2007－BIO－9 (3)
2007／6／14

# Self-organizing clustering: non-hierarchical clustering for large scale DNA sequence data

天野晃[†,‡,§], 市川裕章[†], 中村英光[†], 沼寿隆[†], 深海薫[‡,§], 長村吉晃[†], 小野寺夏生[§]
† 農業生物資源研究所, ‡ 理化学研究所バイオリソースセンター, § 筑波大学

近年、クラスタリングは長大な DNA シークエンスを分類あるいは解析するための、基本的かつ重要な方法と認識されている。われわれは、多数のかつ非常に大きな DNA シークエンスをオリゴヌクレオチド頻度にもとづき分類する、Self-Organizing Clustering (SOC) と呼ぶ方法およびプログラムを開発した。プログラムはコマンドラインプログラムパッケージとして開発され、インターネットを通してダウンロードおよび利用が可能である。プログラムのダウンロードは、http://rgp.nias.affrc.go.jp/programs/より、利用は、http://rgp.nias.affrc.go.jp/SOC/より行なえる。

# Self-organizing clustering: non-hierarchical clustering for large scale DNA sequence data

Kou AMANO[†,‡,§], Hiroaki ICHIKAWA[†], Hidemitsu NAKAMURA[†], Hisataka NUMA[†],
Kaoru FUKAMI-KOBAYASHI[‡,§], Yoshiaki NAGAMURA[†] and Natsuo ONODERA[§]
† National Institute of Agrobiological Sciences, ‡ RIKEN BioResource Center, § University of Tsukuba

Recently, clustering has been recognized as an important and fundamental method that analyzes and classifies large-scale sequence data to provide useful information. We developed a novel clustering method designated as Self-organizing clustering (SOC) that uses oligonucleotide frequencies for large-scale DNA sequence data. We implemented SOC as a command-line program package, and developed a server that provides access to it enabling visualization of the results. SOC effectively and quickly classifies many sequences that have low or no homology to each other. The command-line program is downloadable at http://rgp.nias.affrc.go.jp/programs/. The on-line web site is publicly accessible at http://rgp.nias.affrc.go.jp/SOC/. The common gateway interface (CGI) scripts for the server is also provided within the package.

## 1 Introduction

Self-organizing clustering (SOC) rapidly produces non-hierarchical clusters from a large amount of DNA sequence data by using oligonucleotide frequencies. By evading time-consuming and complicated sequence alignments, it works effectively to classify sequences that have low or no homology to each other and that cannot be classified with existing clustering tools. When sequences are obtained from promoter regions, non-coding regions, or are randomly sampled from several species, they rarely have homology with each other. SOC can classify even such sequences.

A prototype of SOC was originally developed as a command-line program (SOC commands) and released in 2003 [1, 2]. We then made the commands accessible through a web browser designated as the SOC server.

## 2 Program overview

The clustering algorithm of SOC is based on the k-means and learns to move cluster nodes, which represent clusters, expressing the centroids of clusters. The time complexity of the calculation is $O(cnml)$, where $c$, $n$, $m$, and $l$ refer to the number of clusters specified by the user, number of sequences, size of the vector that is comprised of oligonucleotide frequencies, and number of loop iterations, respectively.

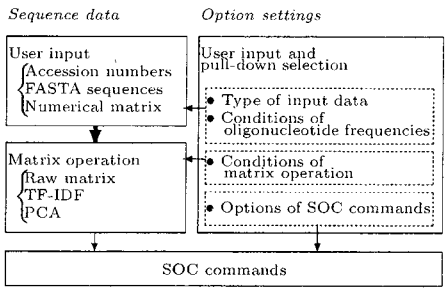


Figure 1: Overview of the SOC server.

The work flow of user input and settings. A thick arrow represents data transformation. Thin arrows represent data transfer. Left brackets "{" indicate selection of items. Items with dots surrounded by dash-boxes "• text" indicate conditions necessary for the operation.

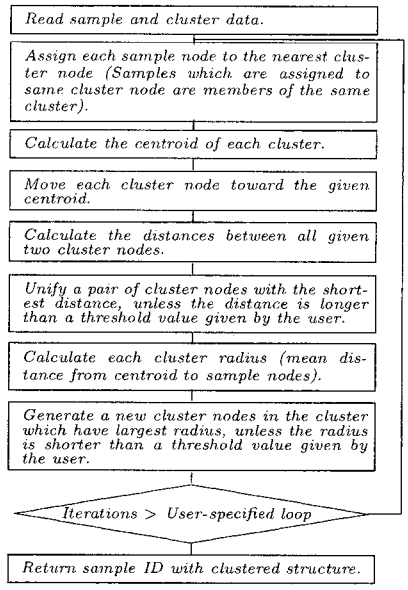


Figure 2: Flow chart of learning process.

Unifying and generating operations are executed with times set in user-specified loop. Therefore, this operation has little influence in complexity of the calculation.

# 3 Algorithm and implementation

The SOC server consists of the following four functions: (1) Preprocessing: generating a numerical matrix whose elements correspond to the oligonucleotide frequencies of the sequences. (2) Initializing: placing cluster nodes at random or according to user specification. (3) Cluster node alignment: learning how to move each cluster node so that it expresses the centroid of the cluster. (4) Visualization: displaying the results graphically. Functions (1) and (4) are realized by a common gateway interface (CGI), while (2) and (3) are executed by the SOC commands. The CGI scripts are written in Perl language with a GD module, and these are installed only on a Linux PC at present. The SOC commands are written in C language, and their compilation and execution have been verified on many operating systems, including HP-UX, Linux, Mac OS X, SGI-IRIX, and SUN OS. **Figure 1** shows the work flow corresponding to function (1) where the inputs by the user are handled and transferred to the SOC commands through the web browser. In initializing process corresponding to function (2), the user can select a cluster generating mode from four methods: "Diagonal", "random=Value", "node=Central", and "Grid". In these modes, `soc-init` command generates cluster nodes on diagonal line, at random, on the samples near to the centroid of the sample coordinates, or on lattice points of the sample space, respectively. **Figure 2** shows the algorithm of the learning process corresponding to cluster node alignment (function 3) executed by `soc-lm` command. **Figure 3** shows snapshots of the browser connected to the SOC server. In the data input page (**Figure 3a**), the user can select the query type from among "FASTA sequence", "Accession number", and "Numerical matrix". When "FASTA sequence" is selected, the SOC server generates a numerical matrix of oligonucleotide frequencies from the input sequences. When "Accession number" is selected, the server accesses the DNA sequence database to obtain the relevant sequences and generates a numerical matrix. On the other hand, when "Numerical matrix" is selected, the user can directly input a numerical matrix. In any case, the

Table 1: Use of SOC to cluster 629 coding regions from three domains based on the pentanucleotide frequency.

| | Cluster | | | |
|---|---|---|---|---|
| Domain | 1 | 2 | 3 | 4 |
| Eukarya | 542 | 1 | 2 | 1 |
| Bacteria | 1 | 62 | 0 | 0 |
| Archaea | 0 | 0 | 20 | 0 |

Sizes of coding regions are between 2.0 and 2.2 kbp. Initial value of clusters was 25. Each value shows the number of cluster members.

user can use normalized vectors when the box "Use per kbp value" was checked. The analysis is executed by cron daemon at specified intervals. When the job is completed, the system notifies the user by an e-mail containing the URL where the result is located.

One of the advantages of the SOC server is that it provides the user with some clues to avoid problems of local solutions caused by initial value dependency, which often occur in learning type clustering methods. The following two functions achieve the advantage, when their respective options are specified.

One of the functions generates cluster nodes with equal intervals in a principal component space. In the initializing process, principal component analysis (PCA) is made based on the oligonucleotide frequencies. The SOC server divides the principal component space into equal intervals and generates cluster nodes on the lattice points. As a consequence, it is likely that the SOC server is able to bypass the initial value dependency and local solutions.

Another function is for dynamic generation and unification of cluster nodes. In the learning process, it is often observed that some members of a cluster are located at a distance from the cluster center, while different clusters are very closely located. In these cases, the expected results might not be obtained. To avoid the cases, the SOC server generates new clusters to locate outlying specimens and unify closely related clusters into one. Using these procedures, the SOC server maintains the validity of the cluster radius.

# 4 Application

## 4.1 Genome-wide clustering of *Caenorhabditis elegans*

**Figure 4** shows genome-wide SOC clustering and mapping of *Caenorhabditis elegans* based on the tetranucleotide frequency. We segmented each chromosome sequence of *C. elegans* into 1,000 bp pieces and obtained 100,092 frames without overlaps. The remainder was assigned to both ends of each chromosome. And the frames containing over 5% character of 'N's were discarded. Thus, we used 94,460 frames to classify them into five to ten clusters. **Figure 4a** and **b** show the concept behind the mapping and the obtained results, respectively. In **Figure 4b**, the localization pattern of cluster 1 members looks like a broken line at the centromeric portion of each chromosome. This pattern correlated with that of the TTAGGC telomeric repeat (CeRep26)[3, 4]. The result indicated that SOC can detect the localization pattern of a sequence repeat, even if the presence of the repeat is not recognized *a priori*. The other clusters (IDs 2–6) made no localization. In this mapping method, only one or a few of the clusters could usually detect characteristic distributions.

## 4.2 Clustering of coding regions from three domains

We selected 629 coding regions extracted from the following five species that belong to three domains: *Aeropyrum pernix* (Archaea), *Rhizobium radiobacter* (*Agrobacterium tumefaciens*) strain C58 (Bacteria), *Arabidopsis thaliana* (Eukarya, Plantae), *C. elegans* (Eukarya, Animalia), and *Saccharomyces cerevisiae* (Eukarya, Fungi). To avoid the influence of sequence length, only coding regions of 2.0–2.2 kbp in length were used for SOC clustering. Such coding regions were obtained from the whole genome sequences of *A. pernix* (20 coding regions), *R. radiobacter* (63), and *S. cerevisiae* (231), and from chromosome I sequences of *A. thaliana* (228) and *C. elegans* (87). We classified them on the conditions that the lengths of oligonucleotide are 2–5 bp, and the initial number of clusters are 5 to 50 (with steps of 5). The best result was obtained, when the pentanucleotide frequencies were used and the initial number of clusters was specified as being between 20 and 30 (**Table 1**). Interest-

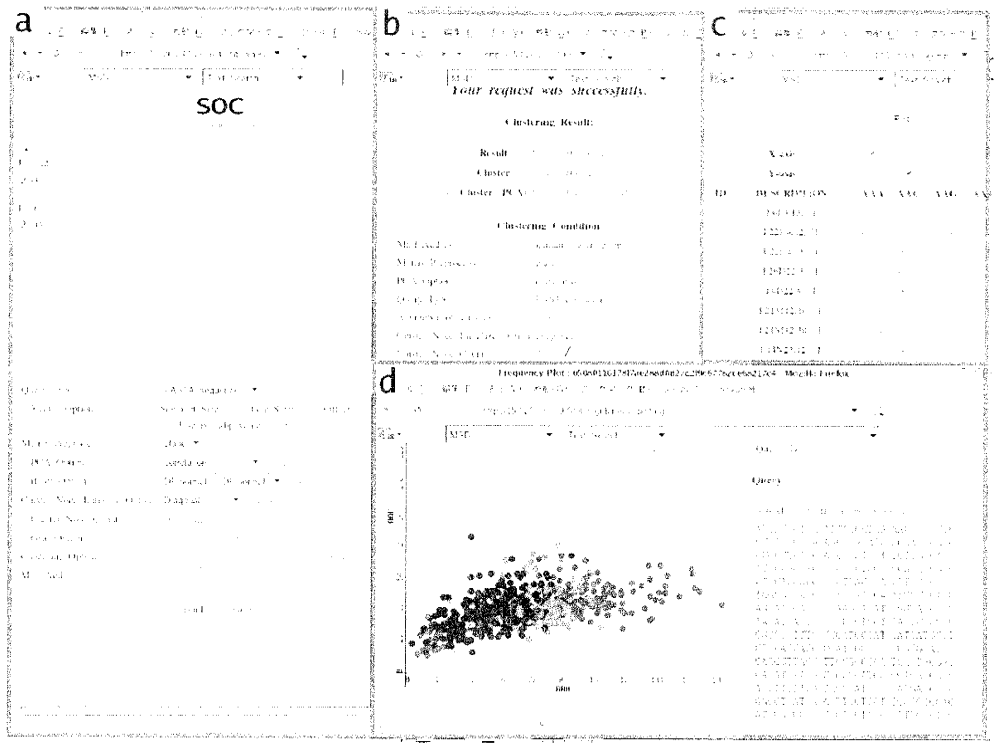


Figure 3: Snapshots of the SOC server.

(a) Data input page. The user can input sequence data, matrix options, and cluster node options. (b) Summary page. The options specified by the user in "data input page" are summarized. (c) Plot setting page. The user can select the parameters for X and Y axes. (d) Plot page. Samples belonging to a cluster are plotted in the same color. The user can obtain sequence information (sequence, sequence ID, and cluster ID) when the dots are clicked.

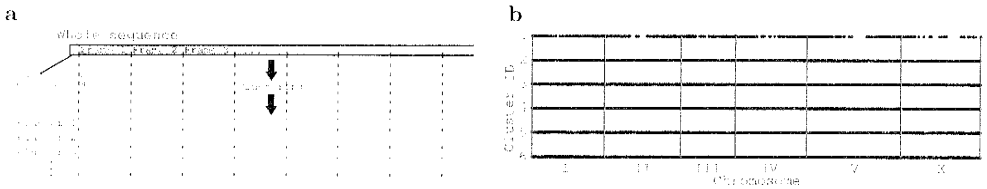


Figure 4: Clustering and mapping of all six chromosomes of *Caenorhabditis elegans*.

(a) Concept of mapping. Frames that are the same in size and that have no overlaps are prepared from each chromosome. All the frames (members) of the clusters are mapped to their native positions on the genome. (b) Result of mapping. The size of frames was 1,000 bp. The initial number of clusters was specified six. SOC found the regions corresponding to the locations of TTAGGC repeats (Cluster 1).

ingly, most of the coding regions were classified into their specific clusters. The cluster 1 was characterized by AT-rich sequences and most Eukarya coding regions were classified into the cluster. The clusters 2 and 3 were characterized by poly-G/poly-C sequences and by (GC)n tandem repeats, respectively. SOC revealed the sequence characteristics of individual species for all the sequences except five coding regions from Eukarya and Bacteria. A unique coding region classified in cluster 4 was derived from *A. thaliana* and bore a repeated motif of 72 bp, which is a very peculiar feature for a protein coding region. The sequence contains 18 leucine-rich repeats (LRR), and no other sequence with high homology to the LRR sequence was found in the DDBJ/EMBL/GenBank database. In SOC, such a feature was not considered to belong to any other cluster. The other four coding regions also had atypical sequence feature of each domain and were positioned on the outskirts of the cluster of the other members. Thus, these results demonstrated that SOC can recognize, distinguish and extract the features of individual sequences with high accuracy.

# 5 Concluding remarks

In this paper, We showed two examples of clustering for a number of nucleotide sequences using SOC. The results indicate that SOC is suitable for clustering a large number of diverse sequences and extracting specific sequence characteristics among them.

# Acknowledgement

We thank Prof. M. Iwasawa (School of Library and Information Science, University of Tsukuba), and Dr. T. Yamaguchi (Cybernet Systems Co., Ltd.) for helpful comments on SOC. CGI development for the server was assisted by Mitsubishi Space Software Co., Ltd. This work was partly supported by a grant from the Ministry of Agriculture, Forestry and Fisheries of Japan (Green Technology Project: EF1001 and EF1004).

# References

[1] Amano,K.: Clustering DNA sequences using self-organizing method. *Abstracts of the 5th Meeting of Japan Society for Information and Media Studies*, pp. 5-8 (2003).

[2] Amano,K., Nakamura,H. and Ichikawa,H.: Self-organizing clustering: A novel non-hierarchical method for clustering large amount of DNA sequences. *Genome Informatics*, Vol. 14, pp. 575-576 (2003).

[3] The *C. elegans* Sequencing Consortium: Genome sequence of the nematode *C. elegans*: A platform for investigating biology. *Science*, Vol. 282, pp. 2012-2018 (1998).

[4] Wicky,C., Villeneuve,A. M., Lauper,N., Codourey,L., Tobler,H. and Müller,F.: Telomeric repeats (TTAGGC)n are sufficient for chromosome capping function in *Caenorhabditis elegans*. *Proc. Natl. Acad. Sci. USA*, Vol. 93, pp. 8983-8988 (1996).