Hayashida and Akutsu *BMC Systems Biology* 2010, **4**(Suppl 2):S13
http://www.biomedcentral.com/1752-0509/4?issue=S2/S13

BMC Systems Biology

PROCEEDINGS Open Access

Comparing biological networks via graph compression

Morihiro Hayashida*, Tatsuya Akutsu*

From Optimization and Systems Biology
Zhangjiajie, China. 20 – 22 September 2009

Abstract

Background: Comparison of various kinds of biological data is one of the main problems in bioinformatics and systems biology. Data compression methods have been applied to comparison of large sequence data and protein structure data. Since it is still difficult to compare global structures of large biological networks, it is reasonable to try to apply data compression methods to comparison of biological networks. In existing compression methods, the uniqueness of compression results is not guaranteed because there is some ambiguity in selection of overlapping edges.

Results: This paper proposes novel efficient methods, CompressEdge and CompressVertices, for comparing large biological networks. In the proposed methods, an original network structure is compressed by iteratively contracting identical edges and sets of connected edges. Then, the similarity of two networks is measured by a compression ratio of the concatenated networks. The proposed methods are applied to comparison of metabolic networks of several organisms, *H. sapiens, M. musculus, A. thaliana, D. melanogaster, C. elegans, E. coli, S. cerevisiae,* and *B. subtilis,* and are compared with an existing method. These results suggest that our methods can efficiently measure the similarities between metabolic networks.

Conclusions: Our proposed algorithms, which compress node-labeled networks, are useful for measuring the similarity of large biological networks.

Background

Development of algorithms for comparing various kinds of biological data is one of the important topics in bioinformatics and systems biology. Methods for comparison of DNA and/or protein sequences have been extensively studied and have been applied to analyses of real sequence data quite successfully. Due to increased interests in systems biology, extensive studies have recently been done on comparison of biological networks.

For comparison of metabolic networks, Ogata et al. developed a method based on clustering [1], Tohsato et al. extended a multiple sequence alignment technique to multiple alignment of metabolic pathways using a scoring scheme based on EC (Enzyme Commission) numbers [2], Pinter et al. applied a tree matching technique to alignment of metabolic pathways [3], and Wernicke and Rasche developed a simple backtracking algorithm utilizing the local diversity property [4]. For comparison of protein-protein interaction networks, Kelley et al. developed PathBlast using dynamic programming [5], Liang et al. developed NetAlign using a clique-based method for computing maximal common subgraphs [6], Li et al. developed MNAligner using integer quadratic programming [7], Singh et al. developed IsoRank algorithm based on Google's PageRank method [8], and Zaslavskiy et al. developed a gradient ascent-based method and a message passing-based method [9].

On the other hand, data compression methods have been applied to comparison of large sequence data [10,11] and protein structure data [12,13]. Since it is still difficult to compare global structures of large biological

* Correspondence: morihiro@kuicr.kyoto-u.ac.jp; takutsu@kuicr.kyoto-u.ac.jp
Bioinformatics Center, Institute for Chemical Research, Kyoto University, Gokasho, Uji, Kyoto, 611-0011, Japan
Full list of author information is available at the end of the article

networks and data compression-based methods can be applied to comparison of large-scale sequence data, it is reasonable to try to apply data compression methods to comparison of biological networks. In this paper, we propose such methods.

In order to apply data compression to biological networks, data compression methods for graphs are required. For compression of graphs, Adler and Mitzenmacher developed a method based of Huffman coding of vertices [14], Peshkin developed GRAPHITOUR based on iterative contractions of identical edges [15], and Cook and Holder developed SUBDUE based on contraction of frequent subgraphs and MDL (minimum description length) principle [16], which was further extended to EDIF for lossless compression by Yang et al. [17]. However, the method by Adler and Mitzenmacher does not seem to be useful for comparison of networks because it does not make much use of structural information. In GRAPHITOUR, the uniqueness of compression results is not guaranteed because there is some ambiguity in selection of overlapping edges (isomorphic graphs may be differently compressed depending on the orderings of vertices in input data), which is not suitable for comparison of network structures. This point is also unclear in EDIF and SUBDUE. Therefore, we develop in this paper novel graph compression methods for which it is guaranteed that two isomorphic graphs are compressed in the same way. Using these compression methods, we measure the similarity of two networks by means of the universal similarity metric (USM) proposed by Li *et al.*[11]. USM is defined using Kolmogorov complexity which represents the amount of information contained in data, and is obtained by removing redundant parts maximally. Therefore, Kolmogorov complexities are approximated by compression sizes.

We apply the proposed methods to comparison of metabolic networks, and compare the results with those of GRAPHITOUR. The results of hierarchical clustering for several organisms suggest that the proposed methods outperforms GRAPHITOUR, and can adequately measure the similarities between metabolic networks.

Methods

Graph compression method

Since our proposed methods are based on GRAPHITOUR, we briefly review the GRAPHITOUR algorithm [15]. GRAPHITOUR is based on iterative contractions of identical edges. In order to efficiently contract edges, GRAPHITOUR selects edges appearing most frequently, and solves an instance of *maximum cardinality matching* problem, which finds as many edges as possible such that no two edges share a common vertex.

Figure 1 shows an example of contraction of identical edges. The graph of (A) contains 4 edges labeled with ‘a’ and ‘b’, 2 edges with ‘a’ and ‘a’, 1 edge with ‘b’ and ‘b’, and 1 edge with ‘a’ and ‘c’. GRAPHITOUR selects edges with ‘a’ and ‘b’ because they appear most frequently, and solves the maximum cardinality matching problem for their edges. However, optimal solutions are not necessarily uniquely determined. (B) shows a contracted graph after the top-left edge with ‘a’ and ‘b’ is substituted with a new vertex labeled with ‘ab’. On the other hand, (C) shows a contracted graph after the top-right edge with ‘a’ and ‘b’ is substituted. This example implies that GRAPHITOUR can generate different compressed graphs. In order to measure the similarity of networks, the same compressed graph should always be obtained. Therefore, we improve GRAPHITOUR for that purpose, and propose the following algorithm, which we call CompressEdge, to uniquely determine contracted edges in each iteration.

```
Procedure CompressEdge
Input: undirected graph $G(V, E)$ with labeled vertices $V$ and edges $E$ (a total order $\le$ is defined for the set of labels $L$, and each $v \in V$ is labeled with $l_v \in L$);
Output: induced compression rules $R$ and compressed graph;
Begin

  $R := \emptyset$;
  $s(l) := \{l\}$ for each label $l \in L$;
  while $|E| > 0$
    $\varepsilon(l_1, l_2) := \{(v_1, v_2) \in E \mid (l_{v_1}, l_{v_2}) = (l_1, l_2)$ where $l_{v_1} \ge l_{v_2}$, $l_1 \ge l_2\}$;
    $\varepsilon := \{\varepsilon(l_1, l_2) \mid$ no two edges in $\varepsilon(l_1, l_2)$ share a common vertex$\}$;
    if $\varepsilon = \emptyset$ then return $(R,G)$;
    $\varepsilon_{most} := \{\varepsilon(l_1, l_2) \in \varepsilon \mid |\varepsilon(l_1, l_2)| \ge |\varepsilon(l_3, l_4)|$ for all $\varepsilon(l_3, l_4) \in \varepsilon\}$;
    select $\varepsilon(l_1, l_2)(\in \varepsilon_{most})$ such that $s(l_1) \cup s(l_2) < s(l_3) \cup s(l_4)$
        or $(s(l_1) \cup s(l_2) = s(l_3) \cup s(l_4)$ and $(l_1, l_2) < (l_3, l_4))$
        for all $\varepsilon(l_3, l_4) \in \varepsilon_{most}$,
        where $l_1 \ge l_2$, $l_3 \ge l_4$;
    add a new label $l_n$ to $L$ such that $l_n > l$ for all $l \in L$;
    $s(l_n) := s(l_1) \cup s(l_2)$;
    $R = R \cup \{l_n \leftarrow (l_1, l_2)\}$;
    for each edge $e \in \varepsilon(l_1, l_2)$
        substitute $e$ with a new vertex labeled with $l_n$;
  return $(R, G)$;

End
```

This proposed algorithm avoids contraction of edges which share a common vertex. In the example of Figure 1, our algorithm does not choose edges whose endpoints are ‘a’ and ‘b’, instead chooses the second

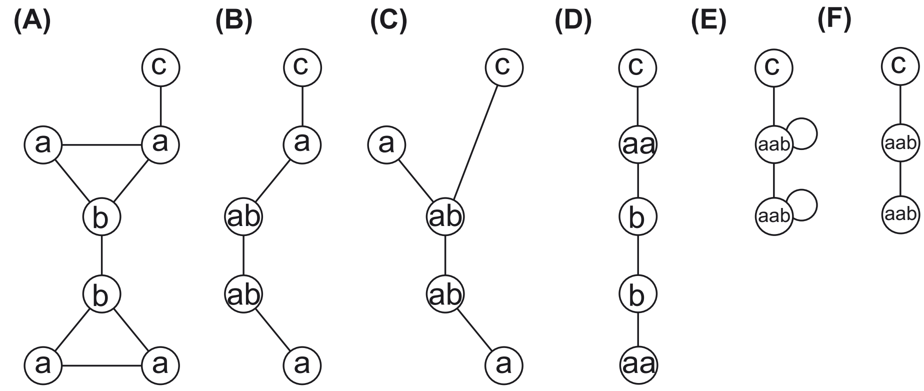

Figure 1 Example of contraction of identical edges. (A) Graph with 7 vertices and 8 edges. (B, C) Graphs contracted from graph (A) by GRAPHITOUR. (D) Graph contracted from graph (A) by our proposed method. (E) Graph contracted from graph (A) by substituting two edges ('a', 'a') and ('a', 'b') with 'aab'. (F) Graph contracted from graph (A) by substituting three vertices and the edges between the vertices with 'aab'.

candidate edges whose endpoints are 'a' and 'a', and obtains the graph of (D) as the result. It should be noted that the proposed algorithm does not solve the maximum cardinality matching problem because it selects only edges such that all edges with the same labels do not share a common vertex.

However, it is not sufficient to uniquely determine contracted edges because there can be more than one set which has the same number of edges, that is, $|\varepsilon_{most}| > 1$. Therefore, we introduce a total order to sets of labels to determine the priority of edges. Each edge has a pair of labels (l_1, l_2) corresponding to two endpoints of the edge. Let s_1 and s_2 be sets of labels. We can define a total order for s_1 and s_2 as follows.

First, we sort s_1 and s_2 by descending order, respectively. We compare i-th elements $s_1^{(i)}$, $s_2^{(i)}$ of s_1 and s_2, and define $s_1 < s_2$ if $s_1^{(i)} < s_2^{(i)}$ and $s_1^{(j)} = s_2^{(j)}$ (for all $j < i$) hold for some i. The proposed algorithm selects edges with the smallest set of labels from ε_{most} according to the total order. For example, if we compare $s_1 = \{l_1, l_3\}$ with $s_2 = \{l_3, l_2\}$ under $l_1 < l_2 < l_3$, s_1 and s_2 are sorted as (l_3, l_1) and (l_3, l_2), respectively, and we have $s_1 < s_2$.

When edges with (l_1, l_2) are contracted, a new label l_n is added to L, where $l_n > l$ for all l $(\neq l_n) \in L$. In computational experiments, Morgan index [18] based on graph structures is assigned to each vertex. However, new added labels themselves do not reflect the original graph structure. Therefore, in order to make effective use of the total order of original labels, we introduce a set of labels for each label l, $s(l)$, which consists of only original labels. Then, $s(l_n)$ is defined to be $s(l_1) \cup s(l_2)$ when (l_1, l_2) is substituted with l_n. The algorithm compares $s(l_1) \cup s(l_2)$ with $s(l_3) \cup s(l_4)$ before comparing edges of (l_1, l_2) and (l_3, l_4). For example, for the graph of Figure 1D, the algorithm selects edges with ('aa', 'b') as contracted edges because it appears most frequently. However, if there is another edge with ('b', 'b') than shown in Figure 1D, edges of ('aa', 'b') and ('b', 'b') are compared. We suppose that 'a'<'b'<'c'<'aa' and 'aa' was obtained by contracting edges with ('a', 'a'). Then, the corresponding sets to ('aa', 'b') and ('b', 'b'), $s_1 = s(\text{'aa'}) \cup s(\text{'b'}) = \{\text{'a','a','b'}\}$ and $s_2 = s(\text{'b'}) \cup s(\text{'b'}) = \{\text{'b','b'}\}$, are compared, sorted as {'b','a','a'} and {'b','b'}, respectively, and we have $s_1 < s_2$. Then, edges with ('aa', 'b') are selected, and contracted to vertices with a new label 'aab', where 'aab'>'aa' and $s(\text{'aab'}) = s(\text{'aa'}) \cup s(\text{'b'}) = \{\text{'a','a','b'}\}$.

Extension to contraction of multiple edges

In the previous algorithm, identical edges are contracted at each iteration. In this section, we propose another algorithm, which we call CompressVertices, to contract identical sets of multiple connected edges. In order to uniquely determine the contracted sets of connected edges, we must introduce a total order to a set of edges. For that purpose, we apply degree sequence [19], which is defined to be the non-increasing sequence of degrees of vertices. For example, the degree sequence of Figure 1A is (3, 3, 3, 2, 2, 2, 1). Moreover, in order to distinguish labels of vertices, we introduce the non-increasing sequence of pairs of the degree and the label for each vertex included in the set of edges, which we call dl-sequence. In dl-sequence, the degree is not calculated for the original graph, but is for the set of edges, and we define the inequality of elements of dl-sequence by; $(d_1, l_1) > (d_2, l_2)$ if $d_1 > d_2$ or ($d_1 = d_2$ and $l_1 > l_2$). Then, we can define a total order for dl-sequences $dl_1 (= ((d_1^{(1)}, l_1^{(1)}), \ldots, (d_k^{(1)}, l_k^{(1)})))$ and $dl_2 (= ((d_1^{(2)}, l_1^{(2)}), \ldots, (d_k^{(2)}, l_k^{(2)})))$ by $dl_1 < dl_2$ *if* $(d_i^{(1)}, l_i^{(1)}) < (d_i^{(2)}, l_i^{(2)})$ and $(d_j^{(1)} = d_j^{(2)}$ *and* $l_j^{(1)} = l_j^{(2)}$ *for all* $j < i)$ hold for some i. For example, the dl-sequence of two connected edges of ('a', 'a') and ('a', 'b') in Figure 1A, that is a–a–b, is ((2, ‘a’), (1, ‘b’), (1, ‘a’)). Here, in the example, if the edges are substituted with a new vertex, a self loop is remained to the new vertex due to the edge of (’ a’, ‘b’) as shown in Figure 1E. The remained edges should be also contracted. Therefore, we modify CompressEdge, and propose the following algorithm to contract identical sets of vertices and the edges between the vertices instead of individual edges. In the example of Figure 1E, the graph of Figure 1F is obtained by this algorithm.

Procedure CompressVertices(*M*)

Input: maximum number of vertices M and undirected graph $G(V, E)$ with labeled vertices V and edges E

Output: induced compression rules R and compressed graph;

```
Begin
  $R := \emptyset$;
  $s(l) := \{l\}$ for each label $l \in L$;
  while $|E| > 0$
    $V = \emptyset$; $m := 2$;
    while $V = \emptyset$ and $m \le M$
      $(V, dl_i) :=$ SelectVertices($m$);
      $m := m + 1$;
    if $V = \emptyset$ then return $(R, G)$;
    add a new label $l_n$ to $L$ such that $l_n > l$ for all $l \in L$;
    $s(l_n) := \bigcup_{(d_{i'}, l_{i'}) \in dl_i} s(l_{i'})$;
    $R = R \cup \{l_n \leftarrow dl_i\}$;
    for each set of vertices $u \in V$
      substitute $u$ with a new vertex labeled with $l_n$;
  return $(R, G)$;
End
```

```
Subprocedure SelectVertices($m$)
Begin
  $V(dl_i := ((d_1, l_1), \ldots, (d_m, l_m))) := \{\{v_1, \ldots, v_m\} \subset V \mid v_1, \ldots, v_m$ are connected and (the dl-sequence of $\{v_1, \ldots, v_m\}) = dl_i\}$;
  $V := \{V(dl_i) \mid u_1 \cap u_2 = \emptyset$ for all $u_1, u_2 \in V(dl_i)\}$;
  if $V = \emptyset$ then return $(\emptyset, \emptyset)$;
  $V_{most} := \{V(dl_i) \in V \mid |V(dl_i)| \ge |V(dl_j)|$ for all $V(dl_j) \in V\}$;
  select $V(dl_i) (\in V_{most})$ such that $s_i (:= \bigcup_{(d_{i'}, l_{i'}) \in dl_i} s(l_{i'})) < s_j (:= \bigcup_{(d_{j'}, l_{j'}) \in dl_j} s(l_{j'}))$
      or ($s_i = s_j$ and $dl_i < dl_j$) for all $V(dl_j) \in V_{most}$;
  return $(V(dl_i), dl_i)$;
End
```

First, the algorithm tries to find two vertices and the edge between the vertices to be contracted. If it does not find such two vertices, it tries to find more than two vertices and the edges between the vertices until it finds or up to the given number of vertices, M. Figure 2 shows an example of contraction by the algorithm. The graph of (A) contains 4 edges labeled with ‘a’ and ‘b’, 2 edges with ‘a’ and ‘c’, and 2 edges with ‘b’ and ‘c’. However, it cannot select any edge because they are overlapping each other. Therefore, it tries to find three vertices to be contracted. (B) shows the candidate sets of vertices {’ a’, ‘b’, ‘c ‘} and three edges (’a’, ‘b’), (’b’, ‘c’), and (’c’, ‘a’), which appear most frequently two times in the graph. However, they are overlapping again. Finally, it selects the candidate sets of (C), vertices {’a’, ‘b’, ‘b’} and two edges of (’a’, ‘b’), which appear also most frequently. (D) shows the graph contracted from the graph (A) by substituting the selected vertices and edges with a new vertex labeled with ‘abb’.

CompressVertices(2) of $M = 2$ is equivalent to CompressEdge. It should be noted that the algorithm uniquely determines the contracted sets of connected edges in each iteration although different subgraphs can be substituted with vertices of the same label in $M \ge 4$. For example, both graphs of a-b-a-b and a-a-b-b are represented as ((2, ‘b’), (2, ‘a’), (1, ‘b’), (1, ‘a’)) in dl-sequence. It is to be noted that for three vertices, different subgraphs cannot have the same dl-sequence. The algorithm is still efficient because it compares dl-sequences instead of comparing subgraphs, where subgraph isomorphism problem is known to be in NP-complete.

Similarity measure

The universal similarity metric (USM) was proposed by Li *et al.*[11], and has been applied to several biological data [12,13]. USM between two objects o_1 and o_2 is defined using Kolmogorov complexity $K(o)$ as follows:

$$USM(o_1, o_2) = \frac{\max(K(o_1 \mid o_2^*), K(o_2 \mid o_1^*))}{\max(K(o_1), (o_2))}, \quad (1)$$

where o_1^*, o_2^* denote shortest programs for generating o_1,o_2, respectively.

Kolmogorov complexity $K(o)$ of an object o is defined to be the length of the shortest program P for a universal Turing machine U which outputs o, and the conditional Kolmogorov complexity of o_1 given o_2 is defined to be the length of the shortest program P which outputs o_1 when o_2 is given as follows:

$$\begin{cases} K(o) = \min\{|P| \mid P \text{ is a program such that } U(P) = o\}, \\ K(o_1, o_2) = \min\{|P| \mid P \text{ is a program such that } U(P, o_2) = o_1\}. \end{cases} \quad (2)$$

It should be noted that $K(o)$ is considered as a measure of the amount of information that the object o contains.

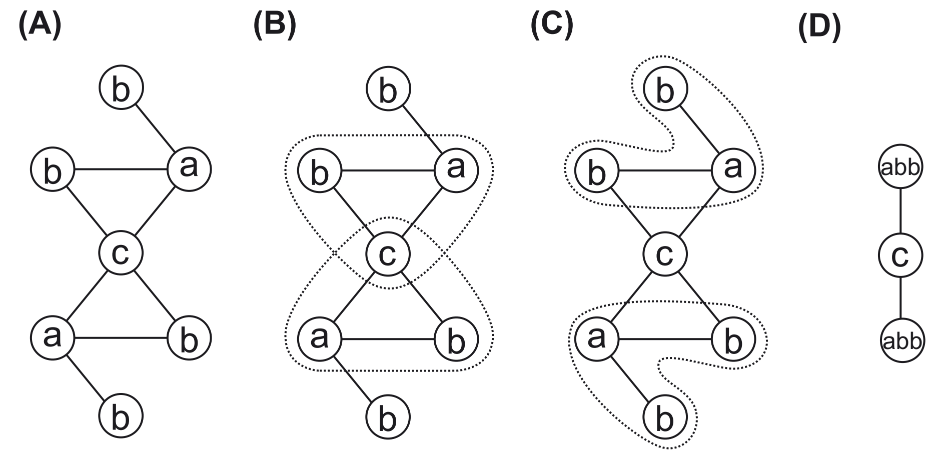

Figure 2 Example of contraction by CompressVertices Example of contraction of identical sets of vertices and the edges between the vertices by CompressVertices. (A) Graph with 7 vertices and 8 edges. (B) Candidate vertices {' a', 'b', 'c '} and three edges ('a', 'b'), ('b', 'c'), and ('c', 'a') to be contracted. (C) Candidate vertices {'a', 'b', 'b'} and two edges of ('a', 'b') to be contracted. (D) Graph contracted from graph (A) by substituting vertices {'a', 'b', 'b'} and the edges with 'abb'.

Since it is not possible to obtain these Kolmogorov complexities for real data, we approximate $K(G)$ of a graph G by $C(G) = |R| + |E_C|$, where $|R|$ means the number of rules extracted from G by our method, and $|E_C|$ means the number of remaining edges after the compression of G. The conditional Kolmogorov complexity $K(G_1|G_2)$ of G_1 given G_2 can be approximated to be $C(G_1 \cup G_2) - C(G_2)$ as in [12,13], where $G_1 \cup G_2$ means the concatenated graph $G'(V', E')$ of $G_1(V_1, E_1)$ and $G_2(V_2, E_2)$ such that $V' = V_1 \cup V_2$, $E' = E_1 \cup E_2$, $|V'| = |V_1| + |V_2|$ and $|E'| = |E_1| + |E_2|$. Even if there are identical vertices (i.e. vertices with identical labels) between G_1 and G_2, they are added to V' as different vertices.

Substituting $K(o)$ of Eq.(1) with $C(G)$, the approximated USM for graph compression, GUSM, between two graphs G_1 and G_2 is given as follows:

$$GUSM(G_1, G_2) = \frac{C(G_1 \cup G_2) - \min(C(G_1), C(G_2))}{\max(C(G_1), C(G_2))}. \quad (3)$$

It should be noted that $GUSM(G, G) = 0$ if $|E_c| = 0$ for $C(G)$. If G_1 and G_2 are similar, then G_1 and G_2 are generated from almost the same set of rules R, that is, $C(G_1 \cup G_2) \approx C(G_1) \approx C(G_2) \approx |R|$, and $GUSM(G_1, G_2)$ approaches to 0.

Results and discussion

To evaluate the proposed measure, we used metabolic pathways for several organisms, *H. sapiens, M. musculus, A. thaliana, D. melanogaster, C. elegans, E. coli, S. cerevisiae,* and *B. subtilis,* from the KEGG database [20] (see Table 1). We used chemical compounds as nodes and chemical reactions as edges. For evaluation purposes, it is not appropriate to use protein-protein interaction networks because the accuracy of the networks is not sufficient [21]. Furthermore, we compared the results with those of GRAPHITOUR because other methods such as PathBLAST [5], NetAlign [6], and MNAligner [7] do not give the similarity of networks, and focus on finding interesting subnetworks or most similar subnetworks to a query network.

In our first computational experiment, all nodes in the metabolic networks were labeled with chemical compounds, and there was only one edge having the same labels, that is, $|\varepsilon(l_1, l_2)| = 1$. Then, our compression algorithm for $G(V, E)$ produced rules R and the remaining graph $G_c(V_c, E_c)$ as $C(G) = |R| + |E_c| = |E|$. This means that G is not compressed. Many other methods also cannot select frequent subnetworks for such networks.

Since we would like to compare network structures for the metabolic networks, we replaced labels with Morgan index [18]. Figure 3 shows an example of

Table 1 Statistics of metabolic pathways

organism	# nodes	# edges
H. sapiens	1550	1673
M. musculus	1518	1640
A. thaliana	1389	1395
D. melanogaster	1238	1250
C. elegans	1049	1009
E. coli	1103	1256
S. cerevisiae	983	1028
B. subtilis	994	1065

Statistics of metabolic pathways for several organisms, *H. sapiens, M. musculus, A. thaliana, D. melanogaster, C. elegans, E. coli, S. cerevisiae,* and *B. subtilis.*

calculation of Morgan index. First, 1 is assigned to each node. Next, the sum of values of adjacent nodes is assigned for each node. This iteration is repeated until the number of different values of Morgan index does not increase. We call the index obtained in this way the original Morgan index. Morgan index obtained by one iteration of this procedure is equivalent to the degree of each node, and Morgan index depends on graph structures.

We fixed the number of iterations of the Morgan index procedure, applied our compression algorithms to individual and concatenated metabolic networks, G_1, G_2, $G_1 \cup G_2$, and calculated $GUSM$ (G_1, G_2) from $C(G_1)$, $C(G_2)$ and $C(G_1 \cup G_2)$. To confirm that our compression algorithms work for measuring the similarity of metabolic networks, we obtained hierarchical clustering results using the nearest neighbor (single linkage) method, and compared them with actual phylogenetic trees and hierarchical clustering results by GRAPHITOUR. Moreover, we performed such experiments with several numbers of iterations from 1 to 20 because the number of iterations of the original Morgan index is at most 11 for the metabolic networks.

Figure 4 shows the results of hierarchical clustering using CompressEdge for metabolic networks of several organisms, *H. sapiens, M. musculus, A. thaliana, D. melanogaster, C. elegans, E. coli, S. cerevisiae,* and *B. subtilis* with Morgan indices of 1, 2, 3, 6, 11, and 12 iterations. The numbers of contracted edges for the metabolic network of H. sapiens with Morgan indices of 1, 2, 3, 6, 11, and 12 iterations were 251, 1367, 1387, 1395, 1395, and 1395, respectively. The results of more than 5 iterations were almost similar to those of 12 iterations. Figure 5 shows the results on the number of different values of Morgan indices for the metabolic networks for 1-20 iterations of the Morgan index procedure. We can see from this figure that the number of different values of Morgan indices is almost constant for more than 11 iterations. For a small number of iterations, it is considered that metabolic networks were not compressed well because many edges have the same labels and share common nodes. This means that the number of iterations is required to be large for measuring the similarity more accurately. However, for that purpose, much larger numbers than the number of iterations of the original Morgan indices, that is at most 11, are not needed because the number of different values of Morgan indices is almost constant for more than 11 iterations (see Figure 5). According to the results of hierarchical clustering in Figure 4, *H. sapiens* was always the nearest to *M. musculus.* Bacterial organisms of *B. subtilis* and *E. coli* were furthest from *H. sapiens* in the result of 12 iterations. It is considered that the result of 12 iterations is almost consistent to actual phylogenetic trees. This suggests that the proposed method can adequately measure the similarities between metabolic networks.

Figure 6 shows the results of hierarchical clustering using CompressVertices(3) for metabolic networks. The result of 3 iterations was already similar to the results of more than 5 iterations. It is considered that there are more overlapping edges in networks for a smaller number of iterations, and our proposed method contracted their edges well for a small number of iterations.

Figure 7 shows the results of hierarchical clustering using GRAPHITOUR for metabolic networks. In the result of 10 iterations, *E. coli* was farthest from *H. sapiens*. The result of 12 iterations was not well clustered because *D. melanogaster* is likely to be closer to

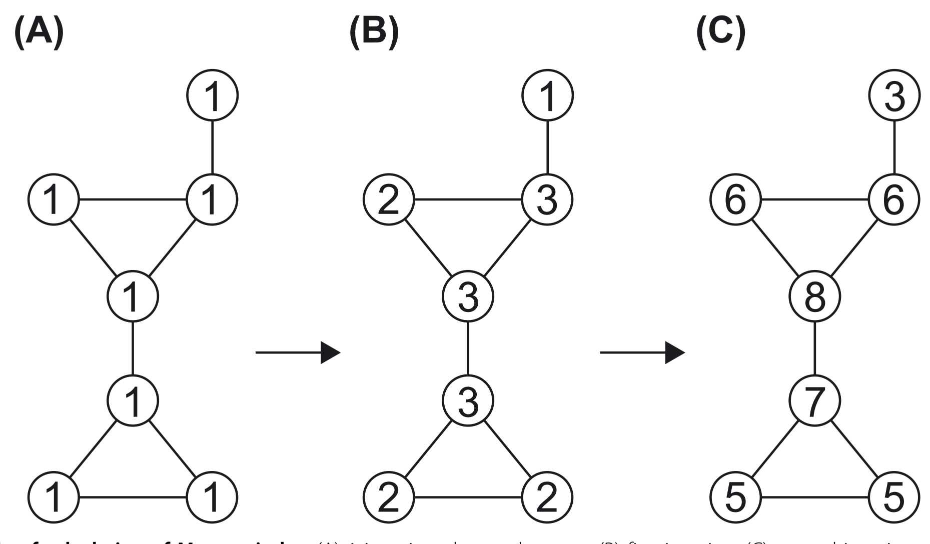

Figure 3 Example of calculation of Morgan index. (A) 1 is assigned to each vertex. (B) first iteration. (C) second iteration.

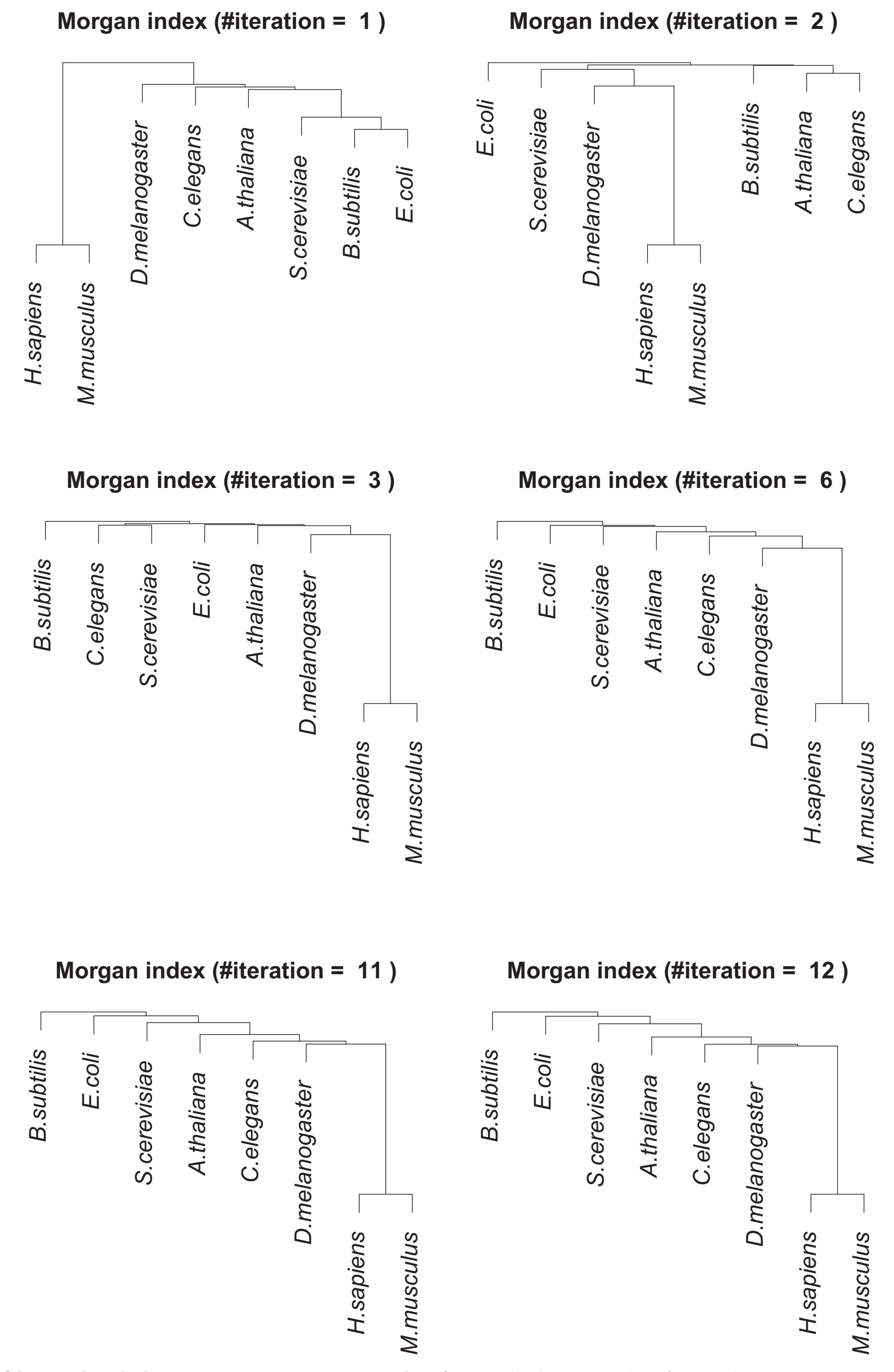

Figure 4 Results of hierarchical clustering using CompressEdge for metabolic networks of several organisms, *H. sapiens, M. musculus, A. thaliana, D. melanogaster, C. elegans, E. coli, S. cerevisiae,* and *B. subtilis* with Morgan indices of 1, 2, 3, 6, 11, and 12 iterations.

H. sapiens than to *C. elegans* [22]. These suggest that our proposed method can measure the similarities better than GRAPHITOUR.

Furthermore, the proposed methods are efficient. The computational time was at most 9 seconds in CompressEdge, 17 seconds in CompressVertices(3), even for the concatenated network of *H. sapiens* and *M. musculus* with Morgan index of 12 iterations. These experiments were done in a single processor core on a PC with Xeon X5460 3.16GHz CPUs and 8GB memory under the Linux (version 2.6) operating system, where the g++ compiler was used with optimization option -O3.

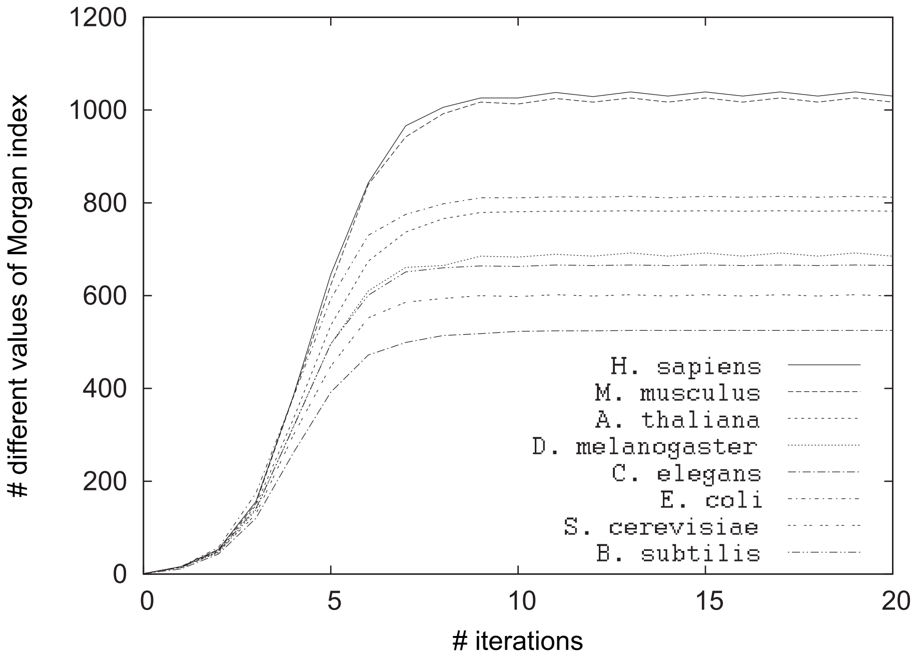

Figure 5 Results on the number of different values of Morgan indices for metabolic networks of several organisms, *H. sapiens, M. musculus, A. thaliana, D. melanogaster, C. elegans, E. coli, S. cerevisiae,* and *B. subtilis* for 1-20 iterations of the Morgan index procedure.

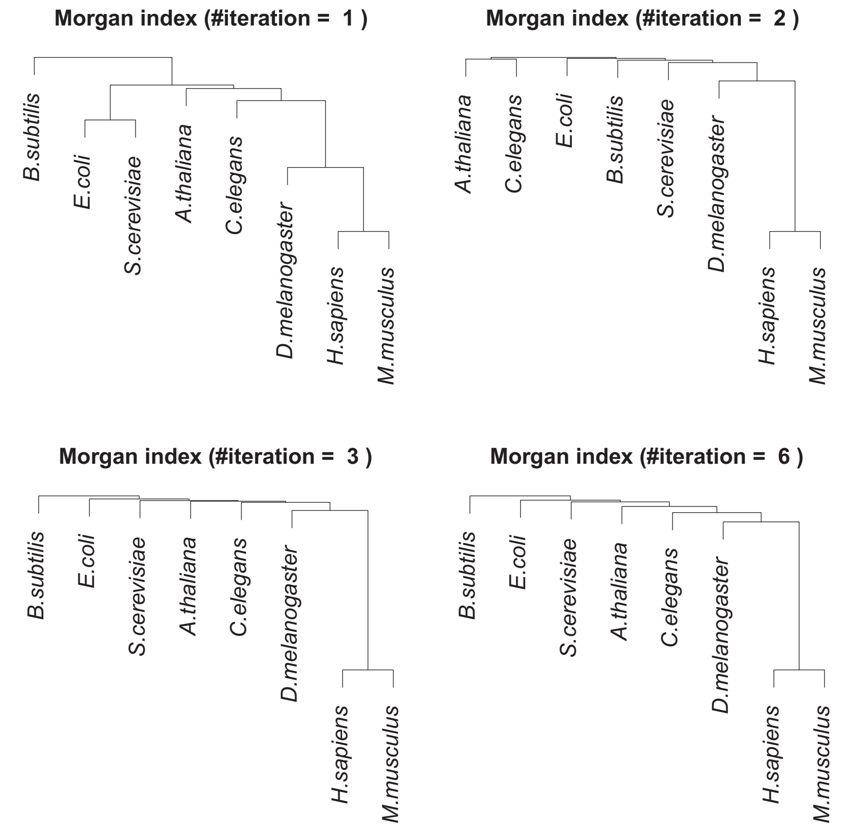

Figure 6 Results of hierarchical clustering using CompressVertices(3) for metabolic networks of several organisms, *H. sapiens, M. musculus, A. thaliana, D. melanogaster, C. elegans, E. coli, S. cerevisiae,* and *B. subtilis* with Morgan indices of 1, 2, 3, and 6 iterations.

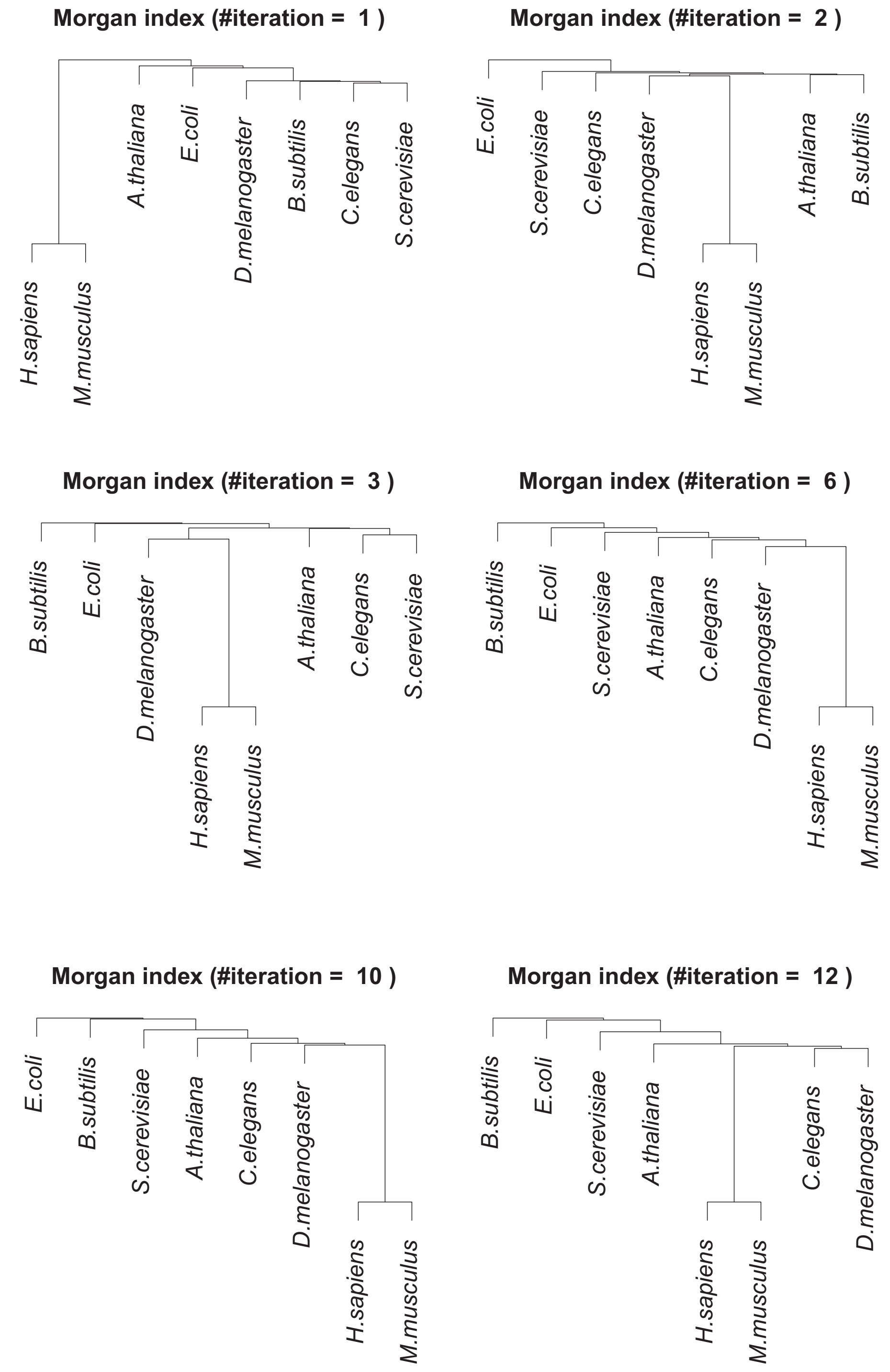

Figure 7 Results of hierarchical clustering using GRAPHITOUR for metabolic networks of several organisms, *H. sapiens, M. musculus, A. thaliana, D. melanogaster, C. elegans, E. coli, S. cerevisiae,* and *B. subtilis* with Morgan indices of 1, 2, 3, 6, 10, and 12 iterations.

We also compared our proposed methods with simple methods based on node and edge numbers. We calculated distances between two organisms as the differences of the number of nodes or edges, and obtained hierarchical clustering results using the nearest neighbor method. We can see from the results (Figures S1 and S2 in the supplementary information web page) that *H. sapiens* was the nearest to *M. musculus*, but other relations were different from actual phylogenetic trees. This result shows that our proposed methods are better than those based on node or edge numbers. In addition, our methods generate rules to contract edges iteratively. It means that they try to find hierarchical substructures for a given graph. We defined the similarity between graphs, GUSM, based on the number of such rules. It can be considered that GUSM implies whether hierarchical substructures are also similar, or not.

We proposed two methods, CompressEdge and CompressVertices(M), where M denotes the maximum number of vertices to be contracted at a time. It should be noted that the minimum number is 2 because a vertex itself cannot be contracted. CompressEdge is equivalent to CompressVertices(2). If a graph has many overlapping edges, CompressEdge (and CompressVertices(2)) would not extract many rules, that is, many edges are not contracted and are remained. Consider a graph G that all edges of G are overlapping. The similarity between G and itself, $GU\ SM\ (G, G)$, must be 0. However, $GU\ SM\ (G, G) = 1$ because any edge of G is not contracted, $C\ (G) = |E_c|$ and $C\ (G \cup G) = 2|E_c|$. Therefore, CompressVertices(M) for $M > 2$ is needed.

Conclusions

In this paper, we have proposed novel methods for compressing biological networks. One of the important properties of the proposed methods is that isomorphic networks are compressed in the same way. Unlike many other methods comparing biological networks, our methods are able to give the similarity metric of their networks. Moreover, our methods are very efficient because they do not compare subgraphs directly. We have applied the proposed compression methods to comparison of metabolic networks, and compared the results with those of an existing method. The results suggest that the proposed compression methods are useful for comparison of biological networks.

Our proposed methods were applied to only undirected graphs in this paper. However, it is possible to extend our algorithms to deal with directed graphs. It is easy to extend CompressEdge, and the modified method can still be efficient. Our methods are efficient as well as GRAPHITOUR, and it is important to keep the efficiency after extending them. Metabolic networks can also be represented as directed graphs and undirected graphs using chemical compounds as nodes and the relation between compounds (e.g., involve the same reaction) as edges. It is an interesting issue to examine whether our methods are robust for any representation of a network.

It is considered in general that random networks are more difficult to be compressed than scale-free networks. However, our methods cannot compress metabolic networks that are known as scale-free networks because all nodes in metabolic networks are labeled with distinct chemical compounds, and there is only one edge having the same labels. Our proposed methods are useful only for comparison of networks. If the same labels can appear multiple times, it is expected that our methods can also differentiate these networks. However, it is difficult to compare them in a simple way because the compression size depends on the distribution of node labels. Since we do not have realistic models for generation of random and scale-free networks with node labels, application of our proposed methods to differentiation of random networks from scale-free networks is left as future work.

Though we have applied the methods to comparison of networks, the application is not limited to comparison. It might be applied to detection of network motifs with hierarchical structures because our method iteratively compresses edges (edges can be replaced by small subgraphs).

One drawback of our proposed compression methods is that it is not a lossless compression method (i.e., the original network cannot be reconstructed from compressed data). Therefore, improvement of the method towards lossless compression is also important future work.

Availability

The source code of CompareVertices is available through the supplementary information web page (http://sunflower.kuicr.kyoto-u.ac.jp/morihiro/gracomp/).

Acknowledgements
This work was partially supported by Grants-in-Aid #19650053 and #21700323 from MEXT, Japan.
This article has been published as part of *BMC Systems Biology* Volume 4 Supplement 2, 2010: Selected articles from the Third International Symposium on Optimization and Systems Biology. The full contents of the supplement are available online at http://www.biomedcentral.com/1752-0509/4?issue=S2

Authors' contributions
Methods were developed by MH and TA. The manuscript was prepared by MH and TA.

Competing interests
The authors declare that they have no competing interests.

Published: 13 September 2010

References

1. Ogata H, Fujibuchi W, Goto S, Kanehisa M: **A heuristic graph comparison algorithm and its application to detect functionally related enzyme clusters.** *Nucleic Acids Research* 2000, **28**:4021-4028.
2. Tohsato Y, Matsuda H, Hashimoto A: **A multiple alignment algorithm for metabolic pathway analysis using enzyme hierarchy.** *In Proc. 8th Int. Conf. Intelligent Systems for Molecular Biology* Bourne PE, Gribskov M, Altman RB, Jensen N, Hope DA, Lengauer T, Mitchell JC, Scheeff ED, Smith C, Strande S, Weissig H 2000, 376-383.
3. Pinter RY, Rokhlenko O, Yeger-Lotem E, Ziv-Ukelson N: **Alignment of metabolic pathways.** *Bioinformatics* 2005, **21**:3401-3408.
4. Wernicke S, Rasche F: **Simple and fast alignment of metabolic pathways by exploiting local diversity.** *Bioinformatics* 2007, **23**:1978-1985.

5. Kelley BP, Yuan B, Lewitter F, Sharan R, Stockwell BR, Ideker T: **PathBLAST: a tool for alignment of protein interaction networks.** *Nucleic Acids Research* 2004, **32**:W83-W88.
6. Liang Z, Xu M, Teng M, Niu L: **NetAlign: a web-based tool for comparison of protein interaction networks.** *Bioinformatics* 2006, **22**:2175-2177.
7. Li Z, Zhang S, Wang Y, Zhang XS, Chen L: **Alignment of molecular networks by integer quadratic programming.** *Bioinformatics* 2007, **23**:1631-1639.
8. Singh R, Xu J, Berger B: **Global alignment of multiple protein interaction networks with application to functional orthology detection.** *Proc. Natl. Acad. Sci. USA* 2008, **105**:12763-12768.
9. Zaslavskiy M, Bach F, Vert JP: **Global alignment of protein-protein interaction networks by graph matching methods.** *Bioinformatics* 2009, **25**:i259-i267.
10. Kocsor A, Kertész-Farkas A, Kaiàn L, Pongor S: **Application of compression-based distance measures to protein sequence classification: a methodological study.** *Bioinformatics* 2005, **22**:407-412.
11. Li M, Badger JH, Chen X, Kwong S, Kearney P, Zhang H: **An information-based sequence distance and its application to whole mitochondrial genome phylogeny.** *Bioinformatics* 2001, **17**:149-154.
12. Hayashida M, Akutsu T: **Image compression-based approach to measuring the similarity of protein structures.** *In Proc. 6th Asia-Pacific Bioinformatics Conference* Brazma A, Miyano S, Akutsu T 2008, 221-230.
13. Krasnogor N, Pelta DA: **Measuring the similarity of protein structures by means of the universal similarity metric.** *Bioinformatics* 2004, **20**:1015-1021.
14. Adler M, Mitzenmacher M: **Towards compressing web graphs. In Proc. 2001 Data Compression Conference** .Storer JA, Cohn M 2001, 203-212.
15. Peshkin L: **Structure induction by lossless graph compression.** *In Proc. 2007 Data Compression Conference* Storer JA, Cohn M 2007, 53-62.
16. Cook DJ, Holder LB: **Substructure discovery using minimum description length and background knowledge.** *Journal of Artificial Intelligence Research* 1994, **1**:231-255.
17. Yang J, Savari SA, Mencer O: **An approach to graph and netlist compression. In Proc. 2008 Data Compression Conference.**Storer JA, Cohn M 2008, 33-42.
18. Morgan H: **The generation of unique machine description for chemical structures - a technique developped at chemical abstracts service.** *Journal of Chemical Documentation* 1965, 107-113.
19. Diestel R: **In Graph Theory.** Heidelberg: Springer-Verlag, 3278.
20. Kanehisa M, Araki M, Goto S, Hattori M, Hirakawa M, Itoh M, Katayama T, Kawashima S, Okuda S, Tokimatsu T, Yamanishi Y: **KEGG for linking genomes to life and the environment.** *Nucleic Acids Research* 2008, **36**: D480-D484.
21. Ito T, Chiba T, Ozawa R, Yoshida M, Hattori M, Sakaki Y: **A comprehensive two-hybrid analysis to explore the yeast protein interactome.** *Proc. Natl. Acad. Sci., USA* 2001, **98**:4569-4574.
22. Mushegian AR, Garey JR, Martin J, Liu LX: **Large-scale taxonomic profiling of eukaryotic model organisms: A comparison of orthologous proteins encoded by the human, fly, nematode, and yeast genomes.** *Genome Research* 1998, **8**:590-598.

doi:10.1186/1752-0509-4-S2-S13
Cite this article as: Hayashida and Akutsu: **Comparing biological networks via graph compression.** *BMC Systems Biology* 2010 **4**(Suppl 2): S13.