International Journal of Computational Intelligence Systems (2023) 16:175
https://doi.org/10.1007/s44196-023-00348-w

RESEARCH ARTICLE

Spark-Based Label Diffusion and Label Selection Community Detection Algorithm for Metagenome Sequence Clustering

Zhengjiang Wu[1] · Xuyang Wu[2] · Junwei Luo[1]

Received: 9 April 2023 / Accepted: 12 October 2023

Abstract
It is a challenge to assemble an enormous amount of metagenome data in metagenomics. Usually, metagenome cluster sequence before assembly accelerates the whole process. In SpaRC, sequences are defined as nodes and clustered by a parallel label propagation algorithm (LPA). To address the randomness of label selection from the parallel LPA during clustering and improve the completeness of metagenome sequence clustering, Spark-based parallel label diffusion and label selection community detection algorithm is proposed in the paper to obtain more accurate clustering results. In this paper, the importance of sequence is defined based on the Jaccard similarity coefficient and its degree. The core sequence is defined as the one with the largest importance in its located community. Three strategies are formulated to reduce the randomness of label selection. Firstly, the core sequence label diffuses over its located cluster and becomes the initial label of other sequences. Those sequences that do not receive an initial label will select the sequence label with the highest importance in the neighbor sequences. Secondly, we perform improved label propagation in order of label frequency and sequence importance to reduce the randomness of label selection. Finally, a merge small communities step is added to increase the completeness of clustered clusters. The experimental results show that our proposed algorithm can effectively reduce the randomness of label selection, improve the purity, completeness, and F-Measure and reduce the runtime of metagenome sequence clustering.

Keywords Metagenome · Sequence clustering · Label diffusion · Spark graphx · Community detection

1 Introduction

In metagenomics, assembling large amounts of sequence data obtained by sequencing is a meaningful way to analyze microbial communities [1]. In the face of new viruses, metagenomics can greatly help uncover the properties of new viruses and early therapeutic warning [2]. However, an efficient and scalable method is needed to assemble the sequences due to a large amount of metagenome sequence data and complex assembly algorithms [3]. In order to achieve this goal, the idea of clustering sequence before assembly has been proposed in recent years, as shown in Fig. 1. So clustering for sequence becomes the focus of research before assembly [4].

Regarding sequence clustering for machine learning-based. Kévin et al. [5] proposed a rank-flexible machine learning-based approach. The method labels sampled fragments according to a given classification level and learns sequence classification model tailored to this resolution level. Liang et al. [6] reported DeepMicrobes, an algorithm for short-read metagenome sequences classification. Regarding deep learning, an NMF-based generalized deep learning multi-view clustering (GDLMC) algorithm was proposed by Wang et al. In GDLMC, the implemented sequential updates the elements in the matrix guided by the learning rate [7]. Furthermore, in GDLC, the gradient values corresponding to the element updates were transformed into generalized

Zhengjiang Wu, Xuyang Wu and Junwei Luo have different contributions for this article.

✉ Junwei Luo
luojunwei@hpu.edu.cn

Zhengjiang Wu
wuzhengjiang@hpu.edu.cn

Xuyang Wu
212109020073@home.hpu.edu.cn

[1] College of Software, Henan Polytechnic University, No. 2001 Century Avenue, Jiaozuo 454000, Henan, China

[2] College of Computer Science and Technology, Henan Polytechnic University, No. 2001 Century Avenue, Jiaozuo 454000, Henan, China

weights and generalized deviations [8]. Regarding sequence clustering for community detection, SpaRC [9] is proposed based on a parallel LPA [10] to cluster metagenome sequence. Lu et al. [11] used the Louvain algorithm [12] instead of LPA in SpaRC. The Louvain algorithm improved the completeness of clustering to a small extent. Regarding other clustering schemes, a pre-assembled binning method based on sparse dictionary learning and elastic network regularization was proposed by Kyrgyzov et al. [13]. This method exploits the sparsity and nonnegativity constraints inherent to k-mer count data and eliminates the interpretability problems associated with SVD.

SpaRC can cluster metagenome sequences and help alleviate assembly dilemmas [9]. It has shown high efficiency, high scalability, and flexible deployment in the face of massive data. SpaRC uses the parallel LPA to cluster and divide different sequences, while LPA is a community detection algorithm based on label propagation [10]. In the initial stage of LPA, a unique label is tagged to each node, and in each subsequent iteration, each node selects the one with the most labels among its neighbor nodes as its community label. One is randomly selected if multiple neighbors are tied for the first number of community labels. Therefore, LPA suffers from a random selection of labels. With the continuous iteration of the algorithm, a tiny error caused by this randomness will be cumulated throughout the operation and cannot be corrected in the whole process.

To address the randomness of label selection during clustering and improve the completeness of metagenome sequence clustering. This paper proposes a parallel label diffusion and label selection community detection algorithm (PLDLS) based on SpaRC [14]. The algorithm uses multiple metrics to calculate the importance of node to select core node, and core node diffusion and fast label selection are performed [15]. Then random nature of label propagation is reduced using label frequency combined with node importance. Last, the small communities resulting from clustering errors are merged. To solve the problem of low computational efficiency due to single-computer processing [16]. The algorithm was deployed on Spark [17], whose fast in-memory computation improves the computational efficiency of sequence clustering.

This paper is organized as follows. Section 2 describes the steps and characteristics of PLDLS. Section 3 describes the datasets, hardware and software environments used for the experiments and reports the experimental results and result analysis. Section 4 concludes the paper and sketches issues for future work.

2 The Spark-Based Parallel Label Diffusion and Label Selection Community Detection Algorithm (PLDLS)

PLDLS, a parallel community detection algorithm in the spark framework, is designed to address the randomness of label selection during clustering and improve the completeness of metagenome sequence clustering. The algorithm combines four steps of node scoring and core node

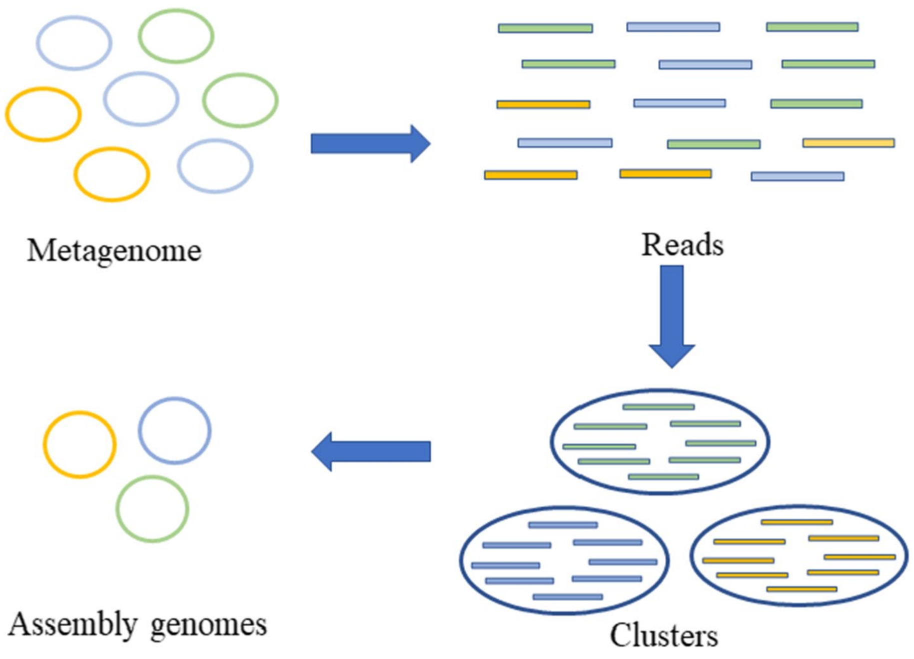

Fig. 1 Metagenome assembly process

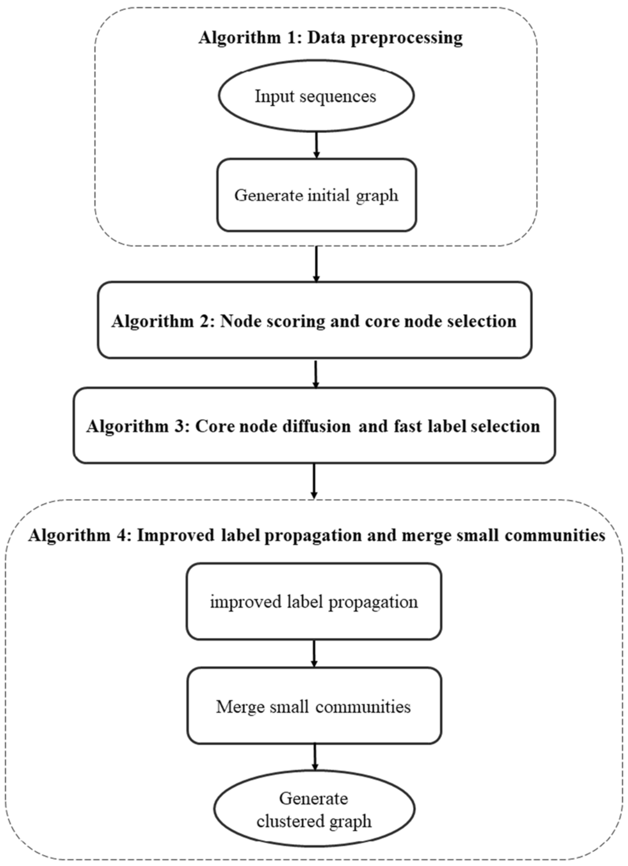
Algorithm 1: Data preprocessing
Input sequences
Generate initial graph
Algorithm 2: Node scoring and core node selection
Algorithm 3: Core node diffusion and fast label selection
Algorithm 4: Improved label propagation and merge small communities
improved label propagation
Merge small communities
Generate clustered graph

Fig. 2 The flowchart of PLDLS for metagenome sequence clustering

selection, core node diffusion and fast label selection, improved label propagation, and merge small communities to cluster the graph. The following is a detailed description of each step. Figure 2 illustrates the PLDLS flowchart for metagenomic sequence clustering.

2.1 Data Preprocessing

To enable the community detection algorithm to deal with metagenome dataset, SpaRC [9] proposed to evaluate whether an edge can be formed by calculating the number of shared k-mers between a pair of sequences, thus constituting a graph.

The sequence is split into k-mer of specified length and is filtered out the same k-mers. Two sequences form an edge if they satisfy the same threshold of the number of k-mers [18]. The composed undirected graph is used as the input data for the community detection algorithm. Each node in the graph represents a sequence.

2.2 Node Scoring and Core Node Selection

To avoid the randomness of label diffusion in the next stage [15]. The Jaccard similarity coefficient [19] and the degree of the nodes are used in the algorithm to define the importance of the nodes. The higher than average node importance is defined as the core node. Core nodes are strongly connected to other nodes in the same community.

The Jaccard similarity coefficient measures the degree of similarity between two sets. It reflects the complete similarity of elements between two nodes by calculating the ratio of the intersection of the neighbor nodes to their union. The similarity between nodes is calculated by (1).

$$\text{similarity}_{i,j}^{\text{Jaccard}} = \frac{|N_i \cap N_j|}{|N_i \cup N_j|} \tag{1}$$

where N_i and N_j denote the set of neighbor of nodes i and j.

The node scoring phase requires two values to calculate the node importance value.

The first value is the sum of the similarity of nodes. The strength of the connection between node i with neighbors is essential in determining whether node i is a core node in this community. Nodes with higher similarity have stronger ties with their neighbor nodes. The sum of similarity between node i and neighbor nodes is calculated by (2).

$$similarity_{sum(i)} = \sum_{j \in N_i} similiarity_{i,j}^{Jaccard} \tag{2}$$

where $similarity_{sum(i)}$ is the sum of similarity between node i and its neighbor nodes. N_i is the set of neighbor nodes of node i. j is a node in N_i.

The second value is the degree of node. The degree of node is defined as the number of neighbors of this node, and higher degree of node indicates that this node has more connections with neighbor nodes [20].

The importance value can be obtained by combining the strength of the node relations with its neighbors (sum of similarity) with the degree of the node. The importance of node i is calculated by (3).

$$NI(i) = similarity_{sum(i)} \times deg(i)^2 \tag{3}$$

Example 1: suppose there is now an initial community network, as shown in Fig. 3. The average node importance is 4.25, where nodes B, C, E, and H are the core nodes. Figure 3 is the initial graph where each node importance has been calculated. In node B, B denotes the node ID, and 6 denotes the node importance.

Algorithm 1 Data preprocessing

```
Input: DataSet of Sequences
Output: Initial Graph
1: for sequences ∈ Data Set do
      Create dictinct (k-mer, sequences)
   end for
2: Divide the sequences of the same k-mer into one (k-mer, sequences) tuple
3: for ((k-mer, sequences) tuple) ∈ set of (k-mer, sequences) tuple do
      if (min_ kmer_count ≤ count(kmer) ≤ max_kmer_count) then
          Generate edge using sequence as the node
      end if
   end for
4: Calculate the number of shared k-mers between two nodes
5: for ((node1,node2) tuple) ∈ set of edges do
      if (shared k-mers ∈ min_share_kmers) then
          Keep edge
      end if
   end for
6: Generate Initial Graph
```

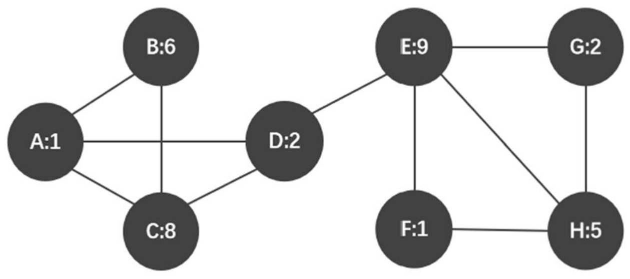

Fig. 3 Example diagram of node scoring and core node selection

Algorithm 2 Node scoring and core node selection

```
Input: Initial graph (the output of algorithm 1)
Output: Graph with core nodes
1: Calculate the importance of all nodes
2: for node ∈ Set of nodes do
      if (the importance of node > the average importance of all nodes) then
          Label as core nodes
      end if
   end for
3: Generate graph with core nodes
```

2.3 Core Node Diffusion and Fast Label Selection

In this section, based on the core node selection in the previous section, the non-core nodes will select the label of the core node that satisfies the conditions as their labels [14]. Since all the nodes have received the labels assigned by the core nodes after the diffusion step, the label selection will be more accurate in the improved label propagation phase.

There are two stages of core node diffusion. In the first-level diffusion stage, the importance of the neighbors of the core node is first calculated based on Eq. (3), and the node with the maximum importance is selected as the essential neighbor of the core node (first-level core node) [14]. The earliest selected node is selected if there is more than one maximum value. Then, the importance values of core node and first-level core node are calculated using Eq. (3). If the core node is less than the importance of the first level core node, the label of the core node is changed to the label of the first level core node. Last, the label of the node with the largest importance value between the two nodes is used as their common neighbors label.

In the second-level diffusion stage, in the beginning, the neighbor nodes of the first-level core nodes (first-level

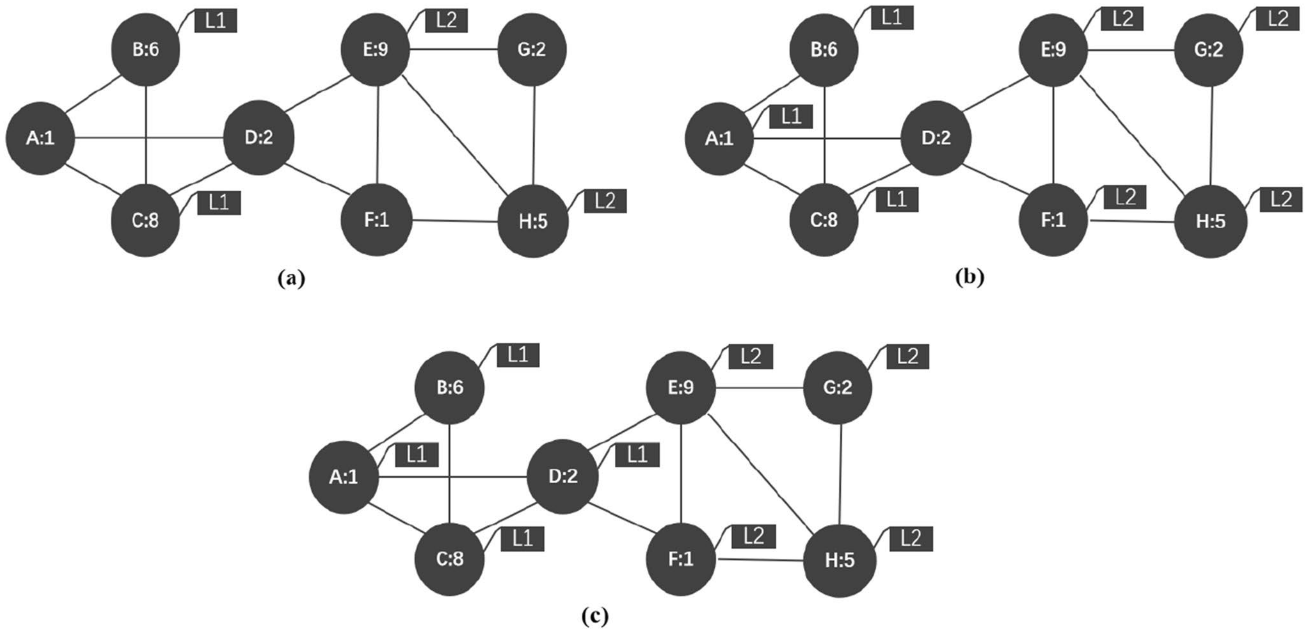

Fig. 4 **a**, **b** is an example diagram of the first-level diffusion stage; **c** is an example diagram of the second-level diffusion stage

neighbor nodes) are found. Secondly, the similarity between the first-level core node and the first-level neighbor node is calculated using Eq. (1). If the importance of the first-level core node is greater than the importance of the first-level neighbor node, and the similarity between them is greater than 0.5. In that case, the label of the first-level neighbor node is updated to the label of the first-level core node. Similarity greater than 0.5 indicates that more than half of the neighbors between the two nodes overlap [21], which suggests that the two nodes originally belonged to the same community.

After the above two diffusion processes, most of the nodes have been given attributed labels. The untagged nodes will choose the most important node among the neighbor nodes as the label.

Example 2: continuation of Example 1. The highest importance among the neighbor of the core nodes are selected as the first-level core nodes according to Fig. 3. In (a) of Fig. 4, The labels of nodes C and E are updated to labels L1 and L2, respectively. Since the importance of core nodes B and H is lower than that of the first-level core nodes C and E, nodes B and H labels are updated to L1 and L2, respectively.

In (b) of Fig. 4, the label of the node with the highest importance between the core node and the first level core node is selected as the labels of their common neighbors. Nodes C and E have the highest importance, so the common neighbor node A of nodes B and C change their labels to L1. The common neighbor nodes F and G of nodes E and H change their labels to L2.

In (c) of Fig. 4, the similarity of first-level neighbor nodes D and C is more significant than 0.5, and the importance of first-level core node C is greater than that of node D. So the label of node D is changed to label L1 of node C.

2.4 Improved Label Propagation

Spark provides the Pregel API [21], a large-scale graph computation framework, in the GraphX component. The algorithm uses the Pregel API to compute the graph during the improved label propagation phase.

After the above stage, all nodes have updated their initial labels. In this stage, the improved parallel label propagation process performs several iterations [19]. All nodes will calculate the label frequency of neighbor nodes synchronously, and then the label with the highest frequency is selected and updated as its label. If there is more than one label with the highest frequency, the node label with the highest importance is selected by calculating the node importance according to Eq. (3). The first label received is selected if the highest importance has more than one equal. Until the labels of all nodes no longer change.

Example 3: continuation of Example 2. Take Fig. 5 as an example. Node D receives two labels L1 and L2. Since the labels have the same frequency, then node D selects label L2 of node E with the highest importance as its label.

Algorithm 3 Core node diffusion and fast label selection

```
Input: Graph with core nodes (the output of algorithm 2)
Output: Graph after Two Propagation
1: for node ∈ set of core nodes do
       if ( Importance (neighbor of core node) > Importance (core node))
           Label neighbor of core node as first level core node (FLCN)
           Send label of FLCN to core node
       Common_nodes (FLCN join core node)
       Send label of Max_importance (FLCN and core node) to common_nodes
       end if
   end for
2: for node ∈ Set of FLCN do
       if (Similarity (FLCN,neighbors)>0.5&&Importance (neighbor of FLCN<FLCN))
           Send label of FLCN to neighbor of FLCN
       end if
   end for
3: for node ∈ unmarked nodes do
       Send label of Max_importance (neighbors) to unmarked node
   end for
4: Generate graph after two propagation
```

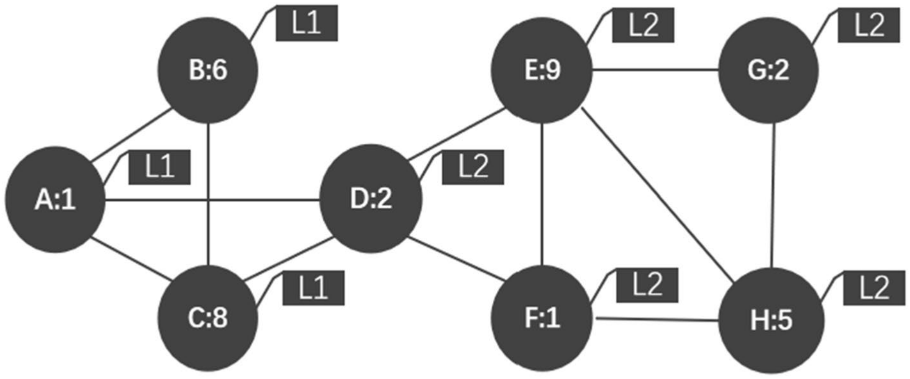

Fig. 5 Example diagram of label propagation step

2.5 Merge Small Communities

After the improved label propagation, most of the nodes received the final label. However, there are still some nodes that belong to the same community that are divided into several small communities. These small communities cause a decrease in completeness, so merge small communities is needed.

In this section, communities with smaller than average community values are categorized as small communities. The node with the highest degree in each small communities is selected as the backup node. If the neighbor node with the highest degree has a different label than the backup node, then label of all nodes in the backup node community are updated to the label of neighbor node with the highest degree. If they are the same, no update is required. This completes the merging of communities and forms the final community.

Example 4: continuation of Example 3. Taking Fig. 6 as an example, all nodes in Fig. 5 in the previous section are divided into two communities. The average community size is 4. The label L1 community is defined as a small

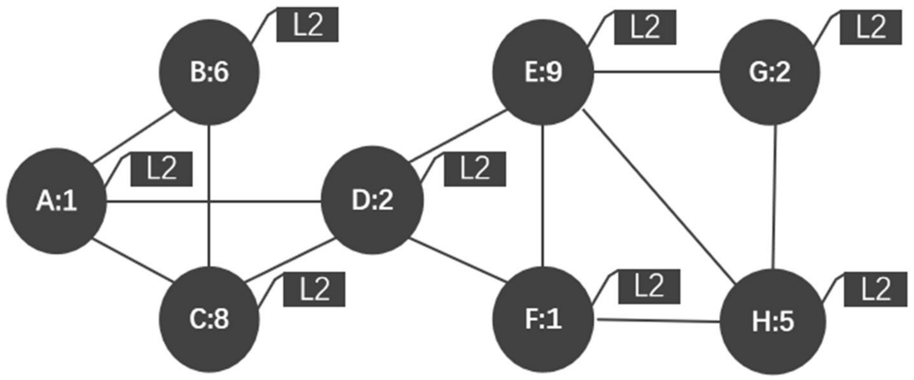

Fig. 6 Example diagram of merge small communities step

Table 1 Dataset

Dataset	Read length(bp)	Read number	Size(GB)
Human Sample_15	150×2	250,000	0.8 G
MouseGut Sample_16	150×2	400,000	1.1 G
MouseGut Sample_18	150×2	400,000	1.2 G

community. Node C is selected as a backup node for the small community. Node D is a neighbor node whose degree is more significant than node C, and both are labeled differently. So the community label belonging to backup node C is updated to label L2. The final result is shown in Fig. 6.

```
Input: Graph after two propagation (the output of algorithm 3)
Output: Clustered graph
1: Perform improved label propagation steps
2: Set of small communities are the communities which are smaller than the average community size
3: for small communities ∈ set of small communities do
       Backup_node ∈ node with the highest degree
         if (Label (backup_node)≠Label (Max_degree(neighbors of backup_node))
             Send label of Max_degree (neighbor of backup_node) to label of backup_node
         end if
   end for
4: Generate clustered graph
```

3 Experiments

3.1 Dataset and Experimental Environment

Table 1 describes the three datasets from CAMI2 [22]. CAMI2 is a simulation dataset from the second CAMI Challenge (https://data.cami-challenge.org/participate). This dataset contains 64 samples with different genome coverage. The first dataset is from the human microbiome. The second dataset uses mouse gut sample16, and the third uses mouse gut sample18.

The algorithms are programmed in Scala programming language and publicly available to download from https://github.com/xuyangwu2023/MetagenomeClustering. The detailed equipment and configuration of each computational nodes are Memory: 16 GB; Disks: 64 GB; Java version: JDK1.8; Scala version: 2.12; Hadoop(HDFS) version: 3.1.1; Spark version: 3.2.0.

3.2 Evaluation Indicators

This paper evaluates the results using four metric measures: purity, completeness, F-Measure, and run-time.

Purity indicates the degree to which a cluster contains the same genome after clustering, and the purity metric for the

Table 2 Purity comparison

Dataset	Algorithm	Mean_Purity (%)	Median_ Purity (%)
Human Sample15	PLDLS	**87.12**	96.30
	LPA	86.94	**96.53**
	COPRA	77.99	87.50
	Louvain	81.11	89.30
MouseGut Sample16	PLDLS	**96.29**	**98.34**
	LPA	92.98	97.76
	COPRA	86.11	95.04
	Louvain	76.44	94.44
MouseGut Sample18	PLDLS	**96.58**	98.07
	LPA	94.76	**99.21**
	COPRA	90.97	95.87
	Louvain	74.56	75.00

Bold indicates the best record of four algorithms for each dataset

results is divided into median purity and average purity. The purity is calculated by (4).

$$P_j = \frac{maxa_{i1}}{\sum_{i=1}^{n} a_{i1}}, 1 \le j \le m \tag{4}$$

Completeness indicates the extent to which sequences of the same genome are clustered in a single cluster. As with purity, the completeness measure for the results is divided into median and average completeness. The completeness is calculated by (5).

$$C_i = \frac{maxa_{1j}}{\sum_{j=1}^{m} 1j}, 1 \le i \le n \tag{5}$$

The F-Measure is derived from the combination of purity and integrity and is calculated by (6).

$$F_\beta = \frac{(\beta^2 + 1)P.R}{\beta^2.P + R} \tag{6}$$

where P is purity, R is completeness, and β is the parameter that adjusts the weights of both equal to 1.

3.3 Comparative Experimental

To validate the effectiveness of PLDLS in metagenome sequence clustering, this paper validates the performance of four algorithms, PLDLS, LPA [9], Louvain [12], and COPRA [23], on three different datasets. COPRA is based on improving the LPA to discover overlapping communities.

Since the actual community segmentation results of this test dataset are known, this paper evaluates the performance of cluster results in four aspects: purity, completeness, F-Measure, and time.

3.3.1 The Purity of Clustering

Regarding purity, the average purity of PLDLS is significantly higher than the other algorithms on the three datasets. However, the median purity of PLDLS on the Human Sample 15 and MouseGut Sample 18 datasets is slightly lower than that of LPA. This occurs mainly because LPA clusters too many low-purity clusters, resulting in a lower average purity than PLDLS. As shown in Table 2.

Figure 7 shows the purity distribution maps derived from the MouseGut Sample16 dataset on the four algorithms. Where (a) is the clustering result derived by PLDLS. Most clusters have their purity concentrated between 90 and 100%, and only a few are below 80%. (b) is the clustering result from the LPA, most of the clusters are concentrated between 80 and 100% purity, and a considerable number of clusters are below 80% region. (c) and (d) is the purity distribution obtained by COPRA and Louvain; nearly half of the clusters are below 80% purity.

We counted the purity data of the ten most significant clusters in the PLDLS and LPA for the MouseGut Sample16 dataset. The data show that although three of the ten most significant clusters in the LPA have higher purity than the PLDLS, the overall purity of the clusters in the PLDLS is higher. The specific results are shown in Fig. 8.

3.3.2 The Completeness of Clustering

Table 3 shows the completeness of the four algorithms on the three datasets. The results show that the median and average completeness metrics of PLDLS on the three datasets are significantly higher than the other algorithms by about 25–60%.

Figure 9 shows the completeness distribution of the MouseGut Sample16 dataset derived from the four algorithms. The clustering completeness of the PLDLS is primarily distributed above 40% and more concentrated in the 80% region. The clustering completeness of the LPA and COPRA is primarily distributed below 40%. The completeness of the Louvain is primarily distributed below 20%.

Figure 10 shows the completeness statistics of the top ten most significant clusters in the clustering results of the LPA and PLDLS on the MouseGut Sample16 dataset. From the data presented by the results in the figure, it can be seen that the completeness of the top ten most significant clusters found by the PLDLS is significantly higher than that of the top ten clusters found by the LPA.

3.3.3 The F-Measure of Clustering

We compared the F-Measure of the PLDLS and the other three algorithms, as Table 4 shows. The F-Measure [24]

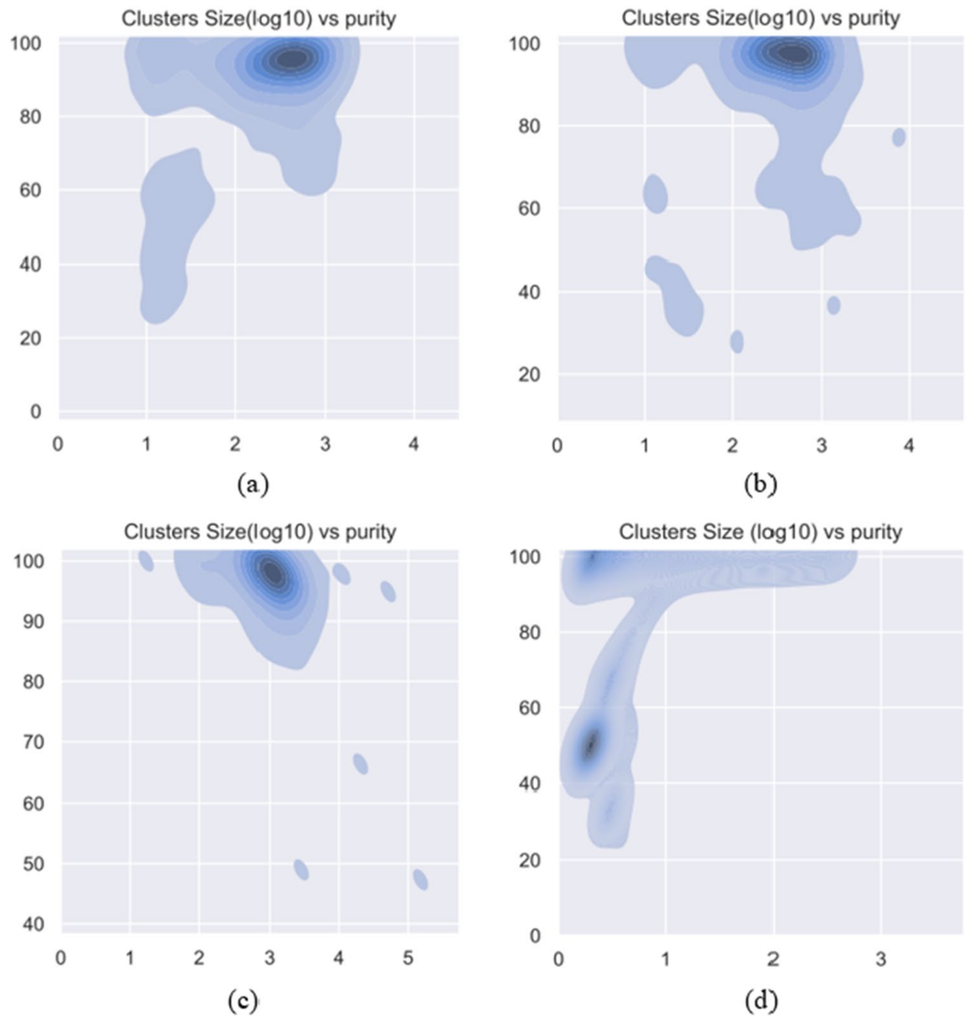

Fig. 7 Purity distribution plots for the MouseGutSample16 dataset. **a** is the purity distribution plot from the PLDLS, **b** is the purity distribution plot from the LPA, **c** is the purity distribution plot from the COPRA, and **d** is the purity distribution plot from the Louvain

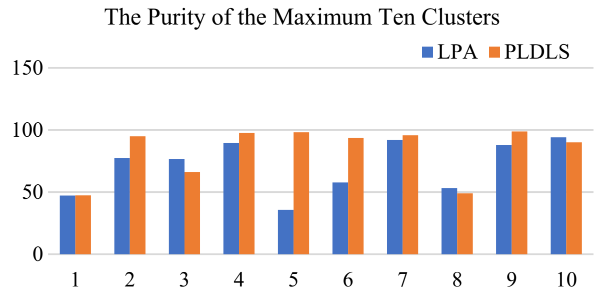

Fig. 8 LPA and PLDLS to find the ten clusters with the greatest purity

Table 3 Completeness comparison

Dataset	Algorithm	Mean_completeness (%)	Median_completeness (%)
Human Sample15	PLDLS	**62.26**	**61.06**
	LPA	28.69	23.79
	COPRA	5.28	4.33
	Louvain	8.56	9.43
MouseGut Sample16	PLDLS	**76.41**	**81.65**
	LPA	19.14	19.07
	COPRA	8.59	7.95
	Louvain	10.81	10.34
MouseGut Sample18	PLDLS	**87.71**	**89.80**
	LPA	60.54	61.68
	COPRA	19.65	21.27
	Louvain	12.39	11.93

of PLDLS is significantly higher than the other algorithms in the three datasets. The clusters formed by PLDLS after clustering are closer to the number of clusters of the actual data, and the closer the clusters formed after clustering is to the number of real clusters, the better the clustering effect is.

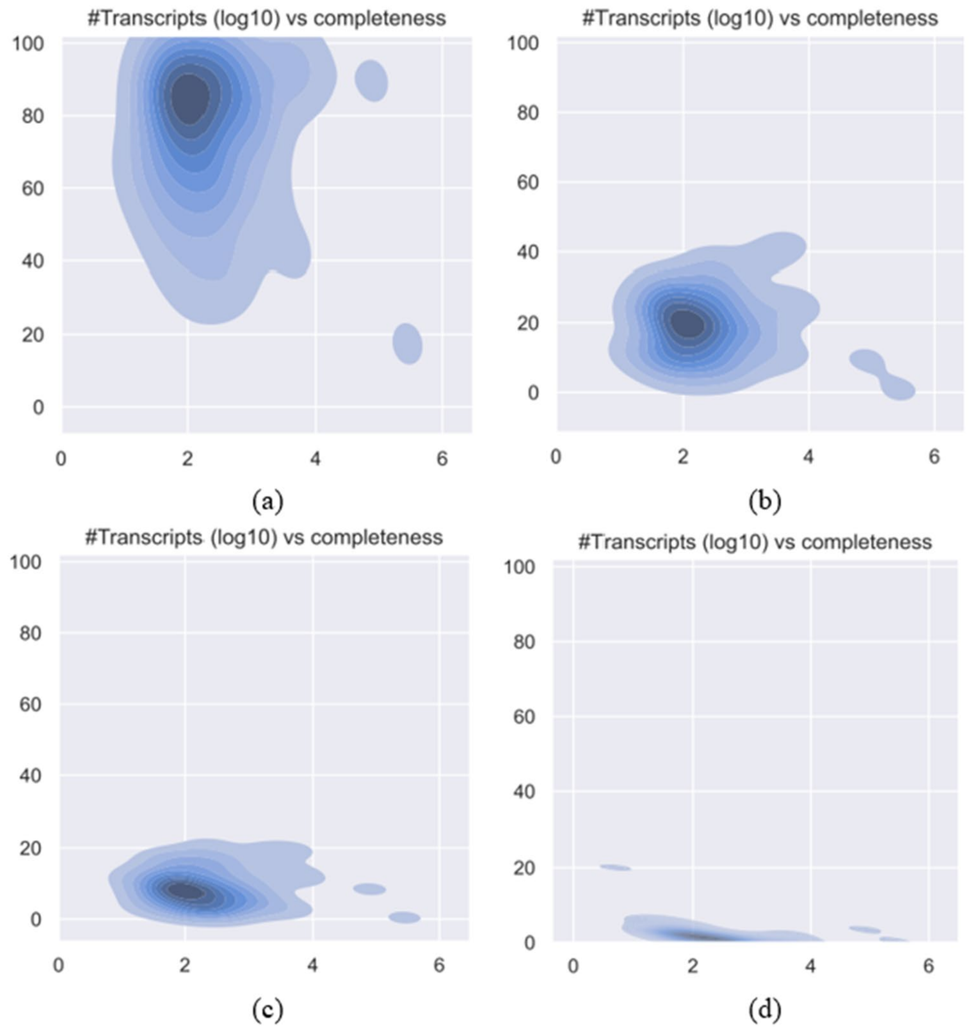

Fig. 9 Completeness distribution of the MouseGut Sample16 dataset. **a** is the completeness distribution from PLDLS, **b** is the completeness distribution from LPA, **c** is the completeness distribution from COPRA, and **d** is the completeness distribution from Louvain

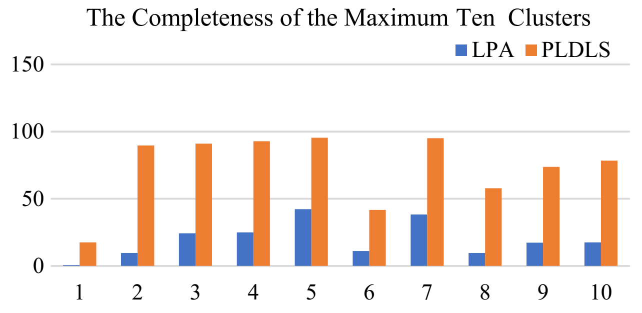

Fig. 10 LPA and PLDLS to find the ten clusters with the greatest completeness

Table 4 F-Measure comparison of four algorithms

Dataset	Algorithm	F-Measure	Clusters
Human Sample15	PLDLS	**0.72**	**343**
	LPA	0.42	4122
	COPRA	0.10	7604
	Louvain	0.15	52,434
MouseGut Sample16	PLDLS	**0.85**	**143**
	LPA	0.31	872
	COPRA	0.15	1390
	Louvain	0.18	19,292
MouseGut Sample18	PLDLS	**0.91**	**152**
	LPA	0.73	2028
	COPRA	0.31	3622
	Louvain	0.21	35,758

3.3.4 The Runtime and Parallelism of Clustering

Spark uses Resilient Distributed Dataset (RDD) as the primary data structure. The data are partitioned and deployed on different nodes, where parallelism is the number of data partitions. According to research, the running efficiency of Spark-based programs is affected by parallelism, so we set

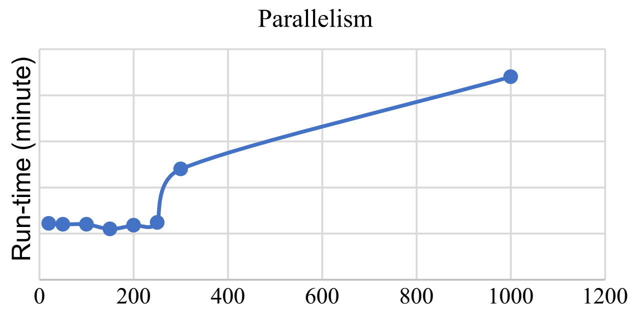

Fig. 11 Impact of parallelism on algorithm execution time

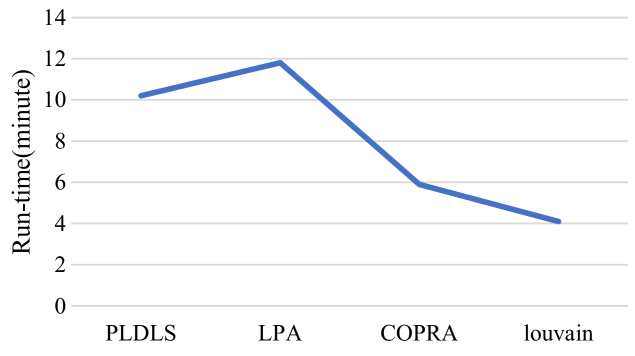

Fig. 12 Time comparison of four algorithms

different parallelism degrees to test the effect on the running time efficiency of the PLDLS.

We test the parallelism of the PLDLS on MouseGut Sample18, as shown in Fig. 11. The experimental results show that when the value of spark.default.parallelism is taken to be no more than six times of Executor-Cores, the program runs with an error of no more than 1 min. If it exceeds six times, the program runtime increases significantly. If the parallelism is set too high, many small tasks will be generated, and these small tasks will take up more resource overhead. Too few partitions will cause the cluster resources not to be fully utilized, making the algorithm performance suffer.

Figure 12 records the clustering runtimes of the four algorithms on the MouseGut18 dataset.

In general, the serial experiments show that PLDLS runs slower than Louvain and COPRA. However, the completeness, purity, and F-Measure metrics of clustering are better than Louvain, COPRA, and LPA. If a trade-off is made between time and clustering effectiveness, the PLDLS is more suitable for sequences clustering.

4 Conclusion

In this paper, spark-based label diffusion and label selection community detection algorithm for metagenome sequence clustering is developed. To reduce the randomness in the sequence clustering process, firstly, the algorithm selects the core sequence based on the importance of the sequence, then selects core sequence and spreads the labels of core sequence to other sequences. Secondly, the algorithm performs improved label propagation based on label frequency and sequence importance. Finally, the small communities are merged.

Experimental evaluation of purity and completeness proves that although the proposed method has a slower running time than COPRA, Louvain, our algorithm improves the purity of the clustering results and significantly improves the completeness of the clusters without degrading the purity. In future work, we want to combine parameter optimization models for metagenome sequence clustering with existing tools.

Acknowledgements All authors acknowledge the financial support received from the National Natural Science Foundation of China No. 61972134.

Author Contributions All authors contributed to the study conception and design. The study was mainly conceived and designed by ZW. The experiments were performed by XW and JL. The first draft of the manuscript was written by ZW and all authors commented on previous versions of the manuscript. ZW and JL edited the manuscript. All authors read and approved the final manuscript.

Funding This work has been supported in part by the National Natural Science Foundation of China under Grant No. 62372156 and 61972134, Innovative and Scientific Research Team of Henan Polytechnic University under No. T2021-3.

Data Availability The datasets used during the study are available at CAMI (cami-challenge.org).

Declarations

Conflict of Interest The authors declare that there is no conflict of interest.

Ethical Approval Not applicable.

Consent to Participate Not applicable.

Consent for Publication Not applicable.

References

1. Yunyan, Z., Min, L., Jiawen, Y.: Recovering metagenome-assembled genomes from shotgun metagenomic sequencing data: methods, applications, challenges, and opportunities. Microbiol. Res. **260**, 127 (2022)
2. Wentao, Z., Fuhan, Y., Shiyu, M., Ruiliang, W., Haotian, C., Yuefei, R., Shenghua, L., Pengfei, W., Yang, Y., Wei, L., Junfeng, Z., Xudong, Y.: Bladder cancer-associated microbiota: recent advances and future perspectives. Heliyon **9**(1), e13012 (2023)
3. Fadiji, A.E., Babalola, O.O.: Metagenomics methods for the study of plant-associated microbial communities: a review. J. Microbiol. Methods **170**(2), 105 (2020)
4. Wang, F.Y., Qin, R., Wang, X., Hu, B.: Metasocieties in metaverse: metaeconomics and metamanagement for metaenterprises and metacities. IEEE Trans. Comput. Soc. Syst. **9**(1), 2–7 (2022)
5. Kévin, V., Pierre, M., Maud, T., Jean-Baptiste, V., Jean-Philippe, V.: Large-scale machine learning for metagenomics sequence classification. Bioinformatics (Oxford, England) (2016). https://doi.org/10.1093/bioinformatics/btv683
6. Qiaoxing, L., Paul, W.P., Yu, L., Bin, Z., Lai, W.: Deepmicrobes: taxonomic classification for metagenomics with deep learning. NAR Genom. Bioinform. (2020). https://doi.org/10.1093/nargab/lqaa009
7. Wang, D., Li, T., Deng, P., Liu, J., Huang, W., Zhang, F.: A generalized deep learning algorithm based on nmf for multi-view clustering. IEEE Trans. Big Data **9**(1), 328–340 (2023)
8. Wang, D., Li, T., Deng, P., Zhang, F., Huang, W., Zhang, P., Liu, J.: A generalized deep learning clustering algorithm based on non-negative matrix factorization. ACM Trans. Knowl. Discov. Data **17**, 1–20 (2023)
9. Lizhen, S., Xiandong, M., Elizabeth, T., Michael, M., Zhong, W.: Sparc: scalable sequence clustering using Apache spark. Bioinformatics (Oxford, England) **35**(5), 760 (2019)
10. Nandini, R.U., Reka, A., Soundar, K.: Near linear time algorithm to detect community structures in large-scale networks. Phys. Rev. E (2007). https://doi.org/10.1103/PhysRevE.76.036106
11. Lu, Y., Deng, L., Wang, L., Li, K., Wu, J.: Improving metagenome Sequence Clustering Application Performance Using Louvain Algorithm, pp. 386–400. Springer, Singapore (2020)
12. Blondel, V.D., Guillaume, J.-L., Lambiotte, R., Lefebvre, E.: Fast unfolding of communities in large networks. J. Stat. Mech. **2008**(10), 100 (2008)
13. Olexiy, K., Vincent, P., Stéphane, G., Bruno, F., Thomas, B.: Binning unassembled short reads based on k-mer abundance covariance using sparse coding. GigaScience (2020). https://doi.org/10.1093/gigascience/giaa028
14. Weitong, Z., Ronghua, S., Licheng, J.: Large-scale community detection based on core node and layer-by-layer label propagation. Inform. Sci. **632**, 1–18 (2023)
15. Hamid, R., Asgarali, B., Esmaeil, N.: PLDLS: a novel parallel label diffusion and label selection-based community detection algorithm based on spark in social networks. Expert Syst. Appl. **183**, 115 (2021)
16. Ketu, S., Mishra, P.K., Agarwal, S.: Performance analysis of distributed computing frameworks for big data analytics: hadoop vs spark. Comput. Sist. (2020). https://doi.org/10.13053/cys-24-2-3401
17. Tang, Z., Zeng, A., Zhang, X., Yang, L., Li, K.: Dynamic memory-aware scheduling in spark computing environment. J. Parallel Distrib. Comput. **141**, 10 (2020)
18. Zhixia, T., Linyue, S., Haihao, Y., Chengyan, W., Zhen, T.: Measuring functional similarity of lncrnas based on variable k-mer profiles of nucleotide sequences. Methods (San Diego, Calif) **212**, 21 (2023)
19. Shital, K., Sudhir, D.: Cross domain-based ontology construction via Jaccard semantic similarity with hybrid optimization model. Expert Syst. Appl. **178**, 115046 (2021)
20. Jai, M., Aditya, S., Amitabha, T.: Exact and approximate results on the least size of a graph with a given degree set. Discrete Appl. Math. **333**, 32 (2023)
21. Lu, H.-C., Hwang, F.J., Huang, Y.-H.: Parallel and distributed architecture of genetic algorithm on Apache Hadoop and Spark. Appl. Soft Comput. J. **95**, 106497 (2020)
22. Fernando, M., Adrian, F., ZhiLuo, D., David, K., et al.: Critical assessment of metagenome interpretation: the second round of challenges. Nat. Methods **19**(4), 429–440 (2022)
23. Gregory, S.: Finding overlapping communities in networks by label propagation. New J. Phys. **12**(10), 103018 (2010)
24. Soleymani, R., Granger, E., Fumera, G.: F-measure curves: a tool to visualize classifier performance under imbalance. Pattern Recognit. **100**, 107146 (2020)