Bioinformatics, 31(11), 2015, 1833–1835
doi: 10.1093/bioinformatics/btv052
Advance Access Publication Date: 30 January 2015
Applications Note

OXFORD

Sequence analysis

SFA-SPA: a suffix array based short peptide assembler for metagenomic data

Youngik Yang, Cuncong Zhong and Shibu Yooseph*

Informatics Department, J. Craig Venter Institute, La Jolla, CA 92037, USA

*To whom correspondence should be addressed.
Associate Editor: Alfonso Valencia

Received on June 16, 2014; revised on December 29, 2014; accepted on January 26, 2015

Abstract

Summary: The determination of protein sequences from a metagenomic dataset enables the study of metabolism and functional roles of the organisms that are present in the sampled microbial community. We had previously introduced algorithm and software for the accurate reconstruction of protein sequences from short peptides identified on nucleotide reads in a metagenomic dataset. Here, we present significant computational improvements to the short peptide assembly algorithm that make it practical to reconstruct proteins from large metagenomic datasets containing several hundred million reads, while maintaining accuracy. The improved computational efficiency is achieved using a suffix array data structure that allows for fast querying during the assembly process, and a significant redesign of assembly steps that enables multi-threaded execution.

Availability and implementation: The program is available under the GPLv3 license from sourceforge.net/projects/spa-assembler.

Contact: syooseph@jcvi.org

Supplementary information: Supplementary data are available at *Bioinformatics* online.

1 Introduction

Next-generation sequencing (NGS) technologies are used routinely in metagenomics, and provide a cost-effective approach to assay microbial communities at a high sequencing depth (Simon and Daniel, 2011). However, the computational analysis of these large datasets is a challenge and is influenced by the genomic composition and taxonomic divergence of the constituent microbes in the sampled community. In particular, effective identification of sequence and function of proteins from metagenomic datasets is hampered by the short read lengths generated by NGS technologies, and also by the fragmented nature of metagenomic assemblies (of nucleotide reads) that are attempted to reconstruct genome sequences. To address this problem, we had previously introduced a framework that reconstructs proteins from metagenomic data by first identifying partial protein sequences (or short peptides) from the nucleotide reads using a gene finder and subsequently assembling these short peptides into long protein sequences using a novel algorithm (Yang and Yooseph, 2013). Our short peptide assembler (SPA) algorithm and framework was compared to the alternate strategy of first assembling the nucleotide reads and then identifying proteins on the assembled contigs; several different nucleotide assemblers were evaluated. We showed that SPA outperforms the alternate strategy under multiple evaluation criteria, including specificity, sensitivity, read assembly rate and chimera rate (Yang and Yooseph, 2013). However, while our evaluations included datasets containing more than 100 million reads, we noted that the computational efficiency of SPA could be improved. Here, we present computational improvements to SPA that result in a substantial speedup of the assembly process, while maintaining assembly output quality. Our new assembly algorithm and implementation, denoted as SFA-SPA, uses the suffix array (Manber and Myers, 1993)—a space efficient data structure that also allows for the computationally efficient querying of patterns in a given text. The suffix array is used to speed up the initial path finding step of SPA, which is a major run time bottleneck. In addition, several assembly steps were significantly redesigned and re-implemented, and these improvements now allow

for the practical application of the SPA framework on large metagenomic datasets containing several hundred million reads using reasonable computational resources.

2 Method

SFA-SPA has four stages: (i) construction of a *de Bruijn* (or *k-mer*) graph (Idury and Waterman, 1995) from the set of short peptide sequences (henceforth called reads), and its subsequent traversal to identify a set of initial paths, (ii) extension and merging of these paths, (iii) clustering of highly similar paths in the resulting path set and (iv) recruitment of unassigned reads to these paths. The details of each stage and the improvements over SPA are presented in Supplementary S1. Here, we summarize the major improvement to stage 1. Let S_P be the amino acid sequence corresponding to a path P obtained by traversing the *de Bruijn* graph (Yang and Yooseph, 2013). We define the *read support* for path P to be the set RS of reads, where each read $r \in RS$ is such that either r is a substring of S_P, or the prefix of r is a suffix of S_P (referred to as prefix overlap), or the suffix of r is a prefix of S_P (referred to as suffix overlap). Each traversal of this directed graph is initiated by a chosen seed k-mer. A path is extended first in one direction by adding a new sink vertex in each step, until a stopping criterion is encountered; subsequently, it is extended in the other direction by adding a new source vertex in each step until a stopping criterion is encountered. Stopping criteria include low *k-mer* coverage or a failure during repeat handling. At each extension step, a sink (or source) vertex is chosen that maximizes read support at that point; only reads in RS that have prefix (or suffix) overlaps of minimum length m are considered. In SPA, the bookkeeping associated with the read support set was implemented using a series of sort and set operations, and this was computationally expensive. In SFA-SPA, this is instead implemented using a generalized suffix array constructed from the input reads. Prefixes (or suffixes) of candidate path extensions are used as queries to identify the next sink (or source) vertex to extend the path. Suffix array construction was enabled using *libdivsufsort*, an open source library (Mori, 2008). In addition, longest common prefix querying capability was implemented and utilized to expedite the suffix array search (Kasai, *et al.*, 2001; Manber and Myers, 1993). We also implemented a data structure to obtain the number of prefix occurrences from the suffix array query results. The other major changes in SFA-SPA include improved prioritization and scheduling of seeds during graph traversal, simpler repeat handling and a restructuring of the graph traversal, read recruitment, multiple sequence alignment and sequence profile generation steps, to enable multithreading. In addition, several components in SPA were simplified, removed or re-implemented efficiently in SFA-SPA to reduce runtime. Instead of multiple rounds of clustering and consensus calls after each assembly step in SPA, SFA-SPA performs a single round of clustering and consensus call; furthermore, read placement during the initial path identification stage only allows exact matches. SPA's path extension step (using supporting reads to extend a single path in each direction) was also not implemented in SFA-SPA.

3 Results

The evaluations were done on a Linux server with Intel Xeon 2.0 GHz 64-bit processors and 128 GB RAM. Four datasets were used for evaluating the performance of SFA-SPA and SPA: (i) *latco+strep*: a simulated dataset containing Lactobacillus and Streptococcus strains (53 million reads); (ii) *marine.sim*: a simulated marine metagenome dataset (91 million reads); (iii) *oral.sim*: a simulated oral metagenome dataset (103 million reads); (iv) *stool.hmp*: a real human microbiome dataset generated from a stool sample (73 million reads). The simulated marine and oral metagenomes were evaluated under two sequencing error rate scenarios—0% error (reported here) and 1% error (see Supplementary S3.1).

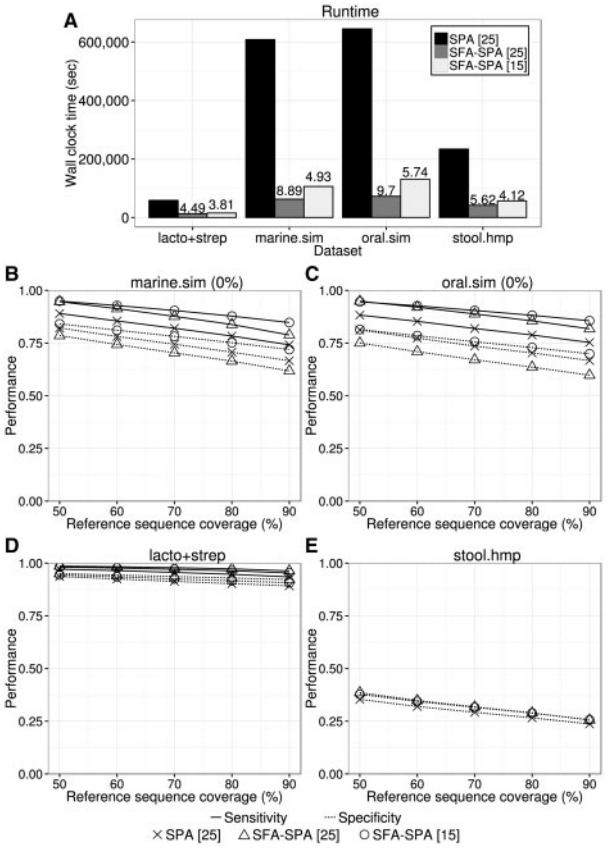

Fig. 1. Performance of SFA-SPA and SPA on the four datasets: (**A**) Total run time (wall clock) bar graph. The number above each SFA-SPA bar shows the fold speedup over SPA for that dataset. SPA is single threaded while the algorithm redesign allows SFA-SPA to use multiple threads (8 threads were used here). (**B,C,D,E**) Specificity (SP) and Sensitivity (SN) plots. The *x*-axis denotes the reference protein sequence length coverage while the *y*-axis denotes the SP and SN values. The value in square brackets denotes minimum overlap length (m)

SFA-SPA is significantly faster than SPA (Fig. 1A), with a nearly 10-fold speedup (for $m = 25$) for the total run time on the largest dataset (*oral.sim*) that we evaluated. This speed gain comes primarily from the use of the suffix array, and also from improved seed scheduling and multithreading; in single thread mode, SFA-SPA achieves close to a 13-fold speedup (for $m = 25$) over SPA for the graph traversal step and a 5.5-fold speedup overall (Supplementary S3.1, Fig. S2). With the computational gain, it becomes possible to tune the parameter settings to improve the accuracy of SFA-SPA compared to SPA. On these datasets, SFA-SPA ($m = 25$) has higher sensitivity but lower specificity compared to SPA ($m = 25$) (Fig. 1B–E). SPA's higher specificity is due to multiple rounds of path clustering, merging, extension and consensus calls which result in slightly fewer fragmented final paths. Path fragmentation can be ameliorated by lowering m; at $m = 15$, SFA-SPA has higher specificity and sensitivity compared to SPA (at $m = 25$) on these data and also on data with 1% sequencing error rate (Supplementary S3.1). Although SFA-SPA ($m = 15$) takes longer than SFA-SPA ($m = 25$)

(due to additional lookups needed to calculate the size of the read support set), it is still nearly six-fold faster than SPA ($m = 25$) on these data (Fig. 1A). Chimera rates are very low in both implementations and SFA-SPA achieves similar read assembly rates as SPA (Supplementary S3.1).

In addition to the effect of the minimum overlap length m, the effects of other parameters (minimum seed depth s and minimum read support r) on the performance of SFA-SPA were also evaluated (Supplementary S3.2 and S3.3); these evaluations highlight the relative importance of the overlap parameter m on the assembly quality. The peak memory usage for SFA-SPA is higher than that of SPA, with SFA-SPA using 95 GB RAM (2.8 times more than SPA) for the *oral.sim* dataset (Supplementary S3.1). We note though that the sizes of the suffix array and its associated data structures grow linearly with the input size. The increased memory usage for SFA-SPA is expected given the space versus run time tradeoff in our algorithm redesign; however, this is not an issue since hardware costs associated with these physical memory size requirements are reasonable. Finally, the run time of SFA-SPA is comparable to that of other nucleotide sequence assemblers for metagenomic data (Supplementary S3.4).

Funding

This material is based upon work supported by the National Science Foundation [grant number DBI-1262295].

Conflict of Interest: none declared.

References

Idury,R.M. and Waterman,M.S. (1995) A new algorithm for DNA sequence assembly. *J. Comput. Biol.*, **2**, 291–306.

Kasai,T. *et al.* (2001) Linear-time longest-common-prefix computation in suffix arrays and its applications. In: Amir,A. (ed.) *Combinatorial Pattern Matching*. Springer, Berlin, pp. 181–192.

Manber,U. and Myers,G. (1993) Suffix arrays: a new method for on-line string searches. *SIAM J. Comput.* **22**, 935–948.

Mori,Y. (2008) libdivsufsort—a lightweight suffix-sorting library. https://code.google.com/p/libdivsufsort/. (24 February 2004, date last accessed).

Simon,C. and Daniel,R. (2011) Metagenomic analyses: past and future trends. *Appl. Environ. Microbiol.*, **77**, 1153–1161.

Yang,Y. and Yooseph,S. (2013) SPA: a short peptide assembler for metagenomic data. *Nucleic Acids Res.*, **41**, e91.