An Orthologous Group Clustering Technique based on the Grid Computing

J.S. Oh[1], T.K. Kim[2], S.S. Kim[3], H. R. Kwon[4], Y.C. Kim[4], J.S. Yoo[5], W.S. Cho[1]

[1] *Department of Management Information System, Chungbuk National University, CheongJu, 361-763, Korea*

[2] *Department of Information Industrial Engineering, Chungbuk National University, CheongJu, 361-763, Korea*

[3] *Department of Computer Science, Chungbuk National University, CheongJu, 361-763, Korea*

[4] *School of Life Science, Chungbuk National University, CheongJu, 361-763, Korea*

[5] *Department of Information Communication Engineering, Chungbuk National University, 361-763, Korea*

Email : {ofang, tkkim, sskim04, mrjesus, youngkim, yjs, wscho}@chungbuk.ac.kr

ABSTRACT: Orthologs are genes having the same function across different species that specialize from a single gene in the last common ancestor of these species. Orthologous groups are useful in the genome annotation, studies on gene evolution, and comparative genomics. However, the construction of an orthologous group is difficult to automate and it takes so much time. It is also hard to guarantee the accuracy of the constructed orthologous groups. We propose a system to construct orthologous groups on many genomes automatically and rapidly. We utilize the grid computing to reduce the sequence alignment time, and we use clustering algorithm in the application of database to automate whole processes. We have generated orthologous groups for 20 complete prokaryotes genomes just in a day because of the grid computing. Furthermore, new genomes can be accommodated easily by the clustering algorithm and grid computing. We compared the generated orthologous groups with COGs (Clusters of Orthologous Group of Proteins) and KO (KEGG Ortholog). The comparison shows about 85 percent similarity compared with previous well-known orthologous databases.

1 INTRODUCION

Recently, complete genomes of various species are sequenced rapidly by using automated tools. One of the most important issues for the sequence data is to predict the functions of the genes in the segment. Orthologous groups are the most useful concept to estimate the function of the genes.

An orthologous group is a collection of the genes in two or more species that have evolved from a common ancestor. Ancestral genes have been distributed into the genome of the various organisms by the speciation during the biological evolution processes. Therefore, the distributed genes of each organism have the biologically same function and sequence pattern. Sequence alignment is needed to identify the orthologous relationship among the species. In contrast, paralogs are genes in different species that are originated by the duplication of a gene in a common ancient genome. Paralogs evolve new functions whereas orthologs retain the same function in the course of the evolution, even if these are related to the original one.

Identification of orthologs is critical for reliable prediction of gene function in newly sequenced genomes. However, construction of the orthologous groups are difficult task to automate and it takes a lot of sequence alignment time as the number of complete genomes increases. In addition, it is hard to guarantee the accuracy of the constructed orthologous groups.

To solve these problems, we develop a system to cluster orthologous groups automatically and rapidly. We first utilize grid computing to speed-up huge number of pair-wise sequence comparisons for large amount of sequences. We then devise a new clustering algorithm to cluster orthologous groups automatically. In the sequence comparisons between two genomes, we exploit the cut-off value in order to raise the functional accuracy of orthologous groups.

We clustered orthologous groups for 20 complete prokaryote genomes just in a day by applying grid computing; note that it takes more than two weeks if we use a single server for the same task. To verify the accuracy of the generated orthologous groups, we compared generated orthologous groups with other well-known orthologous databases such as COGs (Clusters of Orthologous Group of Proteins) [12] and KO (KEGG Ortholog) [13]. The composition shows that more than 85% of the result is similar to the KO and COGs. Note that KO and COGs have been constructed manually or semi-manually, and they provide accuracy but need long time to accommodate newly complete genomes.

In conclusion, our system provides automatic clustering of orthologous groups with high speed and reasonable accuracy. It also has a good extensibility in accommodating newly completed genomes.

The paper is organized as follows. In Section 2, we introduce related work. In Section 3, we describe the method and algorithm for ortholog clustering in detail. In Section 4, we present system architecture. In Section 5, we show system performance and accuracy. In Section 6, we present conclusion and future works.

2 RELATED WORK

2.1 COGs (Clusters of Orthologous Group of Proteins)

This work was supported by the Regional Research Centers Program of the Ministry of Education & Human Resources Development in Korea.

The COGs [6, 12] released in 1999 is the second database constructed on the basis of sequence databases in NCBI. COGs provides the results clustered with the some complete genomes such as bacteria, archaea and eukaryote by using the ortholog concept. COGs are constructed as the following phases; first perform the all-against-all protein sequence comparison for genome sequences using by BLAST [3, 4], and then if ortholog in the 3 lineage at least exists, composite one orthologous group. The COGs has the advantage in classifying many complete sequences and adding the new complete sequences easily [7]. However, COGs requires manual works to reduce the paralogs and false positive, it has a limitation in automation to complete the whole processes.

2.2 KO (KEGG Orthology)

KO [13] consists of the orthologous genes extracted from metabolic pathways and regulatory pathways [10]. KO has higher accuracy classifying the orthologous groups in the manual work. Note that KO has difficulty in accommodating new genomes.

2.3 INPARANOID

The INPARANOID [8] presents the method to cluster orthologs and in-paralogs from pairwise species comparisons automatically. The INPARANOID is useful for reduction of the false-positive and detection of in-paralogs which have the same function as that of orthologs. However, it can't be extended to cluster orthologs from multiple species.

2.4 Grid Computing

Grid computing has been primarily adapted in scientific problem solving such as large scale of data analysis. In bioinformatics area, grid is mainly being applied in sequence alignment [11] and protein secondary structure prediction [9] in which large scale sequence data is involved. Not only are there so many works to process but also each of them takes much time to be completed in these operations. That is why it is burdensome and limited to process them in a single or parallel system. So, the grid computing is useful for solving problems like these.

3 CLUSTERING OF ORTHOLOGOUS GROUPS

In this section, we present the method and algorithm to cluster orthologous groups. To describe method and algorithm in detail, let assume the following notations in Fig.1.

Notation
G_i : *The genome of species i*
G : *Set of genomes*, $G_i \in G$
GN_i : *The number of genes of* G_i
N : *The number of species*
g_i^j: j^{th} *gene of Genomes* G_i
OG : *The set of Orthologous group*
$CR(G_i, G_j)$: *The result of compare between* G_i *and* G_j, $i \leq j$

Figure 1: Notation

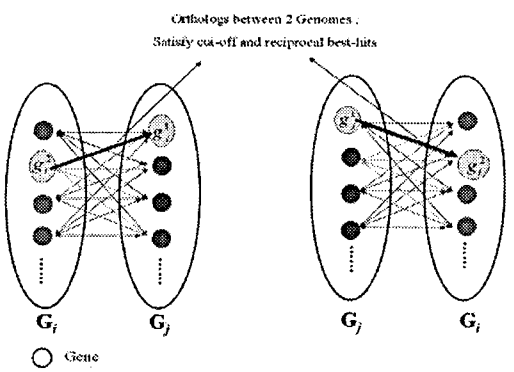
3.1 Search the reciprocal best hits between two genomes

To construct the orthologous group, we have to find reciprocal best hits between two genomes. INPARANOID [8] has the similar process. We'll describe our search method step by step.

First, we prepare protein sequences input-data in multiple FASTA formats. Input-data are expected to include the complete set of protein sequences from variable species. Incomplete sets of genes may result in an incorrect list of orthologs.

Second, we compare similarity between two genomes using BLAST from prepared data. For two genomes, G_i and G_j, we compare all-against-all between data set G_i-G_j and G_j-G_i with BLAST, and then search all pairwise similarity scores between two genomes. In order to reduce false-positive of the outcome of BLAST, we use score cut- off and overlap cut-off. Score cut-off is used to exclude spurious matches. We normally use a score cut-off of 50 bits and overlap cut-off of 50 percent.

Third, we find all pairwise similarity scores that satisfy the cut-off score between two genomes, and then search the reciprocal best hit to delete paralogs. Fig.2 shows search and result for reciprocal best hits between two genomes with applying cut-off value.



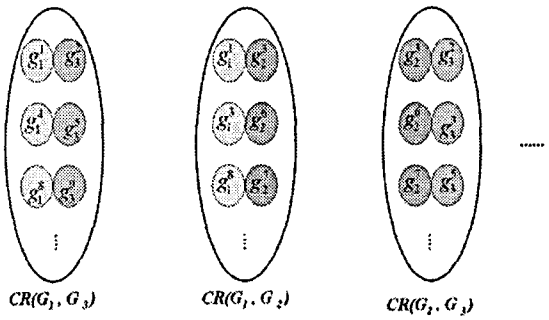


Figure 2: search result of reciprocal best hits between 2 genomes

4 SYSTEM ARCHITECTURE

In this section, we present the system architecture for the clustering of orthologous groups. Fig.3. shows two parts of the system architecture; Hardware-Oriented Grid Computing (HGBS) and Ortholog Clustering Engine (OCE). HGBS searches reciprocal best hits between species and OCE is the part to cluster orthologous groups.

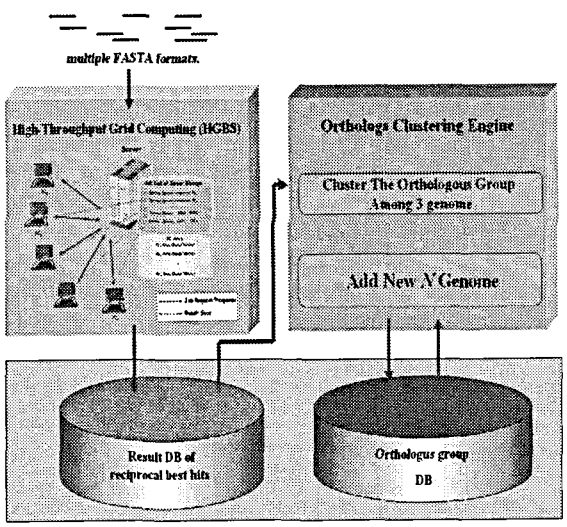


Figure 3: System architecture

4.1 HGBS (A hardware-oriented grid BLAST system)

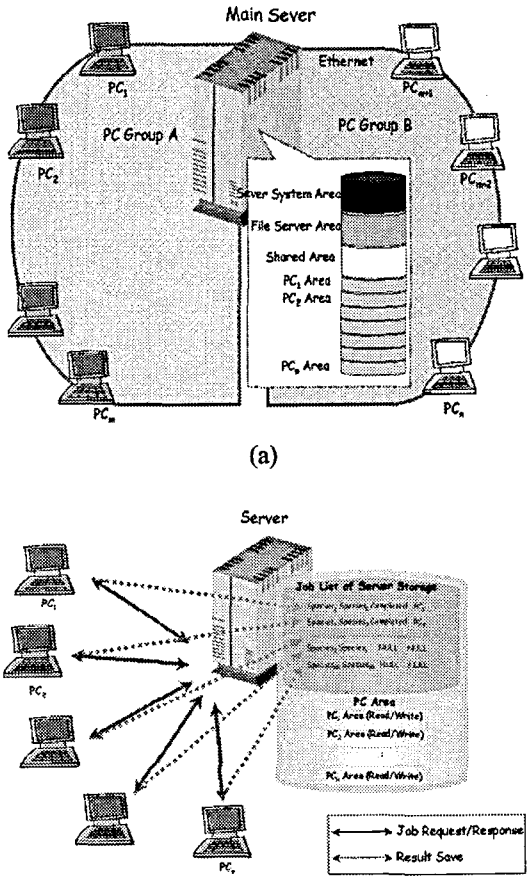
If the number of genome is n, we should operate $n(n\text{-}1)/2$ genomes-genome comparisons to search all reciprocal best hits by BLAST. In each comparison of genomes, G_p and G_q, it requires gene to gene comparisons by the BLAST operation as many as the Eqs.1.

$$f(G_p,G_q) = (\sum_{i-1}^{GN_p}\sum_{j-1}^{GN_q} compare\ (g_p^j, g_q^j)) * 2 \qquad (1)$$

Then, the number of the overall BLAST operations becomes as Eqs.2.

$$F = (\sum_{i-1}^{N-1}\sum_{j=1}^{GN_i}\sum_{k=1}^{GN_{i+1}} compare\ (g_i^j, g_{i+1}^k)) * 2 \qquad (2)$$

Therefore it takes lots of time to process numerous genomes. To solve this problem, we developed HGBS and utilize it in our system [2]. HGBS is the system to process the sequence alignment rapidly, using many PCs in grid environment like Fig.4.



(a)

(b)

Figure 4: (a) System architecture of HGBS and (b) the principle of its operation

As shown in Fig.4(a), we construct the grid environment having 27 PCs in two computer rooms (PC Group A and PC Group B). Each PC boots the Linux OS through reading their own area of the main server via network. Each PC brings and processes the jobs with data and then saves the results in the server node as shown at Fig.4(b). The jobs are maintained in the queue of main server like Fig.4(b) and when requested by the PCs, the main server gives the jobs to the PCs. There are some properties to check the state of the jobs; the job-id, the data list that will be processed, the state whether the job is completed, processing, or idle, job processing start time and stop time, and the PC number. These properties are useful when scheduling the jobs.

The process in HGBS is executed repeatedly until all comparisons are done. The result of the reciprocal best hits is stored into the database. Fig.5. shows result of $CR(G_1, G_2)$ on HGBS. Table consists of uid, geneID, score, and flag. Flag in the table is to identify whether the gene is included in the OG or not.

uid	geneID1	geneID2	score	flag
1	15891167	15606697	1102	NULL
2	15889250	15606949	1006	NULL
3	15888838	15606094	920	NULL
4	15889249	15606950	914	NULL
5	15889244	15605613	719	NULL
6	15888590	15605788	661	NULL
7	15890799	15606774	644	NULL
8	15889173	15606507	634	NULL
9	15890895	15606119	626	NULL
10	15888619	15606309	609	NULL
11	15887591	15606804	599	NULL
12	15888029	15605834	594	NULL
13	15889291	15606895	590	NULL
14	15889025	15606596	589	NULL
15	15887371	15606321	574	NULL
16	15887476	15606302	573	NULL
17	15889457	15607056	573	NULL
18	15888025	15607130	560	NULL
19	15889879	15606090	560	NULL
20	15889997	15605863	538	NULL

Figure 5: Result table of reciprocal best hits on HGBS

4.2 Orthologous groups clustering engine

After finding all reciprocal best hits between two genomes through HGBS, OCE (orthologous group clustering engine) clusters orthologous groups automatically from result table. Our clustering process uses the database in order to automate all steps without manual-works. All result of clustering is stored in the database and the clustering algorithm is performed through the query. A lot of selection, insertion, update, and join operation arise in this work. Therefore, the execution by the database is necessary. Fig.6 shows the clustering process using by the database.

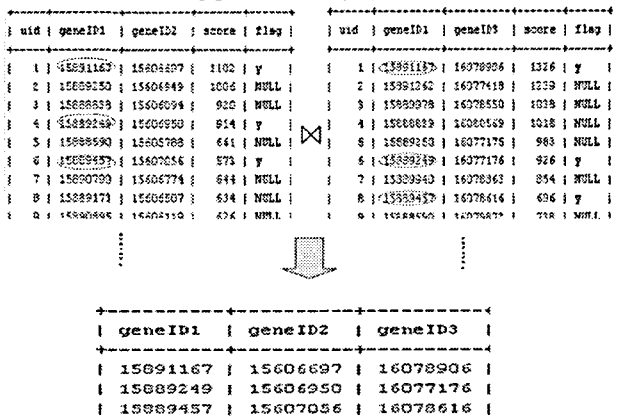
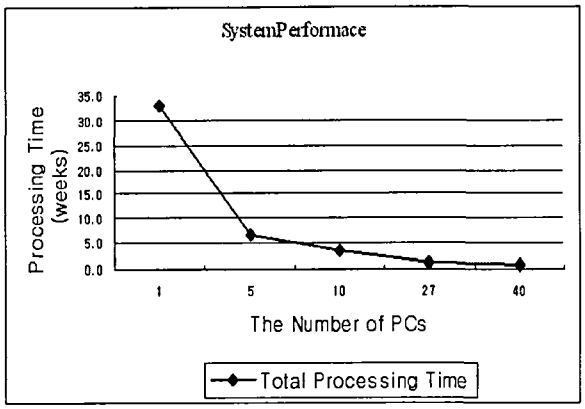


Figure 6: clustering process using by database

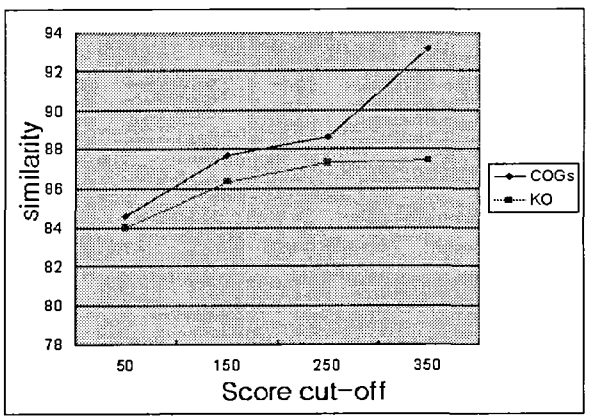
As shown in Fig.6, result of reciprocal best hits $CR(G_1, G_2)$ and $CR(G_1, G_3)$ are joined to search common genes. Flag column updates NULL to 'y' to mark common genes and insert common genes found in *OG* table. All process can be executed efficiently without manual work by the clustering algorithm.

5 EVALUATION

We evaluated performance of the grid computing and accuracy of the constructed orthologous groups. 89 prokaryotes complete genomes data has been used in the experiment. Fig.7(a) shows the interrelation between the number of PCs and the total processing time. As the number of PCs increases, the processing time decreases from 33 weeks to less than 1 week. For accuracy comparison, we selected 20 genomes contained in COGs and KO, then generated orthologous groups by using our system with the cut-off scores from 50 bits to 250 bits. Fig.7(b) shows that our result has more than 85% similarity in the comparison with the results of COGs and KO. As the cut-off value increases, the similarity approaches 93% in the case of COGs.



(a)



(b)

Figure 7: (a) Grid computing performance in processing pair-wise sequence comparison and (b) comparison with COGs and KO for various score cut-off values.

6 CONCLUSION AND FUTURE WORK

We proposed an ortholog clustering system based on grid computing and verified its performance and accuracy. Our system has several advantages to construct the orthologous groups. Our system constructs the orthologous groups automatically and rapidly by using an elaborate clustering algorithm on a grid computing environment. Note that most precious systems adopted manual or semi-automated approaches. As the genome sequences are pouring, rapid construction of orthologous groups in an automatic fashion is becoming an important issue. We believe that our system will be a solution for this problem.

In the future, we will construct the orthologous groups for 89 complete genomes including eukaryote. Also we will increase the functional accuracy of the orthologous groups.

REFERENCES

[1] Fitch, W. M. (1970). Distinguishing homologous from analogous proteins. Syst. Zool. 19, 99-113.

[13] Kim T.K., Oh S.K., Lee K.H., Roh D.H., and Cho W.S., "HGBS : A Hardware-Oriented Grid BLAST System,"in *Proc. of the 5th IEEE/ACM Int'l Symposium on Cluster Computing and the Grid, BioGrid 2005.*

[6] S. F. Altschul, et al., "Basic Local Alignment Search Tool," Journal of Molecular Biology, 215:403-410, 1990.

[7] S. F. Altschul et al., "Gapped BLAST and PSI-BLAST: A new generation of protein database search programs," Nucleic Acids Research, 25:3389-3402, 1997.

[2] Tatusov, R. L., Koonin, E. V. and Lipman, D. L. 1997. A genomic perspective on protein families. Science 278, 631-637.

[4] Tatusov, R. L., Natale, D. A., Garkavtsev, I. V., Tatusova, T. A., Shankavaram, U. T., Rao, B. S. et al.(2001). The COG database: new developments in phylogenetic classification of proteins from complete genomes. Nucl. Acids Res. 29, 22-28.

[5] Tatusov, R. L., Galperin, M. Y., Natale, D. A. & Koonin, E. V. (2000). The COG database: a tool for genome-scale analysis of protein functions and evolution. Nucl. Acids Res. 28, 33-36.

[10] Remm, M., Storm, C.E., and Sonnhammer, E.L. 2001. Automatic clustering of orthologs and in-paralogs from pairwise species comparisons. J. Mol. Biol. 314: 1041-1052.

[12] Soojin Lee, et al., "Exploring protein fold space by secondary structure prediction using data distribution method on Grid platform," Bioinformatics, Advance Access published on July 29, 2004.

[9] Yamanishi, Y., Akiyasu C. Yoshizawa, Itoh, M., Katayama, T., Kanehisa, M., "Extraction of Organism Groups from Whole Genome Comparisons" , Genome Informatics 14, 438-439, 2003.

[11] Y. L. Kuo, et al., " Construct a Grid Computing Environment for Bioinformatics," In Proc. of the International Symposium on Parallel Architectures, Algorithms and Networks(ISPAN'04), 1087-4089, 2004.

[3] COGs official homepage. http://www.ncbi.nlm.nih.gov/COG/

[8] KO official homepage. http://www.genome.jp/kegg/ko.html

APPENDIX

Clustering of Orthologous Groups

After finding the reciprocal best hits about all of the genomes, and then we make orthologous group by clustering the results of it. We devised a new automatic clustering algorithm without manual work. Fig.8 shows a summery of our clustering algorithm for *n* genomes.

When clustering *n Genomes*
Algorithm ClusteringGenomes

```
OrthologClustering among 3 Genome
for m=4 ... n
  OrthologClustering
end for
```

Figure 8: clustering algorithm for *n* genomes

Our ortholog clustering algorithm consists of two-phase. The first phase is to make the orthologous group among 3 genomes from the results of reciprocal best hits, because we want to classify the orthologous group that includes more than the 3 lineage like the COGs. Fig.9 (a) shows orthologous group clustering algorithm among 3 genomes. Let assume optional 3 genomes G_1, G_2 and G_3. Result of reciprocal best hit for G_1, G_2, and G_3 are $CR(G_1, G_2)$, $CR(G_1, G_3)$ and $CR(G_2, G_3)$. As shown 2, 12, 17 lines in Fig.4 (a), search the common genes from $CR(G_1, G_2)$, $CR(G_1, G_3)$ and $CR(G_2, G_3)$, and then mark the common genes found like 3, 6, 9, 12, 15, 18 line in Fig.9(a). Since the common genes found don't need reference anymore which are excluded in the next process. Result of common genes saves in *OG* as 4,7,10,13,16,19 lines. 5, 8, 14 lines are process to add result of reciprocal best hits which are not contain in *OG* on above process.

The second phase is the work to add new n_{th} $(n> 3)$ genome from 3 genome *OG* (first phase) and $CR(G_i, G_j)$ which not include in *OG*. This phase is shown in Fig.9(b). 2 to 11 lines are process to search and add common gene in result of reciprocal best hits between *OG* made by first phase.

From line 13, we cluster the new OGs with reciprocal best hit genes through the sequence comparison based on not included genes in OG. This is similar to the previous clustering method. We perform this process to add as many genes into OGs as possible.

Algorithm OrthologClustering among 3 Genomes
Require: *OG* and $CR(G_1, G_2)$, $CR(G_1, G_3)$, and $CR(G_2, G_3)$

```
1  begin
2    search the common genes of CR(G1, G2) and CR(G1, G3)
3    mark the common genes of CR(G1, G2) and CR(G1, G3)
4    save the result in OG

5    search the common genes of OG and CR(G2, G3)
6    mark the common genes of CR(G2, G3)
7    save the result in OG

8    search the common genes of OG and CR (G2, G2)
9    mark the common genes of CR(G2, G2)
10   save the result in OG

11   search common genes of CR(G1, G2) and CR(G2, G3)
12   mark the common genes of CR(G1, G2) and CR(G2, G3)
13   save the common genes in OG

14   search the common genes of OG and CR(G1, G3)
15   mark the common genes of OG and CR(G1, G3)
16   save the common genes in OG

17   search the common genes of CR(G1, G3) and CR(G2, G3)
18   mark the common genes of CR(G1, G3) and CR(G2, G3)
19   save the common genes in OG
20 end
```

(a)

```
Algorithm OrthologClustering
Require: OG and CR(G_i, G_j)
1 begin
2   for i = 1 ... i = m-1 do
3     search the common genes of OG and CR(G_i, G_m)
4     mark the common genes of CR(G_i, G_m)
5     save the common genes in OG
6   end for

7   for i = 2 ... i = m-1 do
8     search the common genes of OG and CR(G_i, G_m)
9     mark the common genes of CR(G_i, G_m)
10    save the common genes in OG
11  end for

12 for i = 1 ... i = m-2 do
13   for j = i+1 ... j = m-1 do
14     search the common genes of CR(G_i, G_j) and CR(G_i, G_m)
15     mark the common genes of CR(G_i, G_j) and CR(G_i, G_m)
16     save the common genes in OG

17     for k = 1 ... k = m-1 do
18       if i != k then
19           search the common genes of OG and CR (G_k, G_m)
20           mark the common genes of OG and CR (G_k, G_m)
21           save the common genes in OG
22           if j == k then
23              search the common genes of OG and CR (G_k, G_m)
24              mark the common genes of OG and CR (G_k, G_m)
25              save the common genes in OG
26           end if
27       end if
28     end for
29   end for
30 end for
31 for i = 1 ... i = m-2 do
32  for j = i+1 ... j = m-1 do
33    search the common genes of CR(G_i, G_j) and CR(G_j, G_m)
34    mark the common genes of CR(G_i, G_j) and CR(G_j, G_m)
35    save the common genes in OG

36    for k = 1 ... k = m-1 do
37      if j != k then
38         search the common genes of OG and CR (G_k, G_m)
39         mark the common genes of OG and CR (G_k, G_m)
40         save the common genes in OG
41          if i == k then
42            search the common genes of OG and CR (G k, G_m)
43               mark the common genes of OG and T CR (G k, G_m)
44               save the common genes in OG
45          end if
46       end if
47     end for
48   end for
49  end for

50  for i = 1 ... i = m-2 do
51    for j = i+1 ... j = m-1 do
52      search the common genes of CR(G i, G_m) and CR(G j, G_m)
53      mark the common genes of  CR(G i, G_m) and CR(G j, G_m)
54      save the common genes in OG

55      for k = j+1 ... k = m-1 do
56         search the common genes of OG and CR(G k, G_m)
57         mark the common genes of OG and T CR(G k, G_m)
58         save the common genes in OG
59       end for
60     end for
61  end for
62 end
```

(b)

Figure 9: (a) clustering algorithm among 3 genomes, (b) add new *n (n > 3)* genomes from 3 genomes *OG* and *CR (G_i, G_j)* which not include in *OG*