Identifying Core Operons in Metagenomic Data

Xiao Hu[1], Iddo Friedberg[1,2,*],

[1]Department of Veterinary Microbiology and Preventive Medicine, Iowa State University, Ames, IA 50010, USA [2]Program in Bioinformatics and Computational Biology, Ames, IA 50010 USA

[*]idoerg@iastate.edu

Abstract

An operon is a functional unit of DNA whose genes are co-transcribed on polycistronic mRNA, in a co-regulated fashion. Operons are a powerful mechanism of introducing functional complexity in bacteria, and are therefore of interest in microbial genetics, physiology, biochemistry, and evolution. While several methods have been developed to identify operons in whole genomes, there are few that can identify them in metagenomes. Here we present a **P**ipeline for **O**peron **E**xploration in **M**etagenomes or POEM. At the heart of POEM lies a neural network that classifies genes as intra- or extra-operonic. POEM then looks for proximity associations between identified intra-operonic genes, to identify *core operons* in the metagenome. Core operons are operons that may exist in more than one species in the metagenome, and being evolutionarily conserved increases the probability of accurate prediction. We tested POEM using several different assemblers on a simulated metagenome, and we show it to be highly accurate. We also demonstrate its use on a human gut metagenome sample, and discover a putative new operon. We conclude that POEM is a useful tool for analyzing metagenomes beyond the genomic level, and for identifying multi-gene functionalities and possible neofunctionalization in metagenomes

Introduction

It is estimated that 5-50% of bacterial genes reside in operons [1, 2], and the characterization and understanding of operons is central to bacterial genomic studies. Experimental approaches, chiefly RNA-Seq, are the most reliable way to identify operons; however, it is not feasible to perform experiments to characterize all operons. Over the years, several computational operon-prediction techniques have been developed. Generally, computational operon identification methods include three steps: 1. identify genes that are in an operon and, conversely, genes that do not participate in an operon; 2. identify features typical of each group; 3. train a classifier with these features and build a discriminating model.

Computational operon prediction methods have been developed since the late 1990's (For a comprehensive review see: [3].) Naïve Bayes models have been used since early 2000's for predicting operons [4–6]. Another method used microarray data to identify the different expression profiles of adjacent gene pairs in operons and outside of operons. The differential expression profiles and intergenic distances were used as as features to train a Bayesian classifier [7]. Comparative genomic methods were also used to identify operons by detecting conserved gene clusters across several species [8–10]. Other methods include particle swarm optimization [11, 12], and neural networks [13].

There are several operon databases that include automated and experimental-based operon annotation [14–18]. However, a manual curation method is not suitable for the rapid growing number of bacterial genomes, few of which are experimentally assayed for operons. Also, experimental studies tend to use data from model species, and cross-species prediction may not work well [19].

The challenge of discovering operons is compounded when trying to discover operons in metagenomic data. Major additional confounders include the large loss of genetic information, short contigs that rarely assemble into a full genome, and misassembly that might produce chimeric contigs [20]. At the same time, metagenomic data contain rich information that cannot be gleaned from clonal cultures; it is therefore necessary to investigate how well we can predict operons in metagenomic data. Some work has been done including use of proximity and guilt-by-association [21, 22].

While a genome contains the total genetic information of an organism, a metagenome is a partial snapshot of a population of genomes. We therefore can rarely expect an operon discovery method to provide the entire content of operons from metagenomic data. However, predicting whether genes participate in an operon provides valuable additional information to the functional annotation of a metagenome. In this study we present a method that (1) classifies genes in metagenomes into "operonic" and "non-operonic" classes and (2) provides

functional annotations for the operons it reconstructs from metagenomic data. To this end, we introduce the concept of metagenomic *core operons*. A core operon comprises a set of operonic genes that may have orthologs in several species in the metagenome. Commonly, metagenomic analysis pipelines provide the distribution of biological function the metagenome has based on a normalized count of functionally-annotated ORFs. Our method, a Pipeline for Operon Exploration in Metagenomes or POEM, adds more information as it considers the evolutionary conservation of co-transcribed genes in the species constituting the microbial community. This additional information is valuable for understanding the genetic potential of a microbial community based on its metagenomic information.

Materials and Methods

An overview of the POEM pipeline is shown in Fig. 1. The heart of the pipeline lie the Operon identification and operon core structure that POEM performs. The other steps are performed with third-party tools, and are modular. Below we elaborate upon the varisou stages in the POEM pipeline.

[Figure 1 about here.]

Metagenome assembly

POEM uses the IDBA-UD *de-novo* assembler, but the user may supply their own assembler. Short read assemblers are usually based on De Bruijn graphs and are sensitive to the sequencing depth, repetitive regions, and sequencing errors [23]. For clonal bacteria, this assembly algorithm works relatively because it is easy to estimate the sequencing depth and the bacterial genomes are often compact and have few repetitive regions. However, in metagenomes it is hard to estimate the amount of sequence data that are needed for good functional coverage, and the genomes from closely related species may contain many highly conserved genes which may be interpreted as repetitive regions. Although *de novo* assemblers for

metagenomes are still at an early stage [24], there are several tools developed for this task including MetaVelvet-SL, IDBA-UD,and Megahit [25–29]. In this study we also compare the effect these assemblers have on the accuracy of POEM.

Gene prediction

We chose to use an *ab-initio* method for gene calling, as opposed to calling by sequence similarity. First, because *ab-initio* gene calling is faster in bacterial and archaeal genomes, with little accuracy sacrificed: the predicted accuracy of some methods can reach 98% [30–33]. Second, metagenomic data contain many genes with no similarity to known genes, so using a homology based method may result in a large number of open reading frames (ORFs) that are not predicted as such (false negatives). Recently, a series of gene prediction tools have been developed or optimized for metagenomic data, including Glimmer-MG, Metagene, Metagenemark, Prokka, Prodigal, and Orphelia [31, 33–37]. POEM uses Metagenemark or Prokka to predict genes. As in the contig assembly stage, this part can be modified by the user.

Removing ORF redundancies

Once ORFs are identified, we remove redundant ORFs with an ID of >98% using CD-HIT [38, 39]. The assumption is that genes with a very high sequence ID were taken from the same species or highly similar strains and are therefore redundant information.

Gene function annotation

While there are many ways to annotate gene function [40], a fast and acceptably accurate way to do so typically employs sequence similarity matching against a reliable functionally annotated sequence database. Here we used the COG database as a reference. POEM uses both BLAST and DIAMOND [41], which

trades off speed and sensitivity. The functional assignment is done by choosing the top hit in COG above the e-value threshold ($Evalue = 10^{-3}$).

Operon prediction

At the core of POEM lies a novel method we developed for predicting operons. POEM predicts if any given pair of adjacent genes are intra-operonic by classifying intergenic regions into intra- or extra-operonic. Thus, the operon prediction problem is cast as a binary classification problem.

The operon prediction method goes through the following steps. First, the intergenic DNA sequences of 4,425 operonic and 2,097 non-operonic adjacent genes are extracted from Operon DataBase v2 [16]. The intergenic regions are represented as a k-mer-position matrix (KPM, Figure 2). Two-thirds of the data were used for training a Convolutional Neural Network (CNN) based binary classification model and the remaining 1/3 of the data were used as the test set. We used a CNN model from the Keras package (v1.2.0) to train the classification model [42]. Since the CNN only accepts a fixed size matrix, we convert the KPM to a fixed size matrix by truncating the middle columns or adding all zero columns to the middle of the matrix. Trial-and-error has shown that $k = 3$ produced the best accuracy (Supplementary Figure S1).

To show the CNN's utility, we compared its performance to a simple baseline predictor. The baseline classifier works as follows: if two genes on the same strand have an intergenic distance < 500 nt, then their adjacency is classified as within the same operon (operonic). A larger distance would classify them as non-operonic. The predicted operonic adjacent genes were then connected to form a full operon prediction.

[Figure 2 about here.]

Identifying core operon functionality

To identify operons in metagenomes, we introduce the concept of *core operons*. Core operons are weighted-edge graphs that capture information about predicted orthologous operons or fractions of operons in the metagenome. Each node is a set of orthologous genes that are all annotated by at least one common COG term. The weight of the edge is determined by the frequency of the adjacency of the intra-operon adjacent genes. To determine how well a core operon captures the real operons in a metagenome, we ran a precision-recall analysis using the operons in the simulated database as our standard-of-truth, see Figure 3. Here, precision is the number of correctly predicted intra-operonic genes (true positives) divided by the number of all predictions (true positive and false positive predictions). Recall is the number of correctly predicted intra-operonic genes divided by the all real intra-operonic genes. POEM produces a file that can then be used by Cytoscape [43] to visualize the core operons.

[Figure 3 about here.]

Results

We ran POEM on two different data sets. One comprises simulated reads generated by ART [44] from 48 genomes of Operon DataBase v2 [16]. The genome species used and parameters of ART are shown in Supplemetary Table S1. The second set used is SRR2155174 downloaded from ENA [45]. As a standard of truth, we selected operons from Operon DataBase v2 that are supported by literature. This dataset contains 8,194 genes and 5,621 adjacent genes in 2,589 operons. These standard-of-truth operons are *true operons* (TOs). TO genes are genes within real operons, and adjacent genes in a TO are TA genes.

	Simulated Reads			SRR2155174			
Feature\Assembler	IDBA-UD	Megahit	Velvet_51	IDBA-UD	Megahit	Velvet_51	Assembly
Size (bp)	132,218,137	134,341,573	100,889,182	131,424,989	135,150,882	85,261,552	122,371,235
GC content	49.51%	49.55%	50.73%	48.11%	48.19%	46.96%	48.04%
Number of contigs	48,508	54,274	61,093	87,992	107,718	146,313	55,925
Max contig length	947,260	549,191	569,707	484,034	249,170	106,439	327,893
Min contig length	100	200	101	100	200	101	500
Mean contig length	2,725	2,475	1,651	1,493	1,254	582	2,188
N50	12,732	7,681	9,312	4,306	2,593	906	4,331
N90	984	888	569	478	426	227	754
Protein-coding gene	154,908	162,282	133,298	175,983	190,946	170,100	147,873

Table 1. Main features of simulated and real metagenome assembly.

Metagenome assembly

We used IDBA-UD, MegaHIT, and Velvet [46] to assemble the simulated and experimental reads; the results are shown in Table 1. IDBA-UD provided the maximal N50 and minimal number of contigs in both datasets. MegaHIT provided the largest genome size and the most protein-coding genes.

Gene prediction is an upper limit to operon prediction

Metagenemark found 7,855 genes of the 8,194 true operon genes in the whole genomes. For the assembly, the gene numbers are 5,116, 5,078, and 3,530 for IDBA, Megahit, and Velvet respectively. These results show a positive correlation between the assembly's N50 and and metagenemark's performance.

Operon prediction and adjacent genes within the operon

We tested the operon prediction module's performance on whole genomes and simulated metagenome assembly. The 4,425 operonic and 2,097 non-operonic adjacent genes mentioned above were used as a True Positive (TP) set and True Negative (TN) set, respectively. The precision, recall, and F_1 for predicted operonic adjacency are defined in the following equations and the statistical results are shown in Table 2.

$$Precision = \frac{True\ Positives}{True\ Positives + False\ Positives}$$

$$Recall = \frac{True\ Positives}{True\ Positives + False\ Negatives}$$

Genome	Whole genome		IDBA_ud		Megahit		Velvet	
Predictor	CNN	Linear	CNN	Linear	CNN	Linear	CNN	Linear
Precision(%)	89.75	69.84	58.74	54.69	57.61	53.56	46.75	46.80
Recall(%)	89.04	98.73	51.05	55.91	48.84	53.42	37.60	41.31
F1 score(%)	89.39	81.81	55.13	54.62	53.40	52.86	41.68	43.88

Table 2. Evaluation of operonic adjacency prediction on simulated metagenomes. **CNN**: a convolutional neural network based classifier; **Linear**: a linear classifier that is based on the intergenic distance and strand.

Real Operon	**Whole genome**		**IDBA**		**Megahit**		**Velvet**	
Classifier	**CNN**	**Linear**	**CNN**	**Linear**	**CNN**	**Linear**	**CNN**	**Linear**
Predicted operon	1,997	1,816	1,232	1,322	1,332	1,377	914	881
Perfect recovery	1,025	469	526	253	470	233	360	160

Table 3. Predicting operons in whole genome and assemblies of simulated metagenomes. **Predicted operon**: $\geq 60\%$ genes in a predicted operon belong to a known operon; **Perfect recovery**: both precision and recall equal one.

$$F_1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

However, these results only reflect POEM's performance on classification of operonic and non-operonic adjacency. To further evaluate POEM's performance on full operon prediction, we report on the precision / recall analysis as illustrated in Figure 4. The total number of true operons in the simulated database was determined to be 2,589. The results are shown in Table 3.

[Figure 4 about here.]

Core Functions Facilitated by Predicted Operons in Metagenomic Data

To show the utility of our method in discovering core functions facilitated by predicted operons, we ran POEM on the metagenome sample SRR2155174, containing the human gut microbiome data. Figure 5A shows a core function predicted from the SRR2155174 data set. The annotations of the core function (Figure 5A) indicates that it is related to lipid transport and metabolism. We found several predicted operons (Figure 5B-E) from the SRR2155174 data set that match the core function. Of the loci in the core operon, only lp_1674 and lp_1675 loci in *Lactobacillus plantarum* WCFS1 (Figure 5E) can be found in

Assembler	**Number of core operons**	**Intersection with True Operons**	**Mean Precision ± SE**	**Mean Recall ± SE**	**Mean F_1 ± SE**
True Operon					
NA	110	NA	0.97 ± 0.12	0.66 ± 0.28	0.75 ± 0.22
Genome					
NA	310	36	0.77 ± 0.34	0.60 ± 0.34	0.67 ± 0.28
Simulated Reads					
IDBA_ud	260	48	0.83 ± 0.31	0.61 ± 0.33	0.71 ± 0.26
Megahit	256	46	0.84 ± 0.31	0.61 ± 0.32	0.71 ± 0.26
Velvet	202	56	0.87 ± 0.30	0.59 ± 0.32	0.71 ± 0.25
SRR2155174					
IDBA_ud	141	25	0.71 ± 0.36	0.55 ± 0.33	0.65 ± 0.24
Megahit	138	26	0.71 ± 0.37	0.54 ± 0.33	0.66 ± 0.24
Velvet	94	11	0.72 ± 0.39	0.48 ± 0.33	0.65 ± 0.25

Table 4. Comparisons of core function. **Intersection with True Operons**: Number of identical core functions shared between tru. **SE**: standard error.

the predicted operons of Operon DataBase [16]. To find the functions of these predicted operonic genes, we looked at the functional annotations for these operonic genes from GenBank [47]. The functional annotations (Supplementary Table S2) show that these operonic genes are likely to be involved in fatty acid biosynthesis. We mapped the predicted operonic genes of *Lactobacillus plantarum* WCFS1 (Figure 5E) to KEGG database [48] and found most of the genes involved in

predicted operons are likely involved in fatty acid biosynthesis and have a high probability of being true operons. Although these operons are involved in the same biological pathway, the genes outside the core function (Figure 5B-E) are diverse. The core function reflects the conservation of operons across species and is more robust and error-tolerant than operons. Core functions may reconstruct the metabolism pathways from the incomplete genome assembly data leveraging the conservation of genes across species. The ability to use core functions as familiar ground from which to explore new conserved proximal genes makes core functions a new and powerful tool for discovering novel operon-encoded pathways in metagenomic sequence data.

[Figure 5 about here.]

Discussion

In this study we we introduce a novel method for predicting operons in genomic and metagenomic data. Our method uses a k-mer based classification of intra-operonic and inter-operonic genes. We then apply our method as a pipeline to the sequence data. Generally we obtain a high precision rate and a lower recall rate on genome and simulated genome data. This means that we are correctly recovering intra-operonic genes and maintaining a low false-positive rate. At the same time, we are recovering a lower fraction of the genes, and the false-negative rate is 33% when recovering operon genes in whole genomes. When recovering operons from metagenomes, our results depend upon the choice of gene-calling software and metagenome assembly. We see the highest precision results using from assembly using Velvet, while using Megahit assemblies produce marginally lower precision but with higher recall in the simulated reads.

In the operonic adjacency prediction step, we use 4,425 literature supported operonic and 2,097 non-operonic adjacency as positive and negative samples. For the 48 whole genome, the CNN classifier has achieved precision of 90.26% and recall of 90.71% on 6,522 operonic and non-operonic adjacency. For the assembly, the precision increase a little, however, the recall decreased significantly. These results consistent with the conclusion from gene prediction step.

We compared CNN with the linear model on whole genome and assembly. The CNN outperforms the linear model, especially the recall for the assembly which has lower quality. These results indicate that our CNN based classifier is a reliable method for the operonic adjacency prediction.

We also evaluate the performance of full operon prediction. This classifier can identify 42.56% of 2,589 literature supported full operons on the 48 whole genomes. However, for the assembly, the number of correct predicted full operon decreases below 14.95%. These results indicate that improving the assembly quality for metagenomic data will significantly improve the precision and recall

for operon prediction step.

Finally, the short contigs typically resulting from metagenome assembly do not contain full operon information. However, we can still extract some information on probable intra-operonic genes and operons structure using POEM. POEM is therefore a useful addition to the arsenal of tools helping us to better understand the functionality of metagenomes.

Availability of source code and requirements

The software and related information are listed below: **Project Name:** POEM

Project Home Page: `https://github.com/Rinoahu/POEM_py3k`

Operating System(s): POEM was tested on GNU/Linux distribution Ubuntu 16.04 64-bit, but we expect POEM to work on most *nix systems

Programming Language: Python

Other Requirements: Python 3.7 and Conda

License: GPLv3

Acknowledgments

Will be provided upon acceptance

References

1. Cherry JL. Genome size and operon content. Journal of theoretical biology. 2003;221(3):401–410.

2. Wolf YI, Rogozin IB, Kondrashov AS, Koonin EV. Genome alignment, evolution of prokaryotic genome organization, and prediction of gene function using genomic context. Genome research. 2001;11(3):356–372.

3. Zaidi SSAS, Zhang X. Computational operon prediction in whole-genomes and metagenomes. Briefings in functional genomics. 2016 Sep;Available from: `http://view.ncbi.nlm.nih.gov/pubmed/27659221`.

4. Craven M, Page D, Shavlik J, Bockhorst J, Glasner J. A Probabilistic Learning Approach to Whole-Genome Operon Prediction. Proc Int Conf Intell Syst Mol Biol. 2000;8:116–27.

5. Bockhorst J, Craven M, Page D, Shavlik J, Glasner J. A Bayesian network approach to operon prediction. Bioinformatics. 2003 jul;19(10):1227–1235. Available from: `http://bioinformatics.oxfordjournals.org/cgi/doi/10.1093/bioinformatics/btg147`.

6. Jacob E, Sasikumar R, Nair KNR. A fuzzy guided genetic algorithm for operon prediction. Bioinformatics. 2005 apr;21(8):1403–1407. Available from: `http://bioinformatics.oxfordjournals.org/cgi/doi/10.1093/bioinformatics/bti156`.

7. Sabatti C, Rohlin L, Oh MK, Liao JC. Co-expression pattern from DNA microarray experiments as a tool for operon prediction. Nucleic Acids Res. 2002 jul;30(13):2886–2893. Available from: `http://nar.oxfordjournals.org/lookup/doi/10.1093/nar/gkf388`.

8. Overbeek R, Fonstein M, D'Souza M, Pusch GD, Maltsev N. The use of gene clusters to infer functional coupling. Proceedings of the

National Academy of Sciences of the United States of America. 1999 Mar;96(6):2896–2901. Available from: http://www.pnas.org/content/96/6/2896.abstract.

9. Chen X, Su Z, Dam P, Palenik B, Xu Y, Jiang T. Operon prediction by comparative genomics: an application to the Synechococcus sp. WH8102 genome. Nucleic Acids Res. 2004;32(7):2147–57. Available from: http://www.ncbi.nlm.nih.gov/pubmed/15096577http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC407844.

10. Moreno-Hagelsieb G, Collado-Vides J. A powerful non-homology method for the prediction of operons in prokaryotes. Bioinformatics. 2002;18 Suppl 1:S329–36. Available from: http://www.ncbi.nlm.nih.gov/pubmed/12169563.

11. Chuang LY, Tsai JH, Yang CH. Binary particle swarm optimization for operon prediction. Nucleic Acids Res. 2010 jul;38(12):e128. Available from: http://www.ncbi.nlm.nih.gov/pubmed/20385582http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC2896535.

12. Chuang LY, Yang CH, Tsai JH, Yang CH. Operon Prediction Using Chaos Embedded Particle Swarm Optimization. IEEE/ACM Trans Comput Biol Bioinforma. 2013 sep;10(5):1299–1309. Available from: http://ieeexplore.ieee.org/document/6517843/.

13. Taboada B, Verde C, Merino E. High accuracy operon prediction method based on STRING database scores. Nucleic Acids Res. 2010 jul;38(12):e130. Available from: http://www.ncbi.nlm.nih.gov/pubmed/20385580http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC2896540.

14. Gama-Castro S, Salgado H, Santos-Zavaleta A, Ledezma-Tejeida D, Muñiz-Rascado L, García-Sotelo JS, et al. RegulonDB version 9.0: high-level

integration of gene regulation, coexpression, motif clustering and beyond. Nucleic acids research. 2016;44(D1):D133–D143.

15. Pertea M, Ayanbule K, Smedinghoff M, Salzberg SL. OperonDB: a comprehensive database of predicted operons in microbial genomes. Nucleic acids research. 2009;37(suppl 1):D479–D482.

16. Okuda S, Yoshizawa AC. ODB: a database for operon organizations, 2011 update. Nucleic acids research. 2011;39(suppl 1):D552–D555.

17. Mao F, Dam P, Chou J, Olman V, Xu Y. DOOR: a database for prokaryotic operons. Nucleic acids research. 2009;37(suppl 1):D459–D463.

18. Taboada B, Ciria R, Martinez-Guerrero CE, Merino E. ProOpDB: prokaryotic operon database. Nucleic acids research. 2012;40(D1):D627–D631.

19. Dam P, Olman V, Harris K, Su Z, Xu Y. Operon prediction using both genome-specific and general genomic information. Nucleic Acids Res. 2007;35(1):288–298.

20. Wooley JC, Godzik A, Friedberg I. A Primer on Metagenomics. PLoS Comput Biol. 2010 feb;6(2):e1000667. Available from: `http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=2829047{&}tool=pmcentrez{&}rendertype=abstract`.

21. Vey G, Moreno-Hagelsieb G. Metagenomic annotation networks: Construction and applications. PLoS One. 2012;7(8).

22. Vey G, Charles TC. MetaProx: the database of metagenomic proximons. Database : the journal of biological databases and curation. 2014;2014. Available from: `http://view.ncbi.nlm.nih.gov/pubmed/25288655`.

23. Li Z, Chen Y, Mu D, Yuan J, Shi Y, Zhang H, et al. Comparison of the two major classes of assembly algorithms: overlap-layout-consensus and

de-bruijn-graph. Brief Funct Genomics. 2012 jan;11(1):25–37. Available from: http://www.ncbi.nlm.nih.gov/pubmed/22184334.

24. Thomas T, Gilbert J, Meyer F. Metagenomics - a guide from sampling to data analysis. Microbial Informatics and Experimentation. 2012 Feb;2(1):3. Available from: https://doi.org/10.1186/2042-5783-2-3.

25. Oulas A, Pavloudi C, Polymenakou P, Pavlopoulos GA, Papanikolaou N, Kotoulas G, et al. Metagenomics: tools and insights for analyzing next-generation sequencing data derived from biodiversity studies. Bioinform Biol Insights. 2015;9:75–88. Available from: http://www.ncbi.nlm.nih.gov/pubmed/25983555http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC4426941.

26. Namiki T, Hachiya T, Tanaka H, Sakakibara Y. MetaVelvet: An extension of Velvet assembler to de novo metagenome assembly from short sequence reads. Nucleic Acids Res. 2012;40(20).

27. Afiahayati, Sato K, Sakakibara Y. MetaVelvet-SL: An extension of the Velvet assembler to a de novo metagenomic assembler utilizing supervised learning. DNA Res. 2015;22(1):69–77.

28. Peng Y, Leung HCM, Yiu SM, Chin FYL. IDBA-UD: A de novo assembler for single-cell and metagenomic sequencing data with highly uneven depth. Bioinformatics. 2012;28(11):1420–1428.

29. Li D, Liu CM, Luo R, Sadakane K, Lam TW. MEGAHIT: An ultra-fast single-node solution for large and complex metagenomics assembly via succinct de Bruijn graph. Bioinformatics. 2014;31(10):1674–1676.

30. Wang Z, Chen Y, Li Y. A brief review of computational gene prediction methods. Genomics, proteomics Bioinforma / Beijing Genomics Inst. 2004;2(4):216–21. Available from: http://www.ncbi.nlm.nih.gov/pubmed/15901250.

31. Zhu W, Lomsadze A, Borodovsky M. Ab initio gene identification in metagenomic sequences. Nucleic Acids Res. 2010;38(12).

32. Hyatt D, Chen GL, Locascio PF, Land ML, Larimer FW, Hauser LJ. Prodigal: prokaryotic gene recognition and translation initiation site identification. BMC Bioinformatics. 2010;11:119. Available from: `http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=2848648{&}tool=pmcentrez{&}rendertype=abstract`.

33. Hyatt D, Locascio PF, Hauser LJ, Uberbacher EC. Gene and translation initiation site prediction in metagenomic sequences. Bioinformatics. 2012;28(17):2223–2230.

34. Kelley DR, Liu B, Delcher AL, Pop M, Salzberg SL. Gene prediction with Glimmer for metagenomic sequences augmented by classification and clustering. Nucleic Acids Res. 2012;40(1).

35. Noguchi H, Park J, Takagi T. MetaGene: Prokaryotic gene finding from environmental genome shotgun sequences. Nucleic Acids Res. 2006;34(19):5623–5630.

36. Seemann T. Prokka: Rapid prokaryotic genome annotation. Bioinformatics. 2014;.

37. Hoff KJ, Lingner T, Meinicke P, Tech M. Orphelia: Predicting genes in metagenomic sequencing reads. Nucleic Acids Res. 2009;37(SUPPL. 2).

38. Li W, Godzik A. Cd-hit: A fast program for clustering and comparing large sets of protein or nucleotide sequences. Bioinformatics. 2006;22(13):1658–1659.

39. Fu L, Niu B, Zhu Z, Wu S, Li W. CD-HIT: Accelerated for clustering the next-generation sequencing data. Bioinformatics. 2012;28(23):3150–3152.

40. Radivojac P, Clark WT, Oron TR, Schnoes AM, Wittkop T, Sokolov A, et al. A large-scale evaluation of computational protein function prediction. Nature methods. 2013;10(3):221.

41. Buchfink B, Xie C, Huson DH. Fast and sensitive protein alignment using DIAMOND. Nat Methods. 2014 nov;12(1):59–60. Available from: http://www.nature.com/doifinder/10.1038/nmeth.3176.

42. Chollet F. Keras: Deep Learning library for Theano and TensorFlow; 2015.

43. Shannon P, Markiel A, Ozier O, Baliga NS, Wang JT, Ramage D, et al. Cytoscape: A software Environment for integrated models of biomolecular interaction networks. Genome Res. 2003;13(11):2498–2504.

44. Huang W, Li L, Myers JR, Marth GT. ART: A next-generation sequencing read simulator. Bioinformatics. 2012;28(4):593–594.

45. Leinonen R, Akhtar R, Birney E, Bower L, Cerdeno-T??rraga A, Cheng Y, et al. The European nucleotide archive. Nucleic Acids Res. 2011;39(SUPPL. 1).

46. Zerbino DR, Birney E. Velvet: Algorithms for de novo short read assembly using de Bruijn graphs. Genome Res. 2008;18(5):821–829.

47. Benson DA, Clark K, Karsch-Mizrachi I, Lipman DJ, Ostell J, Sayers EW. GenBank. Nucleic Acids Res. 2015;.

48. Kanehisa M, Goto S. KEGG: Kyoto Encyclopedia of Genes and Genomes. Nucleic Acids Research. 2000 Jan;28(1):27–30. Available from: https://academic.oup.com/nar/article/28/1/27/2384332.

List of Figures

1 A flowcart of the POEM pipeline. A: assembly; B: Gene calling; C: similarity clustering; D: identify intra-operonic genes; E: identify core operons; F: graph-based visualization 20

2 **A.** k-mer-position matrix construction, shown with a 2-mer example (POEM uses 3-mer). Each row is a k-mer and the column number stands for a position in the sequence. If a specific k-mer appears in the sequence, the corresponding cell of the KPM is set to 1, otherwise, 0; **B.** training and building an CNN based classification model from intergenic of operonic and non-operonic adjacency. 21

3 **Identifying Core Operons**. (1) find orthologous COG-annotated proximal gene pairs and concatenate them. (2) The resulting graph shows the *core operon* (3) Find the most similar operon in the dataset of gold standards and its corresponding GO annotations. In this example, there are 3 true positives (in the box), 1 false positive (COG2160), and 2 false negatives (COG2814 & COG3254). Precision is 0.75 and Recall is 0.6 22

4 Definition of precision and recall for a predicted operon. 23

5 **Mapping core functions to predicted operons**. **A**: predicted core function from SRR2155174 data set; **B-E**: predicted operons in different species. the arrows stand for the strands of genes, the color of box stands for the functional classification used in the COG database and the string above the box stands for the gene names. 24

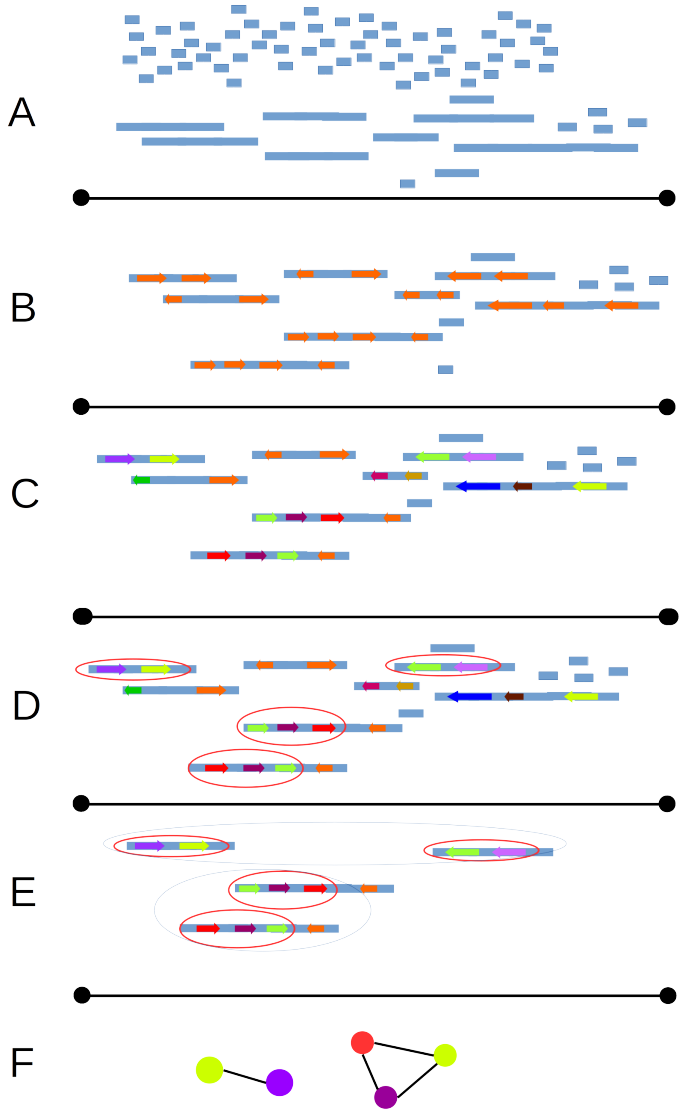


Figure 1. A flowcart of the POEM pipeline. A: assembly; B: Gene calling; C: similarity clustering; D: identify intra-operonic genes; E: identify core operons; F: graph-based visualization

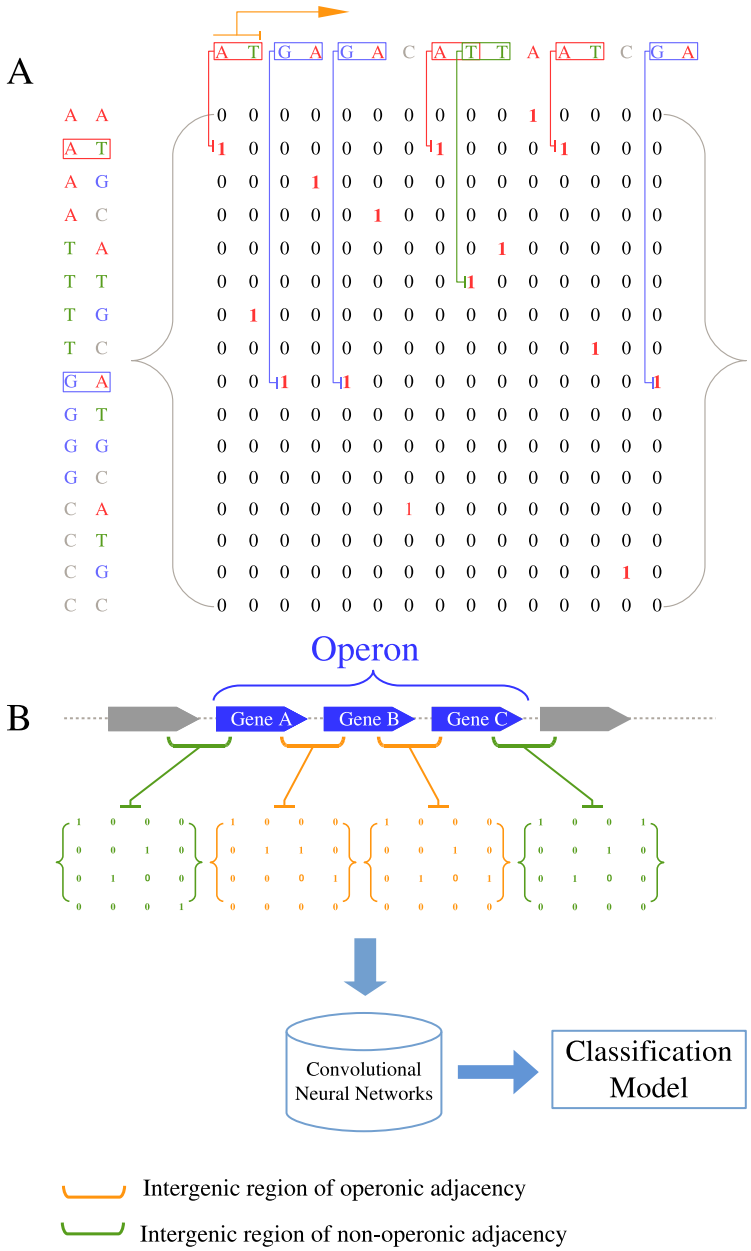


Figure 2. **A.** k-mer-position matrix construction, shown with a 2-mer example (POEM uses 3-mer). Each row is a k-mer and the column number stands for a position in the sequence. If a specific k-mer appears in the sequence, the corresponding cell of the KPM is set to 1, otherwise, 0; **B.** training and building an CNN based classification model from intergenic of operonic and non-operonic adjacency.

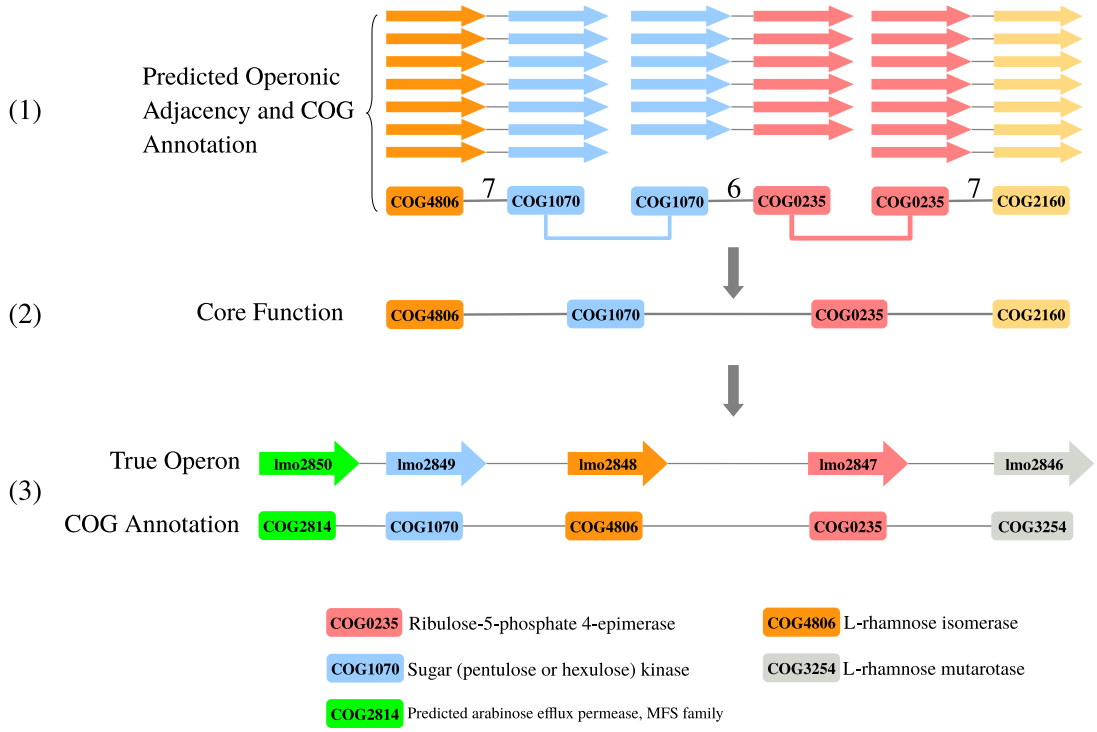
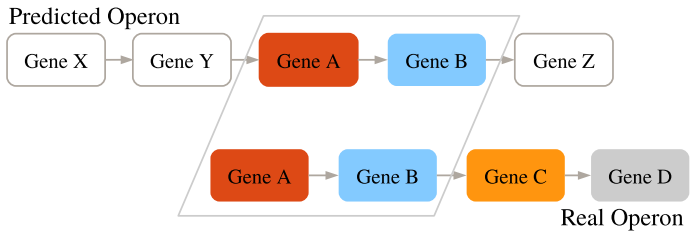


Figure 3. Identifying Core Operons. (1) find orthologous COG-annotated proximal gene pairs and concatenate them. (2) The resulting graph shows the *core operon* (3) Find the most similar operon in the dataset of gold standards and its corresponding GO annotations. In this example, there are 3 true positives (in the box), 1 false positive (COG2160), and 2 false negatives (COG2814 & COG3254). Precision is 0.75 and Recall is 0.6



$$Precision = \frac{TP}{TP+FP} = \frac{2}{5} = 0.4 \qquad Recall = \frac{TP}{TP+FN} = \frac{2}{4} = 0.5$$

Figure 4. Definition of precision and recall for a predicted operon.

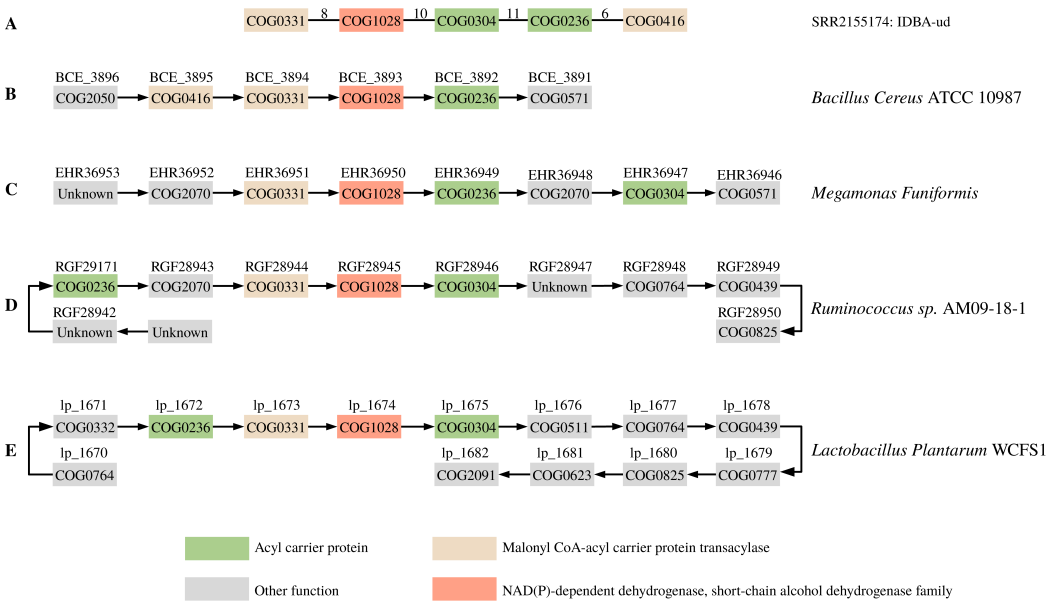


Figure 5. Mapping core functions to predicted operons. **A**: predicted core function from SRR2155174 data set; **B-E**: predicted operons in different species. the arrows stand for the strands of genes, the color of box stands for the functional classification used in the COG database and the string above the box stands for the gene names.