MetaPar: Metagenomic Sequence Assembly via Iterative Reclassification

Minji Kim*, Jonathan G. Ligo*, Amin Emad, Farzad Farnoud (Hassanzadeh)
Olgica Milenkovic and Venugopal V. Veeravalli

*The first two authors contributed equally to this paper.

Department of Electrical and Computer Engineering, University of Illinois, Urbana-Champaign
Email: {mkim158,ligo2,emad2,hassanz1,milenkov,vvv} [at] illinois.edu

Abstract—We introduce a parallel algorithmic architecture for metagenomic sequence assembly, termed MetaPar, which allows for significant reductions in assembly time and consequently enables the processing of large genomic datasets on computers with low memory usage. The gist of the approach is to iteratively perform read (re)classification based on phylogenetic marker genes and assembler outputs generated from random subsets of metagenomic reads. Once a sufficiently accurate classification within genera is performed, de novo metagenomic assemblers (such as Velvet or IDBA-UD) or reference based assemblers may be used for contig construction. We analyze the performance of MetaPar on synthetic data consisting of 15 randomly chosen species from the NCBI database through the effective gap and effective coverage metrics.

I. INTRODUCTION

Metagenomics is a scientific discipline devoted to the study of complex microbial samples in the environment. Unlike classical genomics, where one is faced with the task of processing samples corresponding to one organism, metagenomics is concerned with samples that consist of a mixture of genetic material of different species and strains of bacteria or viruses within a host. With raw data file sizes capable of exceeding hundreds of gigabytes, metagenomic data poses significant new challenges in Big Data signal processing and analysis since genomic assembly is computationally difficult even for single species analysis. Although significant progress was made in the last few years on developing new methods for metagenome assembly, the problem of accurate metasequence profiling remains wide open. To hasten the progress of this new discipline, InnoCentive recently launched a special competition under the auspices of the U.S. Defense Threat Reduction Agency (DTRA) in metagenomic de novo assembly. Specialized access to powerful computers and clusters were given to all active participants for processing relatively small amounts of data (tens of gigabytes). In the absence of such strong computational support, large metagenomic assembly appears to be infeasible. One way to mitigate this issue is to break down the metagenomic data into smaller subsets that may be processed independently and in parallel, using modest computational power. This is the gist of the assembly approach MetaPar for parallel metagenomic assembly that we proceed to describe in the remainder of the paper.

MetaPar mitigates the need for computationally demanding full metagenomic assembly by using an iterative two-stage read classification technique. In the first stage, the microbial identification tool MetaPhyler is used for providing a rough profile of organisms present in the metagenomic mix. By aligning the reads to all genomes of organisms within the identified genera via Bowtie2, one obtains a rough partition of the reads into subgroups. Reads within different identified subgroup are assembled in parallel, producing contigs that may be run through BLAST (Basic Local Alignment Search Tool) to verify the accuracy of the classifier. After this first classification step, some reads may remain unaligned, and require alternative means of processing.

Two options may be pursued in the second round of classification, depending on the number of unaligned reads. If the number of reads is prohibitively large so as not to allow one-pass assembly with a standard metagenomic assembler, the reads are randomly partitioned into subgroups small enough to be assembled. All assemblies are performed in parallel. Unaligned reads are iteratively reassigned between assemblers until no changes in the assembled contigs are reported or until a maximum number of iterations is reached. On the other hand, if the number of reads is small enough to allow for one pass assembly, the same procedure as outlined for the initial step is performed. Related ideas involving dynamic classification of reads were described in [15], but for the purpose of single genome assembly, MetaPar is a simple parallel (and distributable) metagenomic assembler which is able to take advantage of improvements in standard de novo metagenomic assemblers and reference-based assembly, in contrast to recent distributed and parallel assemblers such as Ray Meta [21].

The paper is organized as follows. In Section II, we provide a step-by-step description of the MetaPar algorithm. In Section III, we demonstrate the performance of the method on synthetic Illumina sequencer data, using a randomly selected set of 15 bacterial organisms. We also compute and list the effective coverage and gaps in MetaPar alignments to the identified species’ genomes.

II. AN ALGORITHMIC SOLUTION FOR PARALLEL METAGENOMIC ASSEMBLY

The following terminology is used throughout the paper. When describing a living organism, we will refer to several taxonomy levels, listed from most general to most specific: life, domain, kingdom, phylum, class, order, family, genus, and species. Each organism has a genome, which is a sequence of bases over a four letter alphabet. A read is a substring of a genome or a chromosome (a part of a genome), generated through some sequencing system. The coverage of a base in a genome equals the number of reads that contain the base. Assembly refers to the process of overlapping reads – suffix to prefix – in order to reconstruct the original sequence from which the reads came from, or in order to reconstruct sufficiently long substrings of the genome, termed contigs. Alignment refers to mapping reads onto a given genome.

The metagenome assembly problem may be formulated as follows: given a mixture of reads from genomes of different species providing sufficiently high coverage, reconstruct the original genomes as accurately as possible via some computationally plausible assembly method. Many different assembly methods for metagenomic data were developed in the past few years, using greedy algorithms, reference-based approaches, algorithms based on deBruijn graphs and Eulerian path searches, and many other techniques. Although the accuracy of the aforementioned assembly methods is high for small metagenomic samples, it quickly deteriorates when the metagenomic data contains fragments of a large number of species. Even more important is the fact that the complexity of most assembly methods...
grows exponentially with the number of species, leading to poor performance scaling with sample size. As a result, large metagenomic samples require powerful computers for assembly. This raises the natural question of parallelizing the assembly process.

We next outline the parallel MetaPar assembly algorithm that allows for assembling metagenomes containing several hundred species suitable for commodity computers with 16-48+ GB RAM. The running time of all components, aside from calls to standard assemblers, grows linearly with the size of the metagenomic sample. The block diagram of the algorithm is depicted in Figure 1. Note that some steps in the algorithm are implemented only if the metasample (metagenomic sample) is very large, since in that case, direct assembly is computationally infeasible. The proposed algorithm uses certain techniques related to the authors' MCUIUC algorithm for compression, described in [20].

- **Step 1 (Level I Identification):** The first step of the iterative procedure is to remove as many reads that can be associated with known species from the original sample before running the assembly process. Such a filtering procedure is expected to produce significantly reduced metasamples for subsequent assembly. Filtering is achieved by passing the metasample through a taxonomic identifier, such as MetaPhyler [6]. The gist of the approach in [6] is that almost every genomic substring of length exceeding 20 is unique to a species or a genus. MetaPhyler scans through the reads to identify such substrings and links them to a sequenced species. Identification usually amounts to specifying the genus of organisms, and the abundances of the marker sequences. Due to identification errors, the output of the taxonomic classifier contains both false-positive and false-negative results. Selection of the identified genera used for read removal can depend on factors such as the number of identified organisms, genome lengths and identifier abundance. Simulation on synthetic data involving 15, 30 and ≥ 60 species was used to determine simple abundance thresholds, as described in the next section.

- **Step 2 (Level I Partitioning of the Dataset):**
  1) Once a group of genera of interest is identified in Step 1, representative reference genomes for alignment of reads are selected. Selection is accomplished by using complete genomes of all species within the identified genera in an alignment procedure performed via Bowtie2 [8]. Bowtie2 is designed for ultra-fast alignment of short reads to long genomes, and its complexity scales roughly linearly with the length of the genomes and number of reads (Bowtie2, along with BLAST [11], is one of the most frequently used alignment algorithms). Given that not all correct genera may have been found by MetaPhyler, some reads will be reported as unaligned. Unaligned in this context refers to not having sufficient sequence similarity to any substrings of the chosen references genomes. The percentage of unaligned reads heavily depends on the size of the metagenome, the number of species involved, as well as the number of identifiers of the species used in the identification software.
  2) **Step 3 (Level II Identification):** Given that MetaPhyler may miss identifying a large number of species present in the sample, and that consequently Bowtie2-based partitioning may leave a large fraction of reads unclassified, additional identification procedures are needed. Two different procedures are employed based on the volume of the unaligned reads. If the size of the unaligned metagenome is relatively small, metagenomic assembly based on Velvet, SOAP deNovo or IDBA-UD [8] is used. If the unaligned metagenome is prohibitively large to be assembled by existing assemblers, the unaligned read set is partitioned into an appropriate number of random subsets (The largest number of subsets we needed to run on any dataset was eight.). The subsets are processed in parallel by independent assemblers that produce contigs, which can be run through BLAST to identify additional reference genomes for alignment with Bowtie2. Given that the partitioning of the dataset is random, many reads may appear as stand-alone contigs at the output of the assembler, and are treated as unaligned reads that need to be re-classified [14].

3) **Step 4 (Iterative Re-classification):** Reads that remain unaligned after the described three steps are processed iteratively through Step 3, as long as the number of unaligned reads is higher than a certain threshold or until a maximum number of iterations is executed.

### III. Working Example

Metagenomic samples have vastly different sizes [10], and the exact number of iterations performed in the assembly process, as well as the exact number of parallel read classes used depends on the metagenomic file size. For example, for the “CO182: Coal cuttings from Coal bed Methane well site” sample [19], one has to run eight or more instances of a conventional assembler such as IDBA-UD – in parallel on several machines, or sequentially on one machine – with about 5.3 GB of reads per assembler. The algorithms were executed on computers equipped with dual Intel Xeon E5630 processors (16 threads) and 48 GB RAM in order to not exhaust the available memory at the 20 kmer level for the modified deBruijn graph search. This problem is further exacerbated with sequencing technologies that produce “long reads” (>128 bases in the case of IDBA-UD) which require additional data to be maintained for reads during assembly. In many cases, it is not feasible to increase available memory beyond a certain point per machine. Each required ~140 CPU hours to assemble, which while relatively low is not the critical constraint due to available memory. On the other hand, for most synthetic metagenomes including roughly 15 species, only one assembler is needed. All computations other than the running of parallel assemblers were performed on a computer equipped with an Intel Core i5 3470 and 16 GB of memory, and were primarily I/O and CPU limited rather than memory limited. In the former case (CO182), MetaPhyler only identified genera accountable for 40% of the metareads, while it identified more than 70% of reads in the small synthetic samples. Due to space limitations, we illustrate the performance and the steps of the MetaPar algorithm on a small synthetic sample involving 15 species, and defer the analysis of real metagenomic samples to the full version of the paper.

### A. Simulating the Metagenomic Sample

Species were randomly selected from the NCBI microbial genome database available at [18]. A selected group of 15 organisms is listed in Table I. Of the chosen species, Frankia has the longest genome with 5,511,253 bps (base pairs), while Mycoplasma arthritidis has the shortest genome with 832,175 bps. For each species, we selected the FASTA file containing the complete genome and generated paired

---

**TABLE I. A RANDOMLY SELECTED SET OF 15 SPECIES USED TO ILLUSTRATE THE OPERATING PRINCIPLES OF MetaPar.**

| # | Genus | Species |
|---|---|---|
| 1 | Acidiphilus | Acidiphilus saccarivorans 335_15uid51395 |
| 2 | Aclavimonas | Aclavimonas berkenhemii SSK_15uid51109 |
| 3 | Bacteroides | Bacteroidesfragilis VCH46_uid58159 |
| 4 | Borrelia | Borrelia_garrini NMW1_uid177081 |
| 5 | Corynebacterium | Corynebacterium pseudotuberculosis TV_uid1597673 |
| 6 | Enterobacter | Enterobacter_suboxytoca F0289_uid51793 |
| 7 | Frankia | Frankia_CCU3.uid58597 |
| 8 | Halomicrobium | Halomicrobiun_mukohataei JMM_12256_uid59101 |
| 9 | Helicobacter | Helicobacter_pylori S143_uid62205 |
| 10 | Lactobacillus | Lactobacillus_anamiosus CCM1189_uid4076253 |
| 11 | Mobiluncus | Mobiluncus curtisi ATCC 43063_uid69069 |
| 12 | Mycoplasma | Mycoplasma_arthritidis 158f31_uid58005 |
| 13 | Odoribacter | Odoribacter_pseudophilum JMM2072_uid685979 |
| 14 | Prevotella | Prevotella_dentium F0289_uid58597 |
| 15 | Psychrotroplus | Psychrotroplus_lusus_AFCC000755_uid54205 |

---

1 Very recently, a new approach to species identification was described in [2] that may outperform MetaPhyler. Given that comparing identification software packages is beyond the scope of the paper, we only report results for the more commonly used MetaPhyler package.

2 As already mentioned, a means for parallelizing single genome assembly that shares some of the classification ideas outlined in this step was first reported in [15].
reads using the sim_reads tool accompanying IDBA-UD with settings as in [8], and coverage depth 100. The reads from each species were combined to simulate a metagenomic sample from an Illumina sequencer without quality score information (nevertheless, the algorithm can be easily adapted to include quality information as well, and reads stored in the FASTQ format). The resulting metagenomic sample size was 5 GB in the FASTA format. Note that the chosen synthetic sample, unlike real metagenomic data, had no species from the same genus. How the performance of the algorithm changes in the presence of multiple species per genera will be described in the next subsection.

B. Step 1: Metaphyler Genus Identification

MetaPhyler takes the simulated metagenomic reads as inputs and outputs their taxonomy classifications. We focused our attention on genus classification, as it is the finest reliable level provided by MetaPhyler. The MetaPhyler genus-level output for the input metagenomic reads is given in Fig. 2 Note that each genus identifier appears in the table with the number of reads containing markers and the abundance level of such reads. We only selected genera with abundance or number of reads exceeding a certain threshold. The choice of the threshold is governed by many parameters, including the number of estimated organisms, their genome lengths, the number of known markers in the genomes, as well as the actual output of MetaPhyler. As a guideline, we used the threshold criteria listed in Table II. Of the 28 genera identified, 11 satisfied the threshold criteria, which in this case amounted to more than 1% abundance or at least 1000 reads. The selected set of 11 genera contains all true positives and no false positives. However, MetaPhyler missed identifying the genera of four organisms present in the metagenomic sample, namely Acidilobus, Halomicrobium, Odoribacter, and Psychrophilus.

Species within the metagenomic mixture were identified through an additional procedure described in the next subsection.

TABLE II. A THRESHOLD CRITERIA TO SELECT TOP GENERA OF MetaPhyler OUTPUT.

| # of reported species by MetaPhyler | Abundance (%) | # Reads |
|-------------------------------------|--------------|--------|
| > 35; < 60                          | 0.3          | N/A    |
| ≥ 60                                | ≥ 1000       | ≥ 1000 |

C. Step 2: Read Classification

For the purpose of classifying the reads based on similarity to the selected reference genomes, we used the Bowtie2 algorithm [3]. Using all species of the 11 chosen genera above and building a Bowtie2 index provided a very good metagenomic read alignment rate, equal to 70.29%. Approximately 30% of the reads were not aligned to any reference genomes, so these unaligned reads were assembled via IDBA-UD. The longest 30 resulting contigs were passed through BLAST. BLAST identified 15 of the contigs as Odoribacter splanchicus, 8 as Acidilobus saccharovorans, 6 as Psychrophilus torquis, and 1 as Halomicrobium mukohataei, which were exactly the four species missed by MetaPhyler in Step 1. These 4 species and the 25 species listed in Table II were used as reference genomes for the second iteration of Bowtie2, and the read alignment rate was 99.94%.

D. Assembler performance evaluation

One of the most commonly used statistics for assessing the performance of an assembler is the N50 statistic on contig lengths. The N50 statistic is a threshold value for the length, such that contigs of length longer than or equal to the threshold account for roughly 50% of the total contig length found by the assembler. In other words, it is helpful to think of the N50 parameter as the median of the contig length distribution. Since multiple lengths may satisfy this criteria, the N50 value is often chosen to be the average of all thresholds that satisfy the terms of the definition.
The utility of the N50 value for assessing assembler performance is questionable, since it does not convey important information about what percentage of the length of underlying genomes is actually covered by the contigs and to what extent. This is especially true for reference based assembly. To mitigate this problem, we introduced two performance measures, termed the reference based assembly. To reduce the amount of standard de novo assembly required. Performance was illustrated on a synthetic sample of 15 species. Further work includes designing schemes for efficient partitioning of reads akin to TIGER, incorporation of phylogenetic aligners and incorporation of other classifiers for reads.

### Acknowledgment

This work was supported in part by NSF grants CCF 0809895, CCF 1218764, Emerging Frontiers for Science of Information Center, CCF 093970 and U.S. Defense Threat Reduction Agency through subcontract 147755 at the University of Illinois from prime award HDTRA1-10-1-0086. The authors also gratefully acknowledge many useful discussions with Prof. Jian Ma and Xiaolong Wu at the University of Illinois, Urbana-Champaign.

### References

[1] L. Altschul et al. Basic local alignment search tool, Journal of Molecular Biology, 1990.

[2] O. Francis et al., “Pathoscope: Species identification and strain attribution with unassembled sequencing data,” Genome Research, published in Advance July 10, 2013.

[3] B. Laggerbauer and S.I. Salzberg, Fast gapped-read alignment with Bowtie 2, Nature Methods, 2012.

[4] R. Li et al. “De novo assembly of human genomes with massively parallel short read sequencing,” Genome research, 20.2, pp. 265-272, 2010.

[5] W-T. Liu et al., “Characterization of microbial diversity by determining terminal restriction fragment length polymorphisms of genes encoding 16S rRNA,” Applied and environmental microbiology, 65, no. 11, 4516-4522, 1997.

[6] B. Liu et al., “MetaPhyler: Taxonomic profiling for metagenomic sequences,” 2010 IEEE International Conference on Bioinformatics and Biomedicine (BIBM), 2010.

[7] K. Mavromatis et al. “Use of simulated data sets to evaluate the fidelity of metagenomic processing methods.” Nature methods, 4.6, pp. 495-500, 2007.

[8] Y. Peng et al. IDBA-UD: a de novo assembler for single-cell and metagenomic sequencing data with highly uneven depth, Nat Methods, 2012.

[9] P.A. Pevzner, H. Tang, and M.S. Waterman, An Eulerian path approach to DNA fragment assembly, Proc Natl Acad Sci, USA, 2001.

[10] J. Qin et al., “A human gut microbial gene catalogue established by metagenomic sequencing.” Nature 464, no. 7285, pp. 59-65, 2010.

[11] C. Riesenfeld, P. D. Schloss, and J. Handelsman, “Metagenomics: genomic analysis of microbial communities,” Annu. Rev. Genet., 38, pp. 525-552, 2004.

[12] M-F Sagot, “Spelling approximate repeated or common motifs using a suffix tree,” In LATIN’98: Theoretical Informatics, pp. 525-552, 2004.

[13] N. Siva, “1000 Genomes project,” Nature Biotechnology, 26, no. 3, pp. 256-256, 2008.

[14] D. Zerbino and E. Birney. “Velvet: algorithms for de novo short read assembly using de Bruijn graphs.” Genome research, 18.5, pp. 821-829, 2008.

[15] X-L. Wu et al., “TIGER: tiled iterative genome assembler,” BMC bioinformatics, 13, no. Suppl 19, S18, 2012.

[16] H. Li et al. and 1000 Genome Project Data Processing Subgroup, “The Sequence alignment/map (SAM) format and SAMtools”, Bioinformatics, vol. 25, pp. 2078-9, 2009.

[17] Illumina, Inc. (2013) “Performance and Specifications for HiSeq 2500/1500” [Online]. Available:http://www.illumina.com/systems/hiseq_2500_1500/performance_specifications.nlm

[18] ftp://ftp.ncbi.nlm.nih.gov/Genomes/Bacteria/all.fna.tar.gz

[19] http://www.ncbi.nlm.nih.gov/sra/SRX211003

[20] J. G. Ligo, M. Kim et al. “MCUIUC – A New Framework for Metagenomic Read Compression,” To appear in 2013 IEEE Information Theory Workshop, Seville, Spain, 2013.

[21] S. Boisvert et al., “Ray Meta: scalable de novo metagenome assembly and profiling”, Genome Biology, 13:R122, 2012.