Bioinformatics, 36(13), 2020, 3975–3981
doi: 10.1093/bioinformatics/btaa272
Advance Access Publication Date: 24 April 2020
Original Paper

Sequence analysis

Targeted domain assembly for fast functional profiling of metagenomic datasets with S3A

Laurent David[1], Riccardo Vicedomini[1,2] Hugues Richard[1,*,†] and Alessandra Carbone[1,3,*]

[1]Sorbonne Université, CNRS, IBPS, Laboratoire de Biologie Computationnelle et Quantitative (LCQB), UMR 7238, [2]Sorbonne Université, CNRS, Institut des Sciences du Calcul et des Données (ISCD) and [3]Institut Universitaire de France, Paris 75005, France

*To whom correspondence should be addressed.
[†]Present address: Bioinformatics Unit (MF1), Robert Koch Institute, Berlin 13353, Germany
Associate Editor: Inanc Birol

Received on October 16, 2019; revised on April 11, 2020; editorial decision on April 16, 2020; accepted on April 17, 2020

Abstract

Motivation: The understanding of the ever-increasing number of metagenomic sequences accumulating in our databases demands for approaches that rapidly 'explore' the content of multiple and/or large metagenomic datasets with respect to specific domain targets, avoiding full domain annotation and full assembly.
Results: S3A is a fast and accurate domain-targeted assembler designed for a rapid functional profiling. It is based on a novel construction and a fast traversal of the Overlap-Layout-Consensus graph, designed to reconstruct coding regions from domain annotated metagenomic sequence reads. S3A relies on high-quality domain annotation to efficiently assemble metagenomic sequences and on the design of a new confidence measure for a fast evaluation of overlapping reads. Its implementation is highly generic and can be applied to any arbitrary type of annotation. On simulated data, S3A achieves a level of accuracy similar to that of classical metagenomics assembly tools while permitting to conduct a faster and sensitive profiling on domains of interest. When studying a few dozens of functional domains—a typical scenario—S3A is up to an order of magnitude faster than general purpose metagenomic assemblers, thus enabling the analysis of a larger number of datasets in the same amount of time. S3A opens new avenues to the fast exploration of the rapidly increasing number of metagenomic datasets displaying an ever-increasing size.
Availability and implementation: S3A is available at http://www.lcqb.upmc.fr/S3A_ASSEMBLER/.
Contact: hugues.richard@upmc.fr or alessandra.carbone@lip6.fr
Supplementary information: Supplementary data are available at *Bioinformatics* online.

1 Introduction

Next-generation sequencing of environmental samples (e.g. metagenomics) aims at studying microbial communities (Allen and Banfield, 2005; Eisen, 2007). It is commonly followed by a functional annotation of the predicted coding regions to describe the community's metabolic activities (De Filippo *et al.*, 2012; Escobar-Zepeda *et al.*, 2015). This consists in annotating domains and functional motifs within amino acid sequences (Finn *et al.*, 2011; Ugarte *et al.*, 2018). In metagenomics, annotation is hampered for shorter sequences of 100–150 bp in length—common with current technologies—thus making sequence assembly a prerequisite for any improvement. In this context, a good-quality assembler is necessary, as it increases the length of assembled coding regions. The sheer size of metagenomic datasets typically requires huge time and memory resources when doing *de novo* metagenome assembly (Georganas *et al.*, 2018). Thus, several strategies have been proposed to perform a targeted assembly (Wang *et al.*, 2015; Zhang *et al.*, 2014), based on a preliminary protein domain annotation followed by a domain-guided assembly.

Domain targeted assembly has a second major advantage. Indeed, it can be restrained to a limited number of domains, from a few 10s to the 100s, providing a fast way to 'explore' many large metagenome datasets with a given hypothesis in mind. Metagenomics studies are usually interested in understanding one given function or biochemical pathway across multiple conditions or samples, and, in practice, only a limited number of domains (a few dozens) needs to be annotated when profiling. Examples range from the annotation of RNA transcripts in extreme environments (Buelow *et al.*, 2016), to a particular biochemical reaction in the gut microbiota (Tagliabue *et al.*, 2017; Vital *et al.*, 2017), to the detection of antimicrobial resistance (Jia *et al.*, 2017). Various targeted assemblers were proposed for performing this task. They can either perform an assembly around an identified domain (Zhang *et al.*, 2014) or annotate domains after reads' clustering (Keegan *et al.*, 2016; Wilke *et al.*, 2016). On very large datasets, the first are unable

to scale and the latter are excessively slow. To overcome this limitation, we developed the Scalable Accurate Annotated Assembly tool S3A. S3A combines a step of fast reads clustering [using BCALM 2 (Chikhi *et al.*, 2016)] with an efficient assembly performed from domain annotation. S3A is in practice as accurate, more sensitive and up to one order of magnitude faster than existing targeted domain assembly tools like the SAT assembler (Zhang *et al.*, 2014). It is slightly more precise than the Xander assembler (Wang *et al.*, 2015) showing the same computational efficiency on up to 100 domains. It is on par with traditional assemblers, such as Minia (Chikhi and Rizk, 2013), when considering up to 100 domains. When considering realistic metagenomic dataset analyses on a few dozens of domains, S3A can, in the same running time and final accuracy as a metagenomic assembler, annotate six to eight times more samples.

2 The S3A approach

S3A is a tool for targeted domain search in metagenomic datasets. It is designed as an assembly algorithm of annotated reads addressing the problem of reducing the time complexity of the Overlap-Layout-Consensus (OLC) graph construction step, the bottleneck of targeted assembly. The S3A flowchart is depicted in Figure 1. It starts from a dataset of metagenomic reads, performs a preprocessing of the reads to reduce the size of the set by constructing 'protigs', that is protein unitigs (see Section 3.2), through the detection of open reading frames (ORFs), their assembly in unitigs and a mapping of the nucleotide sequence into an amino acid sequence. Then, it parses protigs for a protein domain annotation, and it constructs the OLC graph, in the amino acid sequence space, based on the overlap of domain annotations: an OLC graph is a directed graph where each node corresponds to a protig, and each edge to an overlap between two protigs. Based on two metrics, used to identify unreliable edges and prune the graph (the longest matching substring length, *lmsl* and the percentage of identity, *ip*; see Section 3), this step performs a Depth-First Search (DFS) of the graph, called 'graph traversal', to identify and assemble a set of overlapping protigs representing consensus DNA regions surrounding protein domains, referred to as contigs. S3A outputs a set of contigs for each targeted domain.

S3A exploits functional domain annotation as a first indicator for protig overlap, making OLC graphs a central choice for targeted assembly. The construction of the OLC graph in S3A is different from the traditional one used for sequence assembly, where read pairwise alignments are evaluated by the Hamming distance of the overlapping region. Instead, in S3A, the quality of overlapping regions is evaluated by the two fast computable measures *lmsl* and *ip*. An OLC graph is also different from a de Bruijn graph, used in (nontargeted) assembly algorithms, which replaces every read with the corresponding set of *k*-mers (Zerbino and Birney, 2008).

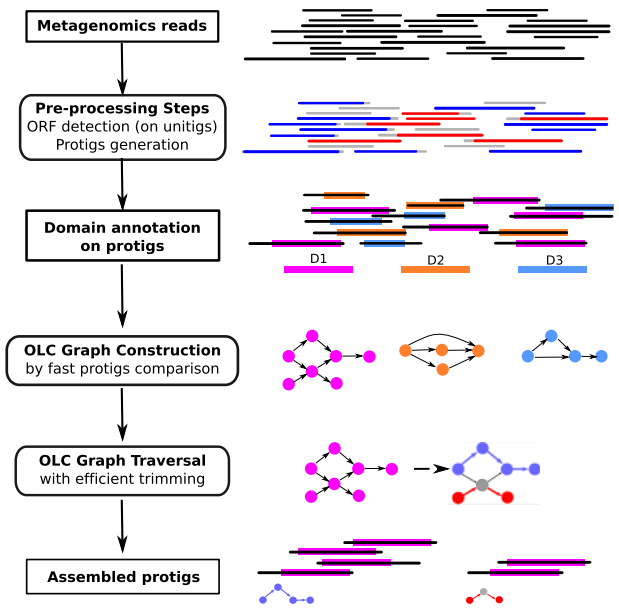

Fig. 1. S3A flowchart. From domain annotated protigs, obtained by reads' preprocessing (ORF identification, unitig generation and mapping into amino acid sequence), S3A performs an efficient protigs comparison and builds an OLC graph. A graph traversal approach based on an efficient resolution of alternative paths combining two measures of sequence overlap, allows to assemble protigs

2.1 S3A key features

The basic choice of S3A to assemble domain annotated sequences is important for directly deriving the functional annotation of the metagenomic sample. The second main motivation to considering domain target assembly is the reduction of the time complexity while retaining the highest accuracy possible. Indeed, by separating and ordering reads by domain in the preprocessing step, the number of read comparisons is highly decreased and the general algorithm performance is greatly improved.

S3A evaluates overlapping reads sharing a common domain annotation, on the basis of two metrics, *lmsl* and *ip*. These metrics provide an overlapping confidence measure that is both complementary and much faster than counting a fixed number of mismatches by dynamically computing an edit distance, as done by other targeted assemblers like SAT (Zhang *et al.*, 2014). They also allow for a tailored OLC graph trimming which is independent on the sequencing technology used and helps reducing the graph complexity. Moreover, *lmsl* is used to resolve ambiguous cases in the absence of transitive edges, and to select the most reliable transitive edges in the OLC graph (Fig. 2).

S3A might create complex OLC graph structures due to *chimeric nodes*, that are nodes with multiple entry and exit edges. In the absence of transitive edges, chimeric nodes are considered unreliable and therefore removed, the goal of S3A being to be as accurate as possible.

Most importantly, the possibility to annotate a reduced set of domains and assemble only reads involving these domains, allows for a fast exploration of metagenomic datasets allowing the user to concentrate on specific functional targets.

3 Materials and methods

3.1 Domain hit

Given a sequence *r* annotated with a given domain *d*, a portion or all of *r* will match to the domain. We define the domain hit region for *r* as the start and end positions of the sequence matching interval, relative to the whole domain (denoted *s* and *e*). Domains' annotation is realized on protigs, that is amino acid sequences generated by a preprocessing step that identifies ORF regions in metagenomic data (see Fig. 1 and Section 3.2).

3.2 Data preprocessing

Metagenomic sequences are prepared before assembly using three main steps. First, an ORF prediction is realized with FragGeneScan (Rho *et al.*, 2010), checking both the forward and the reverse strands of a read. Second, predicted ORF sequences are assembled into unitigs obtained with BCALM 2 (Chikhi *et al.*, 2016), where a unitig is a local sequence assembly whose overlap are not disputed by any other data. This step reduces greatly the time needed for domain annotation. Third, the translation of the nucleotide sequence in amino acid sequence is followed by a functional domain annotation realized with HMMER (Finn *et al.*, 2011) or MetaCLADE (Ugarte *et al.*, 2018). Any type of annotation can be used. Sequences remaining without domain annotation or annotated with more than one domain are discarded.

As a result, S3A performs the assembly of a set of amino acid sequences coming from coding regions and annotated with functional domains. They originate either directly from reads corresponding to ORF sequences or from unitigs constructed from ORF sequences. For simplicity, in the sequel, we shall refer to them as *protigs*.

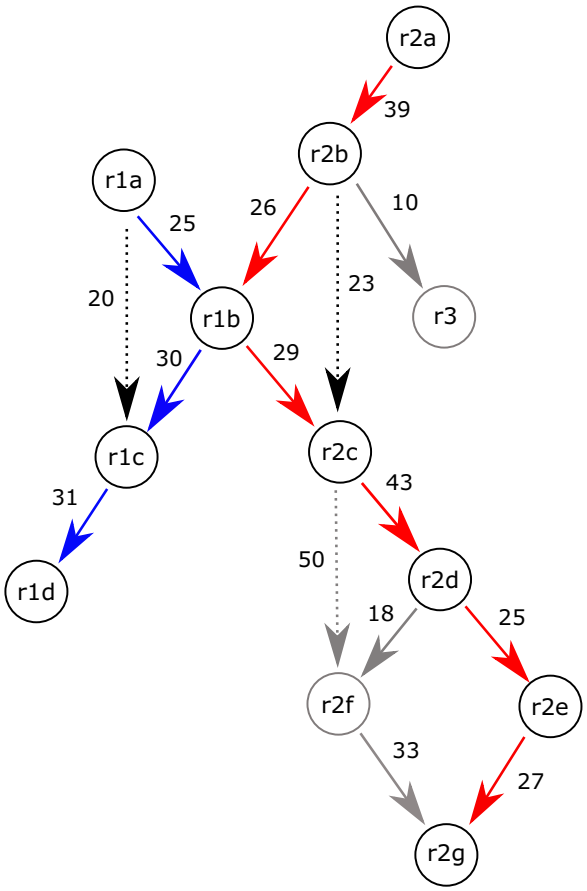

Fig. 2. A toy model illustrating graph pruning and graph traversal for the generation of two contigs by resolving a chimeric node. Each edge is weighted with the *lmsl* of its nodes. We fix a threshold of 15 for acceptable *lmsl*. Note that, for sake of simplicity, the weights are not normalized in this toy example. **Graph pruning step**: the node r3 is removed due to the *lmsl* threshold (see gray edge and gray node). Edges (r1a, r1c) and (r2b, r2c) are identified by transitive reduction and placed in the transitive dictionary, as their *lmsl* weight is not stronger than the *lmsl* weight of any edge in the corresponding path (see edges in dotted lines). Note that the weight of (r2c, r2f) is too high to place this edge in the transitive dictionary and consider it further (gray arrow). **Graph traversal step**: the two source nodes r1a and r2a are considered for traversal, one after the other (their order is chosen arbitrarily). First, the contig (r1a, r1b, r1c, r1d) (blue path) is extracted, the path ambiguity at node r1b being solved thanks to the (r1a, r1c) transitive edge. The same consideration applies to (r2a, r2c) where the transitive edge (r2b, r2c) helps to resolve the ambiguity at node r1b. r2d is a branching node with no transitive edge [(r2c, r2f) has been removed in the graph pruning step], hence the path with highest *lmsl* weight is chosen. The second contig is thus (r2a, r2b, r1b, r2c, r2d, r2e, r2g) (red path). Note that r1b is a *chimeric node*, indicating that the corresponding protig should not be part of two contigs; the presence of transitive edges allows for an unambiguous traversal

3.3 The S3A algorithm

S3A consists of two main steps, retracing the architecture of SAT (Zhang *et al.*, 2014): the OLC graph construction and the OLC graph traversal. However, the corresponding algorithms are significantly different. The OLC graph construction combines domain evidence with a fast estimation of the protigs overlap, and the OLC graph traversal uses an efficient dynamic programming algorithm based on edge weights to guide the traversal more efficiently. These two steps are described in details below.

3.3.1 OLC graph construction

Let M be the number of domains involved in the annotation process. We consider each protig to have a single domain annotation, and protigs not having a single domain annotation are discarded. Protigs are first grouped in a set of M hashtables $\{\mathcal{H}_1, \ldots, \mathcal{H}_M\}$, one per domain. For each hashtable $\mathcal{H}_i$, protigs are sorted according to their s (start) and e (end) position on domain i. This data organization, sorting protigs by their matching position on the domain within domain specific hashtables, breaks down the OLC graph construction to a simple interval traversal algorithm that avoids comparing each possible pair of protigs. Namely, no comparison between pairs of protigs (i) annotated by different domains, nor (ii) having no domain overlapping (where two protigs have a *domain overlap* if their respective domain hits overlap; Supplementary Fig. S2, top) is needed.

The OLC graph is constructed by creating an edge between each pair of protigs (r_i, r_j) that overlap by more than c amino acids on the same domain (Supplementary Fig. S2). To each edge (r_i, r_j), we add two metric values computed from r_i and r_j nucleotidic sequences: the length of the longest matching substring ($lmsl_{i,j}$) and the percentage of identity between r_i and r_j on the domain overlap region ($ip_{i,j}$). We prefer the use of *lmsl* and *ip* over more precise ones (e.g. edit distance), as they estimate sequence similarity much faster.

By construction, the OLC graph is directed, but not necessarily acyclic. Ideally, each occurrence of a domain in a gene should give rise to a path in the graph. Thus, a graph pruning step will make the graph acyclic and a graph traversal step will identify contigs by traversing the graph from each *source* node (nodes without predecessors), and using scoring paths according to the *lmsl* values stored on the edges (see below and Supplementary Methods for algorithmic details).

However, sequencing errors and sequence similarity between genes and species can create ambiguities in the traversal and lead to chimeric nodes (nodes with at least 2 predecessors and 2 successors). To help solve those ambiguities, we identify *transitive edges*, edges that connect two nodes which have an alternative path joining them. These edges can be removed without losing information for the traversal, but are kept in a separate data structure $\mathcal{T}$, as they can help resolve ambiguities raised by chimeric nodes (see Supplementary Fig. S3).

3.3.2 Graph pruning and traversal

Once the OLC graph is built, contigs can be generated from its traversal. The graph traversal is preceded by a pruning phase that removes unreliable edges based on *lmsl* and *ip* values (step 1), and enforce an acyclic graph (step 2). The graph is then simplified by merging linear paths (step 3) and transitive edges that have no impact on the traversal are efficiently removed (step 4) (see Supplementary Methods). The resulting structure is a directed acyclic graph with N sources nodes. N traversals are finally performed to build output contigs.

The graph is then visited in a depth-first manner, using transitive edges to resolve branching during the graph traversal. During the traversal, when a *chimeric node* v is visited, $\mathcal{T}$ is queried to look for transitive edges linking a predecessor u to one of its successors w. If such an edge exists, the traversal will be guided to the path containing both v and w. If not, v is removed, which implies that edges joining any such node to the rest of the graph are discarded as well. As a result, its successors are therefore considered as additional source nodes of the graph.

Bubbles are other kinds of topologies that can lead to errors during the traversal. Bubbles exist in the graph when, given two nodes u and v, two alternative paths starting from u and ending in v exist. In that case, if no transitive edge exists, the ambiguity is resolved by removing the path containing the edge with the smallest *lmsl* value.

3.3.3 Parameters' default values

As default values, we used a minimal domain hit length of 20 amino acids for the protigs, an *ip* threshold of 80% and an *lmsl* threshold of 0.2.

The minimal domain hit length has been chosen to be the same as in the SAT assembler. Since we annotate protigs, which are longer in average than reads (see Supplementary Table S1), the threshold of

20aa is usually satisfied by domain annotation. A lower value would increment the number of false positives.

To compute the *lmsl* between two overlapping protigs, we normalize the longest matching substring length by the mean length of the protigs obtained for the metagenomic dataset under analysis, allowing the default threshold to be used with reads coming from whatever sequencing technology (Illumina, 454). Figure 3A–C (see highlighted nodes in the green curves) shows that for our datasets, a threshold of normalized *lmsl* at 0.2 gives high precision and acceptable sensitivity.

The *ip* threshold has been set to 80% because a significantly different percentage proves to be either too lax or too strict (Supplementary Fig. S1). Figure 3A–C and Supplementary Figure S1 demonstrate that S3A's best performance is achieved around the default values.

3.3.4 Analysis of the time complexity

Let R be the number of annotated protigs and M be the number of domains. If we consider R_M the number of protigs for the same domain, then only protigs annotated from the same domain are compared. Moreover, the protigs are sorted according to their starting position on the domain which gives an overall $O(R \log R_M)$ time complexity for the graph construction. The graph traversal is based on the DFS algorithm, whose complexity is $O(|V| + |E|)$, where $|V|$ is the number of nodes and $|E|$ the number of edges (hence overlaps). Moreover, our transitive reduction step is quadratic in $|V|^2$ in the worst case, as a DFS is performed for every node in the graph. However, the depth of each search is bounded by the number of occurrences of the considered domain.

3.4 Datasets

To evaluate S3A, we considered a total of seven datasets of metagenomic sequences whose properties are summarized in Table 1. For datasets with pair-end information, which is not used in S3A, pairs of reads have been used as independent reads.

Five of the datasets are synthetic datasets. Three of them were simulated according to two technologies sampling 55 equally abundant species (11 archeal and 44 bacterial). To simulate reads we used MetaSim (Richter *et al.*, 2008), based on a read length which is characteristic for 454 and Illumina sequencing, and different coverages (7× and 30×). FlowSim (Balzer *et al.*, 2010) was then applied to obtain insertion and deletion sequencing error patterns corresponding to the respective DNA sequencing technologies. The three datasets have been used to compare S3A and SAT. Two more datasets were taken from the critical assessment of metagenome interpretation (CAMI) challenge (Sczyrba *et al.*, 2017), to compare S3A to the classical metagenomic short-read assembler Minia (Chikhi and Rizk, 2013). CAMI is a worldwide benchmarking challenge aiming at the thorough evaluation of metagenome assembler performance. We selected two types of complexity: low (30 genomes) and high (450 genomes).

A real dataset was taken from a microbial community analysis of the Arid soil of McMurdo Valley in Antartica (Buelow *et al.*, 2016) and a second one from the butyrate-producing community in the gut microbiota (Vital *et al.*, 2017). They have been assembled and annotated by S3A. The McMurdo Valley dataset was analyzed according to the 24 domains, related to soil communities in an extreme desert environment, reported in Buelow *et al.* (2016). These datasets total 12.5 Gbp of sequence for an average 2×100 bp read length. The gut microbiota dataset was analyzed according to three domains

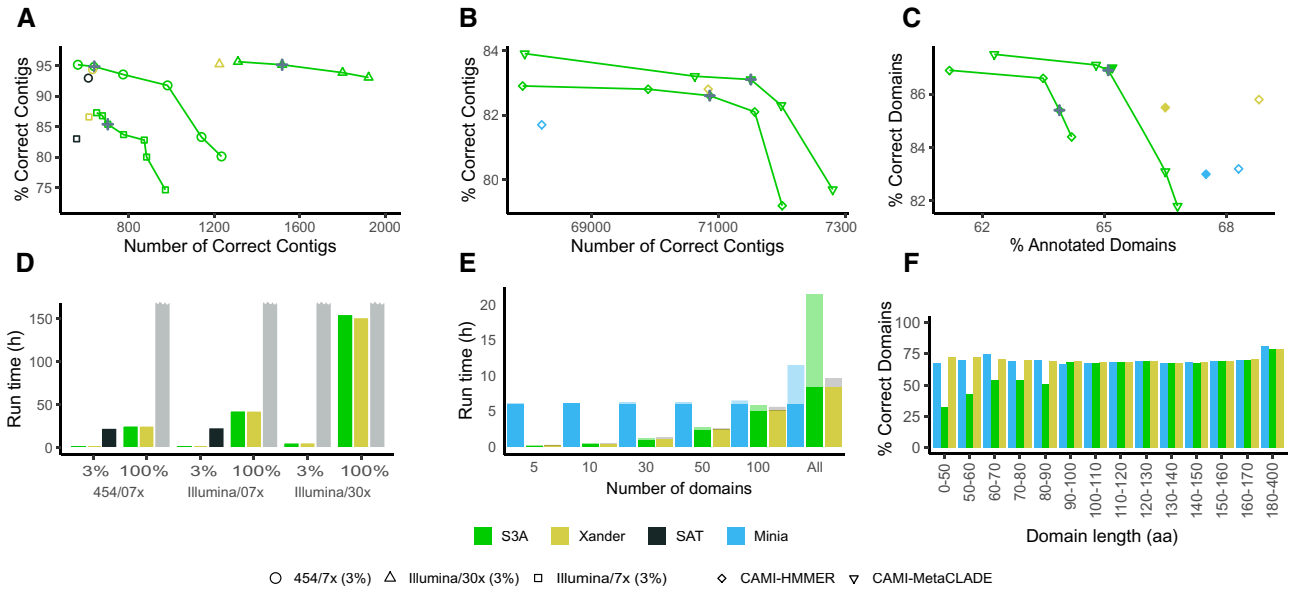

Fig. 3. Performance comparison among S3A, SAT, Xander and Minia. The analyses are reported in green for S3A, gold for Xander, black for SAT and blue for Minia. (**A**) S3A, Xander and SAT precision (reported as '% Correct Contigs') is computed on different simulated datasets (of reduced size, 3% of the reads in the full dataset). The green curves are realized by varying the *lmsl* parameter values and keeping *ip* fixed at 80% (default value). By decreasing the *lmsl* values, the number of assembled contigs augments but the proportion of correct ones decreases, and the curves show how fast precision deteriorates. S3A precision at default values is reported with a gray cross. (**B**) S3A, Xander and Minia precision computed on the high-complexity CAMI dataset and reported as '% Correct Contigs'. Similarly to panel A, green curves are computed by varying the *lmsl* values while keeping the *ip* default value. The upper green curve corresponds to S3A assemblies using MetaCLADE annotation, whereas the lower curve is obtained with HMMER annotation. S3A precision at default values computed using MetaCLADE and HMMER annotations is reported by grey crosses. (**C**) Comparison between S3A, Minia and Xander on domain precision ('% Correct Domains') and domain recall ('% Annotated Domain'). Precision is computed for different *lmsl* thresholds on the whole high complexity CAMI toy dataset. Domain precision and recall have been computed for Minia and S3A using HMMER annotation. S3A was also analyzed on MetaCLADE annotation. S3A performance at default values is shown by a gray cross. Filled points correspond to the average accuracy obtained over 5 selections of 100 domains drawn at random. Compare with panel F (see also text) for differences among S3A, Minia and Xander. (**D**) Run time performance of S3A, SAT and Xander computed on simulated datasets of two sizes (3% or 100% of the sample), for varying technologies (454 or Illumina) or coverages (7× or 30×). SAT execution did not complete after 240 h on the larger datasets (represented as a gray bar). (**E**) Run time performance of S3A, Minia and Xander on the low complexity CAMI dataset on restricted number of domains, from 5 up to 100, and on the full dataset. Time devoted to functional annotation is highlighted in light colors, and time for domain model construction in dark tones (for Xander). (**F**) S3A, Xander and Minia comparison on domain recall, for different domain lengths. For each size range, the number of domain occurrences detected by each assembler is compared to the total number of domain occurrences detected on the dataset gold standard. (Color version of this figure is available at *Bioinformatics* online.)

Table 1. Summary of the characteristics of the datasets used for evaluation

Name	Technology	Annotation	N. genomes	bps covered by reads	Read length	Read count
454/7×	454	MetaCLADE	55	15.5 Gbp	450 bp	3.5M
Illu/7×	Illumina	HMMER	55	15.7 Gbp	150 bp	10.5M
Illu/30×	Illumina	HMMER	55	67.5 Gbp	150 bp	45M
CAMI/low	Illumina	HMMER	30	15 Gbp	2×100 bp	15M
CAMI/high	Illumina	HMMER	450	75 Gbp	2×100 bp	75M
Arid soil—McMurdo valley	Illumina	MG-RAST	NA	12.5 Gbp	2×100 bp	12.5M
Butyrate-producing community	Illumina	Xander	NA	9 Gbp	2×100 bp	9M

reported in Vital *et al.* (2017). The data were originally organized in several datasets, and we considered (with a random choice) three of them. The properties of the two real datasets are reported in Table 1.

3.5 Evaluation procedure

To compare S3A with other assemblers on synthetic data, we need to rely on a ground truth that considers the fact that S3A is restricted to domain annotated regions. To build this ground truth, we perform a domain annotation (with HMMER or MetaCLADE) and analyze the performance of the tools on the same domain annotated ground truth. Gene fraction is defined on the same ground truth.

We evaluate S3A either with respect to contigs correctly assembled around a protein domain, or with respect to correctly annotated domains in the metagenomic dataset. For this, we say that 'an assembly is correct' when the two protigs are correctly localized on the reference genome and 'an assembly is incorrect' when at least one of the protigs involved in the assembly is not correctly placed on the reference genome.

To evaluate S3A on contigs, we rely on the following three quantities: the number of contigs correctly assembled (true positives, TP), the number of contigs incorrectly assembled (false positives, FP) and the number of correct contigs that have not been assembled (false negatives, FN). We compute S3A precision (positive predictive value) as TP/TP+FP and S3A recall (sensitivity) as TP/TP+FN.

To evaluate S3A on domain annotated contigs, we rely on the following quantities: the number of contigs with correct or incorrect assembly that are correctly annotated (TP), the number of contigs with correct or incorrect assembly that are incorrectly annotated (FP), the number of domain occurrences in the ground truth annotation that are missed. Precision and recall are computed as above, where TP, FP and FN are defined with respect to domain annotated contigs. This evaluation does not demand assemblies to be correctly placed on reference genomes, and this is especially important when functionally profiling metagenomic sequences. Indeed, metagenomic datasets are often composed of sequences coming from very close species and an assembly of protigs from close origins appears reasonable for the functional annotation of a community.

3.6 Influence of thresholds to evaluate precision

Along with limiting the complexity of the graph, different threshold values for *lmsl* (at fixed *ip*, see Supplementary Fig. S1) can be used to improve precision while recovering a sufficient portion of true overlaps. In this way, S3A could be used as a step to assemble longer genes, which will subsequently be annotated with better accuracy. Note that the *lmsl* metric is less sensitive to false-positive matches than other measures we tested.

3.7 Evaluation of running time

Running time evaluations of S3A, SAT, Xander and Minia have been realized from simulated and real datasets of metagenomic reads. For S3A running time calculation, we considered the entire S3A pipeline (Fig. 1), including preprocessing steps and domain annotation. All the evaluations have been run on the same machine with the following configuration: Intel Xeon CPU E5-2670 (2.6 GHz), 128 Gb of RAM, using 16 threads.

4 Results

4.1 S3A improves precision, recall and running times over other targeted assemblers

S3A has been tested on three synthetic datasets simulated according to Illumina and 454 technologies, with different read lengths (150 and 450 bp) and coverages (7× and 30×) (see Section 3). They are based on a large number of species (55) which is enough to capture most of the challenges for metagenome assembly (repeated regions, chimeric nodes). Working with simulated data has the advantage that all real overlaps are known and, as a consequence, we could precisely compare S3A with the targeted assembler SAT (Zhang *et al.*, 2014) and Xander (Wang *et al.*, 2015).

Performing targeted assembly improves significantly domain annotation in comparison with annotation on raw reads. Indeed, the false-positive rate of S3A annotated domains decreases between twofold and tenfold with respect to the one observed while annotating raw reads. Note that constructing protigs yields an improvement up to twofold (Supplementary Table S1, middle) and that contig assembly consistently decreases the rate of incorrect domain annotation by producing longer sequences (with a median length increase of 50–70 amino acids over raw reads; Supplementary Table S1).

We limited the analysis to datasets containing a small fraction of the reads (3%), and evaluated S3A, SAT and Xander performance by monitoring the run time and the precision of the predicted contigs. The algorithmic design, based on a smart sorting of reads aligned to a domain and a fast sequence overlap approximation, makes S3A around 10 times faster than SAT and on par with Xander (see Fig. 3B, columns '3%' and Supplementary Table S2). In practice, this means that S3A is capable of performing a targeted assembly for more than a million reads while this task remains impossible for SAT (Supplementary Table S2). S3A shows a slight but clear improvement in precision over SAT (Fig. 3A). In Figure 3A, we monitor S3A behavior with respect to an increasing number of correctly assembled contigs (TP) and show how fast S3A precision deteriorates. On the Illu/30× dataset, characterized by the highest coverage, the proportion of correct contigs assembled by S3A remains almost constant, around 94%, whereas the number of correctly assembled contigs almost doubles. Due to the size of this dataset, SAT could not run.

S3A is also more sensitive than SAT and Xander. Indeed, at equal precision level, S3A recovers 22% more correct contigs in the Illu/7× dataset and 5% more on the 454/7× dataset (Fig. 3A; see Section 3) than SAT demonstrating its robustness according to the technology choice (see Section 3 for threshold's robustness). On the same datasets, S3A is also slightly better than Xander (Fig. 3A). It is definitely more sensitive than Xander on the Illu/30× dataset, where it achieves between 24% (default parameter) and 57% more annotated correct contigs.

As reported in Supplementary Table S3, when precision is evaluated on correct domain annotation, S3A shows to annotate consistently more domains than SAT and Xander for all three restricted datasets.

Compared to Xander, S3A is more precise at an equivalent running time (Fig. 3C and Supplementary Table S2). Note that Xander cannot be run on the models of the MetaCLADE library because it generates its own HMMs (Hidden Markov Models) as part of the assembly step. Also, Xander runs on each domain separately while

S3A annotates several domains at once. In this respect, S3A design is more flexible in the treatment of multiple domains and independent from model construction.

4.2 Gain over whole metagenome assembly

To assess how accurate S3A is in domain annotation, we considered two datasets of low and high (depending on the number of species) complexity from the CAMI challenge (see Section 3). We compared S3A to Xander (Wang *et al.*, 2015) and to the Minia assembler (Chikhi and Rizk, 2013). We chose Minia as it was evaluated among the best tools in the CAMI benchmark (Sczyrba *et al.*, 2017). Like previously observed with simulated data, the precision on predicted contigs of S3A (83.1% MetaCLADE, 82.5% HMMER annotation) is on par with Xander and Minia (82.8 and 81.7%, respectively; compare Fig. 3A and B). However, the number of correctly predicted contigs is higher for S3A (71 512) than for Xander (70 842; Fig. 3B).

Given a list of domains, we wished to test whether they are present in the sample and, possibly, in which proportion. We first restricted our evaluation to the coding regions where a domain was annotated. We further limited the evaluation at the domain level, that is, we counted the number of domains that are correctly recovered in the sample. Figure 3C shows that, on the high complexity CAMI dataset, S3A global accuracy is on par with Minia and Xander, and that S3A can show higher precision in domain annotation. In contrast, Minia and Xander recover a larger fraction of domains than S3A. To explain the discrepancy, observe that S3A relies on matching domain annotated sequences and that, during protigs annotation, short domains are more likely to be missed than longer ones. As a consequence, S3A is expected to identify a smaller number of domain occurrences than Xander or Minia due to lower performance in short domains. We verified this hypothesis on the CAMI dataset by reporting the sensitivity of S3A, Minia and Xander according to domain length in Figure 3E. It shows that the behavior of the three tools is the same for domains larger than 90aa and up to 170aa, that S3A performs better for larger domains (>170aa), and that it is less sensitive than Minia and Xander for domains of length <90aa. However, only a small fraction of the domains are short, with those of length <90aa corresponding to the 8.8% of the total number of domains (Fig. 3F), and we would not expect this to impact the functional profiling of a sample in practice. To construct something more in line with the general use case, we also compared the performance on a randomly chosen set of 100 domains (Fig. 3C, filled points). While the precision of the method does not change much, the sensitivity of S3A is now on par with Minia and Xander.

On the low-complexity CAMI dataset, a much simpler assembly challenge, S3A, Minia and Xander precision is comparable and reaches 98% over more than 75% of annotated genes, as reported in Supplementary Table S4. S3A total running time is relatively longer (22 h) than Xander (10 h) or Minia (11 h) when tested on the whole collection of domains in PFAM v30. Indeed the domain annotation hampers the total running time for the targeted assembly as it is performed before domain assembly. It is less the case for Minia or Xander, where annotation is performed either during or before assembly. However, S3A was not designed to perform a full domain annotation, it is expected to be used when only 5–100 domains needs scrutiny. Restricting the number of domains reduces the running time of S3A such that it becomes faster than Minia by a factor of 6–10 and slightly faster than Xander (Fig. 3D). In practice, it permits to handle many more samples than a general purpose metagenomic assembler like Minia in the same amount of time. This is a significant practical gain when computing resources are limited and the user is studying dozens of domains (Buelow *et al.*, 2016).

4.3 Time performance on real datasets and comparison with MG-RAST and Xander

To assess the performance of S3A in a typical analysis workflow, we considered the microbial community of the Arid soil of McMurdo Valley in Antartica (Buelow *et al.*, 2016) and the butyrate-producing community in the a microbiota (Vital *et al.*, 2017). The McMurdo Valley dataset relies on read clustering and targeted HMMER annotation to uncover 24 domains. S3A correctly detects all domain occurrences much faster than MG-RAST (Keegan *et al.*, 2016; Wilke *et al.*, 2016) (2 h versus 8 h). It reconstructs around 7% more domain sequences than MG-RAST with a HMMER annotation, and 9% more when MetaCLADE is used for annotation. S3A hence provides more information for analysis and quantification (Supplementary Table S5).

A second performance analysis was realized with a gut microbiota dataset (Wang *et al.*, 2015) used to test the Xander assembler. It relied on a targeted HMMER annotation to uncover three domains involved in butyrate production. All domain occurrences have been detected by S3A in 1 h versus 1 h 10 with Xander. In contrast, S3A reconstructs around 3% more domain sequences than Xander with a HMMER annotation, and 5% more when MetaCLADE is used for annotation (Supplementary Table S5).

5 Discussion

The noticeable features of S3A are both its precision and reduced time complexity, especially when focusing on several domains of interest. In this sense, S3A does not try to outcompete traditional metagenome assemblers, but enables a tradeoff between the number of domains that are profiled and the number of samples that can be considered in the same amount of time. S3A performance can thus justify its use over classical assembler when a fast and sensible profiling of a dozen up to a few hundred domains is needed. Detection of antimicrobial resistance (Jia *et al.*, 2017), annotation of specific types of pathogenicity (Gussow *et al.*, 2016) and searching for indicators of a particular biochemical reaction (Tagliabue *et al.*, 2017) are a few examples.

Xander, the other targeted assembly tool we assessed, shows a reduction in running time which is comparable to the one achieved with S3A when up to a hundred domains are considered. However, in our evaluations, S3A showed better performances. In addition, one of Xander limitation is the specification of the profile models, which has to be done by the user, individually for each domain. This impairs the use of richer domain libraries, such as the one from MetaCLADE, which can greatly improve the annotation.

We should highlight that our strategy of graph construction is very general: any kind of string annotation can be provided as an input to our assembler. This could easily results in further improved running times, where the 'costly' step of HMM annotation (performed sequentially on all protigs), could be replaced by an efficient indexing and clustering of the protigs.

In a different path, the manner by which the Overlap-Layout graph is built allows multiple domain annotations, such that a unique graph is built for the whole range of domains. An improvement of our method would be to handle multiple overlapping domain annotations, such that every edge could hold weights for different domains. We believe that this would improve the sensitivity of the weight given to the graph edges and that are used during the graph traversal. For long protigs, it will also allow to reconstruct cases of domain co-occurrence. This is promising in a targeted assembly context, as it should both allow an even faster assembly, while permitting the detection of more precise functions, based on multiple domains.

A limitation of our approach is its dependency on the edge weights, which are used both for the graph pruning and for the traversal. While trying to hold a generic threshold independent on the protig length, the optimal threshold could vary depending on the type of species from which the reads were sequenced. In particular, the longest matching substring length metric is sensible to sequencing errors in the read tips. Applying an error correction tool before annotation, a practice common for *de novo* assembly, could improve the robustness of the *lmsl* and *ip* parameters for S3A. Moreover, while a smaller *lmsl* threshold can increase the number of sequence assemblies, it can also lead to increase incorrect domain identifications, due to the erroneous assembly of domain hits identified in unrelated sequences, highlighting a resulting function which,

in reality, is not present. However, results on different datasets/sequencing technologies show that S3A keeps a very high level of performance, even when default threshold values are changed.

Finally, it is worth mentioning that our data analysis highlights the importance in defining challenging datasets for critical assessments of metagenome interpretations that include new difficulties linked to domain length, by varying the proportion of domains with different lengths contained in the datasets.

Funding

This work was supported by Ministère de la Recherche et de l'Enseignement Supérieur (L.D.); LabEx CALSIMLAB (public grant ANR-11-LABX-0037-01 constituting a part of the 'Investissements d'Avenir' program—reference: ANR-11-IDEX-0004-02 to R.V.); the Institut Universitaire de France (A.C.).

Conflict of Interest: none declared.

References

Allen,E.E. and Banfield,J.F. (2005) Community genomics in microbial ecology and evolution. *Nat. Rev. Microbiol.*, **3**, 489–498.

Balzer,S. *et al.* (2010) Characteristics of 454 pyrosequencing data-enabling realistic simulation with flowsim. *Bioinformatics*, **26**, i420–i425.

Buelow,H.N. *et al.* (2016) Microbial community responses to increased water and organic matter in the arid soils of the McMurdo Dry Valleys, Antarctica. *Front. Microbiol.*, **7**, 1040.

Chikhi,R. and Rizk,G. (2013) Space-efficient and exact de Bruijn graph representation based on a bloom filter. *Algorithms Mol Biol.*, **8**, 22.

Chikhi,R. *et al.* (2016) Compacting de Bruijn graphs from sequencing data quickly and in low memory. *Bioinformatics*, **32**, i201–i208.

De Filippo,C. *et al.* (2012) Bioinformatic approaches for functional annotation and pathway inference in metagenomics data. *Brief. Bioinform.*, **13**, 696–710.

Eisen,J.A. (2007) Environmental shotgun sequencing: its potential and challenges for studying the hidden world of microbes. *PLoS Biol.*, **5**, e82.

Escobar-Zepeda,A. *et al.* (2015) The road to metagenomics: from microbiology to DNA sequencing technologies and bioinformatics. *Front. Genet.*, **6**, 348.

Finn,R.D. *et al.* (2011) HMMER web server: interactive sequence similarity searching. *Nucleic Acids Res.*, **39**, W29–W37.

Georganas,E. *et al.* (2018) Extreme scale de novo metagenome assembly. In: *SC18: International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE, pp. 122–134.

Gussow,A.B. *et al.* (2016) The intolerance to functional genetic variation of protein domains predicts the localization of pathogenic mutations within genes. *Genome Biol.*, **17**, 9.

Jia,B. *et al.* (2017) CARD 2017: expansion and model-centric curation of the comprehensive antibiotic resistance database. *Nucleic Acids Res.*, **45**, D566–D573.

Keegan,K.P. *et al.* (2016) MG-RAST, a metagenomics service for analysis of microbial community structure and function. In: Martin,F., Uroz,S. (eds.), *Microbial Environmental Genomics (MEG)*. Humana Press, New York, NY, pp. 207–233.

Rho,M. *et al.* (2010) FragGeneScan: predicting genes in short and error-prone reads. *Nucleic Acids Res.*, **38**, e191.

Richter,D.C. *et al.* (2008) MetaSim: a sequencing simulator for genomics and metagenomics. *PLoS One*, **3**, e3373.

Sczyrba,A. *et al.* (2017) Critical assessment of metagenome interpretation—a benchmark of metagenomics software. *Nat. Methods*, **14**, 1063–1071.

Tagliabue,A. *et al.* (2017) The integral role of iron in ocean biogeochemistry. *Nature*, **543**, 51–59.

Ugarte,A. *et al.* (2018) A multi-source domain annotation pipeline for quantitative metagenomic and metatranscriptomic functional profiling. *Microbiome*, **6**, 149.

Vital,M. *et al.* (2017) Colonic butyrate-producing communities in humans: an overview using omics data. *mSystems*, **2**, e00130–17.

Wang,Q. *et al.* (2015) Xander: employing a novel method for efficient gene-targeted metagenomic assembly. *Microbiome*, **3**, 32.

Wilke,A. *et al.* (2016) The MG-RAST metagenomics database and portal in 2015. *Nucleic Acids Res.*, **44**, D590–D594.

Zerbino,D.R. and Birney,E. (2008) Velvet: algorithms for de novo short read assembly using de Bruijn graphs. *Genome Res.*, **18**, 821–829.

Zhang,Y. *et al.* (2014) A scalable and accurate targeted gene assembly tool (SAT-assembler) for next-generation sequencing data. *PLoS Comput. Biol.*, **10**, e1003737.