximate overlap. This direction can be useful when error detection and prefiltering techniques are very time consuming and not very reliable.
However, there is a huge difference in term of complexity between finding exact and approximate matching techniques. Therefore, any improvement in time could be valuable when approximate overlap is the target. The naive technique to find approximate overlaps applies a modified version of dynamic programming (DP) on every pair of reads, which consumes O(n2) time where n is the total size of all reads.
In this work, we take advantage of the fact that many reads share prefixes. Accordingly, it is obvious that some work is continuously repeated.
For example, consider the sequences in Figure 1. If dynamic programming is applied on S1 and S2, assuming S2 and S3 share a prefix of length 4, then it is easy to notice that calculation of a portion of DP table of size |S1| X 5 can be avoided when applying the algorithm on S1 and S3 (the shaded area in Figure 1).

Figure 1. DP table for S1,S2 alignment. We assume the following: gap = 1, match =0 and mismatch=1. no calculation for the shaded area is required when calculating S1,S3 table since S2,S3 share the prefix AGCC.
The modification is based on the above observation: first, the reads are sorted in lexicographical order and the largest common prefix (LCP) between every two consecutive reads is found. Let group G denote the reads after sorting. For every string S, we find the DP table for S and every other string in G. Since the reads are sorted, a portion of DP table can be skipped for every string, depending on the size of LCP, which has already been calculated in the previous step.
We implemented the traditional technique to find approximate overlap with and without the proposed modification. The results show that there is an improvement of 10-61% in time. The interpretation for this wide range is that the gain in performance depends on the number of strings. The larger the number of strings is, the better the gain in performance since the sizes of LCPs are typically larger.