A Graph-Based Clustering Method for a Large Set of Sequences Using a Graph Partitioning Algorithm

Hideya Kawaji[1,2]
`kawaji@ics.es.osaka-u.ac.jp`

Yosuke Yamaguchi[1]
`y-yamagu@ics.es.osaka-u.ac.jp`

Hideo Matsuda[1]
`matsuda@ics.es.osaka-u.ac.jp`

Akihiro Hashimoto[1]
`hashimoto@ics.es.osaka-u.ac.jp`

[1] Department of Informatics and Mathematical Science, Graduate School of Engineering Science, Osaka University, 1-3 Machikaneyama, Toyonaka, Osaka 560-8531, Japan
[2] NTT Software Corporation, 223-1 Yamashita-cho, Naka-ku, Yokohama-shi, Kanagawa 231-8554, Japan

Abstract

A graph-based clustering method is proposed to cluster protein sequences into families, which automatically improves clusters of the conventional single linkage clustering method. Our approach formulates sequence clustering problem as a kind of graph partitioning problem in a weighted linkage graph, which vertices correspond to sequences, edges correspond to higher similarities than given threshold and are weighted by their similarities. The effectiveness of our method is shown in comparison with InterPro families in all mouse proteins in SWISS-PROT. The result clusters match to InterPro families much better than the single linkage clustering method. 77% of proteins in InterPro families are classified into appropriate clusters.

Keywords: protein family, clustering, weighted linkage graph, graph partitioning

1 Introduction

The number of protein sequences in public databases grows rapidly with the progress of experimental technologies in molecular biology and large genome projects recently. The large size of sequence data makes it difficult to know relationships among a large set of sequences. It is widely used to cluster large data into meaningful or manageable groups [1, 2]. A number of clustering methods for molecular sequences based on sequence similarities with their homologues, which is computed by homology search programs such as BLAST [3] and FASTA [4], have been proposed [5, 6, 7]. Identification of protein families by clustering based on sequence similarity is a traditional problem in molecular biology. Genome-scale sequence clustering is important as a basis of functional, structural, or other additional analyses.

InterPro [8] is an integrated resource for protein families, domains and sites. Its entries are linked from proteins in SWISS-PROT database [9], which is one of the most annotated protein databases. In InterPro, families are denoted by widely used protein signature databases, such as Pfam [10], PRINTS [11], PROSITE [12], ProDom [13], and SMART [14]. These databases have their own vital tools for classifying novel sequences into their entries. Such tools owe to the different strengths and weaknesses of their underlying analysis methods. It implies that there may remain members of known protein families and novel families.

One of the most widely used clustering methods is the single linkage clustering [15, 16]. It is a kind of hierarchical clustering, and its cluster relationships can be represented by a rooted tree which is also called dendrogram. A cluster is produced by cutting edges of the dendrogram with a threshold. Each family has a different threshold of similarity that is the most appropriate, because there are many

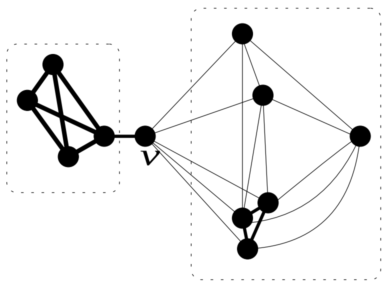

Figure 1: An example of a linkage graph. A vertex corresponds to a sequence, a thick edge denotes high similarity between two sequences in its both end points, and a thin edge denotes low similarity. Absence of edges between any two vertices denotes that the similarity between them is lower than a given threshold. Regions surrounded by broken line denotes distinct clusters.

variations in diverged level or length of sequences. The single linkage clustering method is known as to produce an accurate family in many cases when an appropriate threshold is given. There is a difficulty to give such a threshold for each family. Application of only one threshold for all clusters would produce many too small clusters and a few large clusters.

There is another problem in the single linkage clustering that a sequence may be more similar to the other family member than some members of the same family by chance. Such situation is shown in Figure 1, which could happen because sequence similarities are not metric. It is a weighted linkage graph, which vertices correspond to sequences, edges correspond to higher similarities than given threshold and are weighted by their similarities. Sequence v is more similar to just one member of left cluster rather than right cluster members. It can be thought that v is similar to the left cluster member in residues except for conserved residues in the left cluster, and share conserved residues with the right cluster members. Thus it would be better that v belongs to the right cluster, in which member sequences are moderately similar to v. However, v is classified in the left cluster by the single linkage clustering because of the most similar sequence.

To cope with the two issues, employment of a linkage graph is essentially effective because it enables us to cluster in consideration of not only similarities between members in just a cluster but in the others. A clustering method based on a linkage graph was proposed to classify multi-domain structure sequences [7], using an unweighted linkage graph, which clusters would overlap each other because of the nature of multi-domain sequences. Although domain-based clustering is very important problem recently, we are not concerned here with it. Our aim of this paper is to classify sequences into protein families more accurately. In our approach, clustering sequences into protein families is formulated as a kind of graph partitioning problem of a weighted linkage graph. In a connected graph there is a set of edges whose removal disconnects it, which is called *cut* and its *weight* is defined as the sum of cut edge weights. Graph partitioning with the minimum cut weight into distinct non-empty sets of vertices is employed to cluster mRNA expression data [17]. Computation of the minimum cut in a weighted graph needs polynomial time and does not need any initial partitions. Graph partitioning with the minimum cut weight into balancing distinct two sets is known as an NP-complete problem [18]. One of the best known heuristic algorithms is the Kernighan-Lin algorithm [19]. It is a simple local descent algorithm, which works well to improve an already reasonable initial partition. It swaps two vertices in distinct sets, iteratively. Each iteration takes $O(|N|^3)$ time ($|N|$ is the number of vertices). A more complicated and efficient implementation, which takes $O(|E|)$ time per iteration ($|E|$ is the number of edges), was proposed by Fiduccia and Mattheyses (the FM algorithm [20]). It can classify sequences into accurate families if an appropriate balancing parameter or the number of each family members is given. There is a difficulty to give the parameter before each family is identified.

Based on the FM algorithm, we propose a graph partitioning algorithm without the balancing parameter of clusters, which uses the first minimal cut weight as a criterion of partitioning and may produce an empty set in some cases. Our sequence clustering method applies the graph partitioning algorithm to an initial partition, which is produced by the single linkage clustering, iteratively.

In Section 3, our clustering method is applied to all mouse protein sequences in SWISS-PROT [9] and the result is compared with InterPro [8] families, for demonstration of its effectiveness.

2 Method

2.1 Preliminaries

Definition 1 Given a set of sequences $Seq = \{seq_i | 1 \leq i \leq n\}$, and their pairwise similarities $Sim = \{s(x,y) | x, y \in Seq, s(x,y) \geq 0\}$, and a threshold c. A *weighted linkage graph* is an undirected graph, defined as $G = (V, E, W_E)$, $V = \{v_i | 1 \leq i \leq n\}$, $E = \{(v_i, v_j) |\ v_i, v_j \in V,\ s(seq_i, seq_j) \geq c\}$, $W_E = \{s(v_i, v_j) = s(seq_i, seq_j) |\ (v_i, v_j) \in E\}$.

Definition 2 A *bipartitioning* of a set V is $P = (A, B)$ such that $A \subseteq V,\ B = V - A$. Hereafter we refer to bipartioning as *partitioning.*

Definition 3 Given a weighted linkage graph $G = (V, E, W_E)$ and a partitioning $P = (A, B)$. A *cut* is a set of edges interconnecting A and B, defined as

$$cut = \{(x,y) |\ (x,y) \in E,\ x \in A,\ y \in B\}$$

And *cut weight* of (A, B) is defined as

$$cut\_weight(A,B) = \sum_{(x,y) \in cut} s(x,y)$$

Definition 4 Given a weighted linkage graph $G = (V, E, W_E)$ and a partition $P = (A, B)$. The *gain* of a vertex $a \in A$ is defined as

$$gain(a) = \sum_{b \in B} w(a,b) - \sum_{a' \in A, a' \neq a} w(a,a') \qquad (1)$$

Conversely, for $b \in B$,

$$gain(b) = \sum_{a \in A} w(a,b) - \sum_{b' \in B, b' \neq b} w(b,b') \qquad (2)$$

$gain(a)$ means a decrease of the cut weight when a moves from A to B, i.e., the following equation holds:

$$gain(a) = cut\_weight(A,B) - cut\_weight(A - \{a\}, B \cup \{a\})$$

Note that eq. (1) and eq. (2) imply that a movement of v between A and B changes the gains of just vertices adjacent to v, not all gains.

Definition 5 Given a set of sequences Seq and their pairwise similarities Sim. The similarity of two clusters P and Q is defined as $s(P,Q) = max\{s(x,y) | x \in P,\ y \in Q\}$. A *SL tree* (Single Linkage tree, or dendrogram) is computed as follows. A set of clusters SC is used.

Step 1. Make n singleton clusters and produce n nodes corresponding to the singleton clusters.

Step 2. Repeatly to merge the most similar two clusters P and Q and produce node m, of which children are nodes corresponding to P and Q, until a single cluster remains. The merged cluster of P and Q is referred as $SLC(m)$, and $s(P,Q)$ as the similarity at node m.

2.2 Graph Partitioning Algorithm

Our method aims to improve partitioning of a connected weighted linkage graph from an initial one into a partitioning which does not divide any families, in this section. The FM algorithm would make a good partition when an appropriate balancing parameter is given, because it makes a cluster in which the number of members is appropriate and the members are moderately similar to each other by making partitioning with a minimal cut weight. However, such a balancing parameter can not be given before clustering.

We designed a graph-partitioning algorithm to produce a partition with a minimal cut weight from an initial one, without any balancing parameters. The FM algorithm repeats 'one-pass improvement', which searches for a partitioning with the minimum cut weight while all vertices are moved from one set to the other, keeping the balance of the two sets. Removal of the balancing parameter from the FM algorithm produces a partition with an empty set and a set of all members necessarily, because there is always a partition with the minimum cut weight, 0, in search space of the algorithm. It implies that limitation of search space enables us to produce a partition with non-empty sets in some cases.

Our graph partitioning algorithm is described below. A weighted linkage graph $WLG = (V, E, W_E)$, which is expressed in the *adjacency list* representation, and an initial partition (A, B) are given. Gains of vertex v_i, $gain[i] (1 \leq i \leq |V|)$, a list of vertices P, and a list of gains $G = \{g_1, g_2, ...\}$, are used as working area.

```
(1)  **repeat**
(2)      $P := \{\}$, $G := \{\}$, $GAIN := 0$.
(3)      Compute $gain[i]$ for all vertices.
(4)      **repeat**
(5)          Select an unmarked vertex $v_i$ having the highest gain, and mark $v_i$.
(6)          Append $v_i$ to the last of $P$, and $gain[i]$ to the last of $G$.
(7)          Update the gains of $v_i$ and vertices adjacent to $v_i$, as if $v_i$ had been moved.
(8)      **until** All vertices are marked.
(9)      Find $k$ such that $\sum_{i=1}^{k} g_i$ is the first maximal value.
(10)     **if** $\sum_{i=1}^{k} g_i > 0$
(11)     **then**
(12)         Move vertices $\{p_1, ..., p_k\}$ to the other set of $(A, B)$
(13) **until** $\sum_{i=1}^{k} g_i \leq 0$
```

Figure 2 shows an example of a process 'one-pass improvement', from (2) to (12) in the algorithm. After initialization in (2), a vertex c with the highest gain is marked and appended to P, and its gain is appended to G. The gains of c, e, f and b are updated, as if c had been moved. Such processes are iterated until all vertices are marked. And $k = 2$ is selected as the first maximal value $\sum_{i=1}^{k} g_i$, which corresponds to a partition with the first minimal cut weight, and vertices c and d are moved. The partitioning algorithm returns a solution consisting of an empty set and a set of all vertices in some cases. Such situation is regarded that it should not be partitioned. It gives a criterion of the number of clusters as described in Section 2.3. It is important to note that the cut weight is decreased necessarily in one-pass improvement. It implies that the algorithm stops for any input, because the cut weight must be more than zero.

We do not use special data structure, such as a bucket list. It enables the FM algorithm to select an unmarked vertex with the highest gain in constant time, and to be executed in $O(|E|)$ time per iteration. It is only possible by restriction of the degree of vertices in a graph to a constant value. Thus, the computational time of the selection of the highest gain vertex takes $O(|V|)$ time, because gains of vertices adjacent to moved vertices are updated per iteration of inside repeat, the number is $|V|$ in the worst cases. Thus, our graph partitioning algorithm takes $O(|V|^2)$ time per one-pass

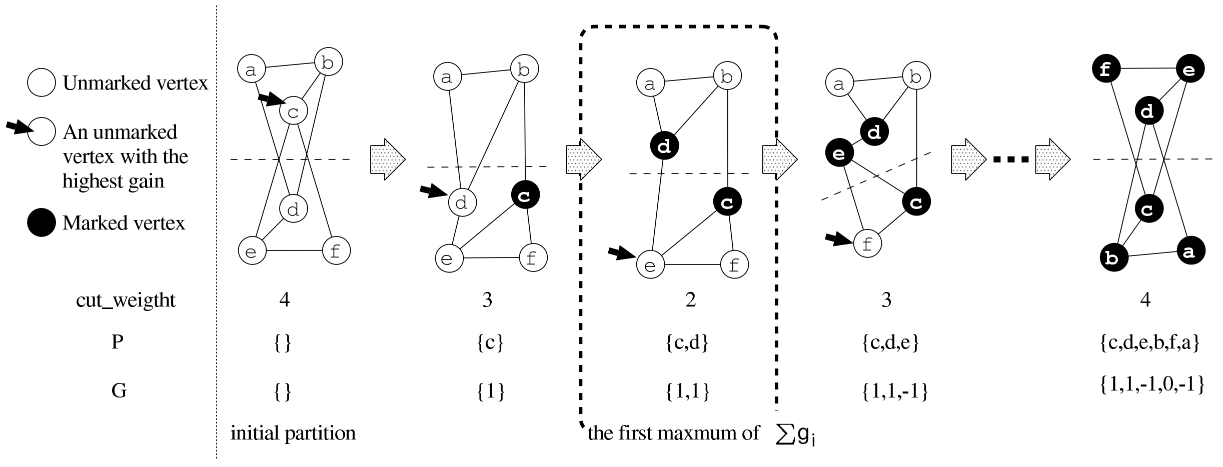

Figure 2: An example of 'one-pass improvement' in graph partitioning process. For simplicity, all edges are weighted by 1. Only a partition with the first maximal $\sum g_i$ are selected in practice.

improvement iteration. The number of one-pass improvement iterations is very small and near to constant in practice. The result of application to all mouse proteins is shown in Section 3. With regard to memory used in the computation, an adjacency list of WLG, which uses the largest space, takes $O(|V| + |E|)$ space.

2.3 Iterative Graph Partitioning

The graph partitioning algorithm presented in the previous section produce two clusters, one of which may be empty in some cases. Its iterative applications are needed in order to produce more than two clusters. In addition, a reasonable initial partitioning should be given for applying the algorithm because it improves the initial one. To cope with these problems, we took an approach based on a dendrogram produced by the single linkage clustering method, a SL tree. It is based on our assumption that a single linkage cluster with the highest threshold is included by or resemble in a protein family, and reasonable as an initial partition to be improved. Giving priority to cluster groups whose members have high similarities to each other enables us to cluster without consideration of various similarity thresholds depending families. Our algorithm to iterate partitioning is described bellow. A connected weighted graph $WLG = (V, E, W_E)$ is given. SLT is used as a SL tree of WLG, and N is used as a list of SLT nodes except for leaves and the root in descending order of their similarities. In addition, (A, B) is used as a working area of a partition of V.

```
(1)  Make $SLT$ for $WLG$, and $N$ for $SLT$.
(2)  **while** there exist any nodes in $N$.
(3)       Remove $n$ from the head of $N$.
(4)       Improve an initial partition $(SLC(n), V - SLC(n))$ into $(A, B)$
               using our graph partitioning algorithm described in Section 2.2.
(5)       **if** $cut\_weight(SLC(n), V - SLC(n)) > cut\_weight(A, B),\ A \neq \phi, B \neq \phi$
(6)       **then**
(7)            Output $A$ as a cluster.
(8)            Delete vertices in $A$ and their adjacent edges from $WLG$.
(9)            Update $SLT$ for new $WLG$, and $N$ for new $SLT$.
(10) Output a set of vertices in $WLG$ as a cluster.
```

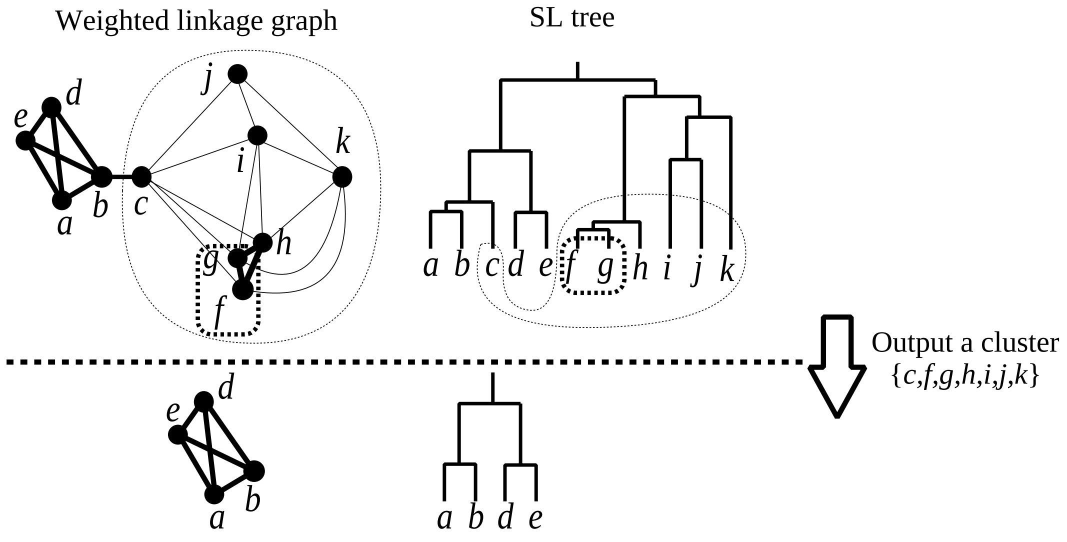

Figure 3: An example of a weighted linkage graph and its SL tree, assuming that a similarity between g and f is the highest in all vertex pairs.

An example of clustering process is showed in Figure 3. Taking a node of SLT from the head of N, and we make the initial partition $(\{f, g\}, \{a, b, c, d, e, h, i, j, k\})$. One of the two sets, $\{f, g\}$, is a single linkage cluster, which similarity is the highest in N. The partition is improved into $(\{c, f, g, h, i, j, k\}, \{a, b, e, d\})$. The improved set, $\{c, f, g, h, i, j, k\}$, is outputted as a cluster of our algorithm. Vertices in the cluster and their adjacent edges in WLG are removed, and SLT and N are updated for the next iteration.

Each iteration process removes a node from N, and the number of iteration is $|N|$, equal to $|V| - 2$, in the worst case. These processes are iterated until there are not any nodes in N. An adjacency list of WLT and SLT, which uses the largest space, takes $O(|V| + |E|)$ space.

2.4 Sequence Clustering Method

The iterative graph partitioning method described in Section 2.3 should be applied to a connected weighted linkage graph. Thus, our method of sequence clustering is described as follows:

1. Compute homologues of all sequences by homology search.
2. Construct a weighted linkage graph with a sequence similarity threshold c, and divide it into connected components.
3. Apply the iterative graph partitioning method to each connected components.

3 Results

For evaluation of our method, all mouse proteins in SWISS-PROT, Release 39.17 of 27-Apr-2001, was used, which includes 4,482 entries. the BLAST program was executed for each protein against all with e-value threshold 0.1. Our clustering method using a graph partitioning algorithm, hereafter we refer to it as CGP, was applied to the mouse proteins with bit score 33 as a threshold. It made 374 clusters, which include three or more protein sequences. There are 3,415 distinct proteins (about 76%) of all. In addition, the single linkage clustering method was applied to the same data using bit score 33 and 100 as a threshold. It made 126 and 242 clusters, which included three or more protein

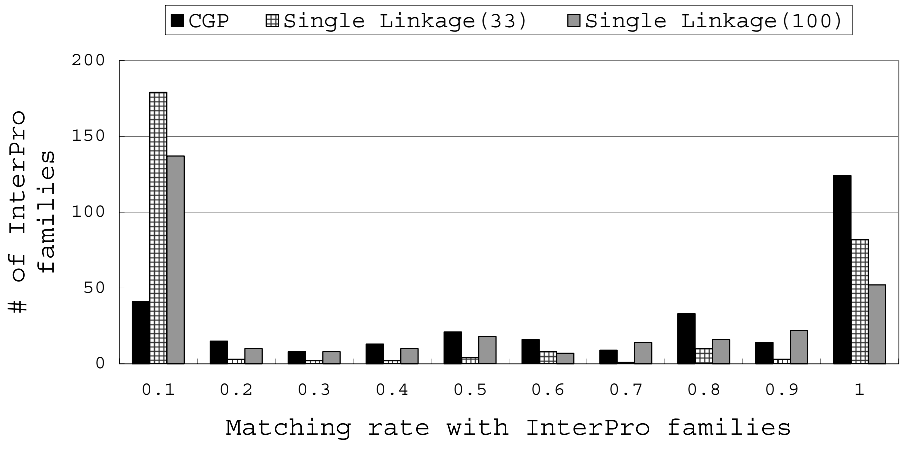

Figure 4: The number of the InterPro families and matching rate with their match. A figure in brackets indicates a threshold used in the single linkage clustering.

sequences, with 3,632 (about 81%) and 3,225 (72%) proteins respectively. Their results were compared with the InterPro cross-references in the SWISS-PROT entries. 4,033 proteins have cross-references to 1,369 InterPro entries. Only InterPro families, not domain or repeats, with three or more members in the mouse proteins was extracted from 1,369 InterPro entries. As a result, 2,147 (48%) of all mouse proteins have cross-references to 294 InterPro families. There are 549 proteins belonging to two or more InterPro families, because of their overlapping.

As an example of clustering results, we focus on the serine proteases, trypsin family (IPR001254). It matches to the largest clusters, 2,920 members and 185 members, in the single linkage clustering with bit score 33 and 100 as thresholds, respectively. The cluster with a threshold 33 include not only the family but also the others. The cluster with a threshold 100 misses three members of the family, complement factor B (P04186), complement C2 (P21180), mast cell protease 3 (P21843), while a cluster of the CGP method overlaps the family completely. It indicates that the CGP method classified the serine proteases, trypsin family accurately, which is impossible to be classified by the single linkage clustering method. To compare a set of clusters with InterPro families, *matching rate* is defined as $|F \cap C|/|F \cup C|$, in which F is a family and C is a cluster. A cluster C is referred to *match* to F when C has the highest matching rate in all clusters made by a clustering method, conversely a family F as match to C when F has the highest matching rate in families. Figure 3 shows the number of InterPro families and their matching rate of their matches in three result of clustering. It implies that the matching rates with the CGP clusters match to the InterPro families much better than with the single linkage clustering method if its threshold is changed. There are 124 (42%) of the InterPro families with matching rate higher than 0.9 in the CGP clusters.

There are some overlapping families in the InterPro, it is difficult to compare InterPro families and the CGP clusters directly. Thus we call F and C *matching pair* in both directions when F matches to C, and C matches to F. There are 213 matching pair in both directions, which are 72% of all 294 InterPro families. The 213 families and CGP clusters include distinct 1,928 and 2,242 sequences, respectively. The number of sequences shared in matching pair is 1,644. It means the CGP method classified 77% of 2,174 sequences in InterPro families into appropriate clusters. Table 2 shows the detail of matching pairs with matching rate 0.5 or more. They are selected such that members of a cluster could belong to its match family with high probabilities. 186 clusters (87%) of matching pairs in both directions have matching rate 0.5 or more. In Table 2, false positive means a cluster sequence which is not included by its match family and is included by the others. Sequences not included by any families are referred to NF, Not in any Families. NF sequences could be candidate members of

Table 1: Relations of matching pairs in both directions. F and C mean an InterPro family and a CGP cluster of matching pair, respectively. *This number includes some redundant sequences because the InterPro families overlap each other. The total number of distinct proteins in F is 1,928.

	No. of F(or C)	No. of proteins shared by F and C	No. of proteins in F	No. of proteins in C
$F = C$	110 (52%)	680 (41%)	680 (34%)	680 (30%)
$F \subset C$	34 (16%)	287 (17%)	287 (14%)	716 (32%)
$F \supset C$	43 (20%)	434 (26%)	615 (31%)	434 (19%)
$F \neq C$	26 (12%)	243 (15%)	414 (21%)	417 (19%)
Total	213	1644	1996*	2247

Table 2: Relations of matching pairs in both directions with matching rate 0.5 or more. F and C are the same meaning to Table 1. FP means False Positives, sequences included by $C \cap \overline{F}$ and any InterPro families except for F. FN means False Negatives, sequences included by $F \cap \overline{C}$. NF means Not in any Families, sequences not included by any families. †This number includes some redundant sequences because the InterPro families overlap each other. The total number of distinct proteins in F and FN are 1,759 and 206 respectively.

	No. of F(or C)	No. of proteins in both F and C	No. of proteins in F	No. of proteins in C	No. of FP	No. of FN	No. of NF
$F = C$	110 (59%)	680 (44%)	680 (38%)	680 (41%)	0	0	0
$F \subset C$	27 (15%)	256 (16%)	256 (14%)	332 (14%)	25	0	51
$F \supset C$	37 (20%)	402 (26%)	518 (29%)	402 (24%)	0	116	0
$F \neq C$	12 (6%)	215 (14%)	328 (18%)	252 (15%)	9	113	28
Total	186	1553	1782†	1666	34	229†	79

the match family.

Our implementation was executed on Compaq AlphaStation XP1000 (CPU: Alpha 21264 500MHz). 277 connected components were produced by dividing a whole data with threshold 33 as pre-processing, which takes a few seconds. The largest component have 2,839 vertices and 62,704 edges in its weighted linkage graph. Our clustering algorithm for the large set components takes 1,039 seconds and 7.9Mbytes memory. Computation for the second or the other components take very few time and memory. Table 3 shows the number of one-pass improvement iteration (outside repeat of our graph partitioning algorithm) and the number of graph partitioning for the largest connected components and the sum of all components. It reveals that the number of one-pass improvement iteration is less than five in most cases. This implies that the iteration of the one-pass improvement does not increase the time complexity too much.

4 Conclusion

A graph-based clustering method was proposed to cluster protein sequences into families using a weighted linkage graph based on sequence similarity. It improves the single linkage clusters automatically using a graph partitioning algorithm, which uses a heuristic of the FM algorithm. The effectiveness of our method was shown by comparison with InterPro families in all mouse proteins in SWISS-PROT. The time complexity of our method takes $O(|V|^3)$ time, granted that the number of

Table 3: Relations between the numbers of one-pass improvement iteration (outside repeat of the graph partitioning algorithm) and the graph partitioning.

No. of one-pass improvement	No. of partitioning (for the largest connected components)	No. of partitioning (for all connected components)
1	82	85
2	1,767	1,958
3	171	179
4	16	2
6	1	2
13	1	1

one-pass improvement iteration is constant. The result was computed in about twenty minutes for the mouse protein sets. If the time complexity is improved and a guideline for a threshold used in construction of a weighted linkage graph is given, it enables us to manage a huge set of proteins easily such as all proteins of more species.

5 Acknowledgement

This work was supported by Research and Development for Applying Advanced Computational Science and Technology (ACT-JST).

References

[1] Kaufman, L. and Rousseeuw, P.J., *Finding Groups in Data: an Introduction to Cluster Analysis*, Wiley, 1990.

[2] Hartigan, J.A., *Clustering Algorithms*, Wiley, New York, 1975.

[3] Altschul, S.F., Madden, T.L., Schaffer, A.A., Zhang, J., Zhang, Z., Miller, W., and Lipman, D.J., Gapped BLAST and PSI-BLAST: a new generation of protein database search programs, *Nucleic Acids Res.*, 25:3389–3402, 1997.

[4] Pearson, W.R. and Lipman, D.J., Improved tools for biological sequence comparison, *Proc. Natl Acad. Sci. USA*, 85:2444–2448, 1998.

[5] Krause, A. and Vingron, M., A set-theoretic approach to database searching and clustering *Bioinformatics*, 14:430–438, 1998.

[6] Enright, A.J. and Ouzounis, C.A., GeneRAGE: a robust algorithm for sequence clustering and domain detection, *Bioinformatics*, 16:451–457, 2000.

[7] Matsuda, H., Ishihara, T., and Hashimoto, A., Classifying molecular sequences using a linkage graph with their pairwise similarities, *Theor. Comput. Sci.*, 210:305–325, 1999.

[8] Apweiler, R., Attwood, T.K., Bairoch, A., Bateman, A., Birney, E., Biswas, M., Bucher, P., Cerutti, L., Corpet, F., Croning, M.D.R., Durbin, R., Falquet, L., Fleischmann, W., Gouzy, J., Hermjakob, H., Hulo, N., Jonassen, I., Kahn, D., Kanapin, A., Karavidopoulou, Y., Lopez, R., Marx, B., Mulder, N.J., Oinn, T.M., Pagni, M., Servant, F., Sigrist, C.J.A., and Zdobnov, E.M., The InterPro database, an integrated documentation resource for protein families, domains and functional sites, *Nucleic Acids Res.*, 29:37–40, 2001.

[9] Bairoch, A. and Apweiler, R., The SWISS-PROT protein sequence database and its supplement TrEMBL in 2000, *Nucleic Acids Res.*, 28:45–48, 2000.

[10] Bateman, A., Birney, E., Durbin, R., Eddy, S.R., Howe, K.L., and Sonnhammer, E.L., The Pfam Protein Families Database, *Nucleic Acids Res.*, 28:263–266, 2000.

[11] Attwood, T.K., Croning M.D.R., Flower, D.R., Lewis, A.P., Mabey, J.E., Scordis, P., Selley, J., and Wright, W., PRINTS-S: the database formerly known as PRINTS, *Nucleic Acids Res.*, 28:225–227, 2000.

[12] Hofmann, K., Bucher, P., Falquet, L., and Bairoch, A., The PROSITE database, its status in 1999, *Nucleic Acids Res.* 27:215–219, 1999.

[13] Corpet, F., Servant, F., Gouzy, J., and Kahn, D., ProDom and ProDom-CG: Tools for protein domain analysis and whole genome comparisons, *Nucleic Acids Res.*, 28:267–269, 2000.

[14] Schultz, J., Copley, R.R., Doerks, T., Ponting, C.P., and Bork, P., SMART: a web-based tool for the study of genetically mobile domains, *Nucleic Acids Res.*, 28:231–234, 2000.

[15] Koonin, E.V., Tatusov, R.L., and Rudd, K.E., Sequence similarity analysis of *Escherichia coli* proteins – functional and evolutionary implications, *Proc. Natl Acad. Sci. USA*, 92:11921–11925, 1995.

[16] Watanabe, H. and Otsuka, J., A comprehensive representation of extensive similarity linkage between large numbers of proteins, *Comput. Appl. Biosci.*, 11:159–166, 1995.

[17] Sharan, R., and Shamir, R., CLICK: A clustering algorithm with applications to gene expression analysis, *Proc. 8th Annual Conf. Intelligent Systems for Molecular Biology*, 307–316, 2000.

[18] Garey, M.R. and Johnson, D.S., *Computers and Intractability*, W. H. Freeman and Company, 1979.

[19] Kernighan, B. and Lin, S., An effective heuristic procedure for partitioning graphs, *Bell Syst. Techn Journ.*, 2:291–307, 1970.

[20] Fiduccia, C.M. and Mattheyses, R.M., A Linear-time heuristic for improving network partitions, *Proc. 19th Design Automation Conf.*, 175–181, 1982.