Proteome Compression via Protein Domain Compositions

Morihiro Hayashida[a], Peiying Ruan[a], Tatsuya Akutsu[a]

[a] *Bioinformatics Center, Institute for Chemical Research, Kyoto University, Gokasho, Uji, Kyoto, 611-0011, Japan*

Abstract

In this paper, we study domain compositions of proteins via compression of whole proteins in an organism for the sake of obtaining the entropy that the individual contains. We suppose that a protein is a multiset of domains. Since gene duplication and fusion have occurred through evolutionary processes, the same domains and the same compositions of domains appear in multiple proteins, which enables us to compress a proteome by using references to proteins for duplicated and fused proteins. Such a network with references to at most two proteins is modeled as a directed hypergraph. We propose a heuristic approach by combining the Edmonds algorithm and an integer linear programming, and apply our procedure to fourteen proteomes of *D. discoideum*, *E. coli*, *S. cerevisiae*, *S. pombe*, *C. elegans*, *D. melanogaster*, *A. thaliana*, *O. sativa*, *D. rerio*, *X. laevis*, *G. gallus*, *M. musculus*, *P. troglodytes*, and *H. sapiens*. The compressed size using both of duplication and fusion was smaller than that using only duplication, which suggests the importance of fusion events in evolution of a proteome.

Keywords: grammar-based compression, protein domain composition, integer linear programming

1. Introduction

A living individual is considered to be an open non-equilibrium system from the viewpoint of statistical mechanics. In an isolated system, the entropy increases according to the second law of thermodynamics. On the other hand, in an open system, a dissipative structure is constructed, and it reaches a reproducible steady state [1]. The DNA base sequences in an individual

are one kind of information to be maintained under non-equilibrium environments, whereas the sequences have been mutated and substituted through evolutionary processes. If random mutation and substitution of bases were always allowed from one generation to another, then the resulted sequences would be completely random. It can be considered that the case corresponds to the isolated system in statistical mechanics, and the entropy of the sequence is maximized.

There are several studies to compress DNA and protein sequences, which might be useful to study the entropy of these sequences. It is known that DNA sequences include abundant repetition and palindromes. Grumbach and Tahi [2] developed the first compression method specified to DNAs, called biocompress-2, which is a lossless algorithm using Lempel and Ziv's approaches [3, 4], and tries to detect repeats and palindromes in DNA sequences. Rivals et al. [5] developed the Cfact algorithm, which uses suffix trees, and tries to detect the longest exact matching repeat. Chen et al. [6] developed the DNACompress algorithm, which tries to detect approximate repeats using some efficient method. Willems et al. [7] developed the context-tree weighting (CTW) method, which was defined as a suffix tree with edges weighted by some occurrence probability. Matsumoto et al. [8] proposed combination methods with CTW for DNA and protein sequences, respectively. Cao et al. [9] proposed an expert model (XM) based on statistical properties and repetition within sequences, and their method outperformed all other DNA and protein sequence compressors. Zhu et al. [10] proposed an approximate repeat vector (ARV) model forming a reference codebook for compression of DNA sequences, and developed an adaptive particle swarm optimization-based memetic algorithm (POMA) to maximize the cover rate and minimize some distance of the code vectors on the sequences. Kuruppu et al. [11] proposed COMRAD (COMpression using RedundAncy of Dna) by adapting an existing compression algorithm, RAY [12], to DNA sequences using some knowledge about alphabet size and sequence evolution. Their method outperformed RLCSA [13] and RLZ [14] for several organisms, and was effective in long-range repetition detection. Compression of sequences in multiple organisms of the same species has been also studied as genomic repositories are rapidly growing. For the purpose, most methods compress such sequences using difference from reference sequences [15, 16]. Unlike these compression methods, we deal with protein domain compositions instead of amino acid sequences. Many proteins contain domains, which are known as functional and structural units in proteins [17]. In addition, the

same domain can be included in multiple distinct kinds of proteins. Furthermore, we make use of compression for understanding of evolution by measuring entropies of proteomes, not for saving memory and disk space. We regard a protein as a multiset of domains, and compress sets of proteins. As far as we know, this is the first study to compress a proteome using such domain compositions.

Several studies have been done for evolution of protein domains [18, 19]. Gene duplication can occur when an mRNA is retrotranscribed to cDNA and randomly inserted into the genome [20]. As a result, the protein can be generated also from the duplicated gene in a different chromosome, and has evolved independently from the original one. It is known that unequal crossing-over induces another gene duplication [20]. If positions of hybridization in crossing-over are not the same between two strands, genes in the strand can be also duplicated. Fusion and fission of genes are evolutionary events that two or more genes in an organism are connected and compose a gene in a descendent organism, and, in contrast, that a gene is split into multiple genes, respectively. Kummerfeld and Teichmann applied their maximum parsimony method to several completely sequenced genomes, and reported that the number of fusion events is about fourfold larger than that of fission events [21]. The number of total domains and the number of domain families in a protein follow power-law and exponential distributions in many organisms, respectively. Nacher et al. [22] proposed evolutionary models including gene duplication, fusion, and internal duplication events to explain both distributions. It should be noted that the internal duplication event is also known as tandem repeats within a gene, and is not different from the usual external gene duplication event [23]. Thus, compressing a proteome is considered to be possible because genes and domains have been duplicated through evolutionary processes. We make use of gene duplication and fusion events, generate a directed hypergraph with weighted hyperedges from a proteome, and try to find the minimum spanning hypertree, where each vertex corresponds to a protein, and an edge weight represents the compressed cost for a protein using some proteins. However, Brejová et al. [24] showed that the problem of finding the minimum directed spanning hypertree in the hypergraph is NP-hard even if each hyperedge has at most three vertices. In addition, they proposed an integer linear programming (ILP) formulation and applied it to the problem of maximizing some likelihood function for detecting signals in DNA, which are short subsequences located near functional sites. Although ILP outputs the exact optimum, if we consider the

gene fusion events that correspond to hyperedges with size three, the execution time is too long to obtain the solution because the number of pairs of proteins in an organism is large. Hence, we propose a greedy method to reduce the number of hyperedges for compressing a proteome. Our method first finds the optimum solution for a graph with usual edges. To this end, it minimizes a cost function, which means the compressed size, and is based on the cost function proposed by Adler and Mitzenmacher [25] for compressing web graphs. After that, hyperedges with size three are added to the solution in some heuristic way. We apply our proposed method to fourteen organisms of *D. discoideum*, *E. coli*, *S. cerevisiae*, *S. pombe*, *C. elegans*, *D. melanogaster*, *A. thaliana*, *O. sativa*, *D. rerio*, *X. laevis*, *G. gallus*, *M. musculus*, *P. troglodytes*, and *H. sapiens*. The results suggest that the same domain would be frequently utilized in higher organisms.

2. Method

In this section, we formulate our problem for compressing a proteome, briefly review the integer linear programming (ILP)-based method for minimum spanning directed hypertree problems [24], and describe our proposed heuristic method.

2.1. Problem formulation

Let $\mathcal{P}$ and $\mathcal{D}$ be the set of proteins and domains in a given proteome, respectively. Each protein P_i ($\in \mathcal{P}$) consists of several domains in $\mathcal{D}$, and is supposed to represent a multiset. For instance, if P_i consists of two D_1s and one D_2, then $P_i = \{D_1, D_1, D_2\}$. We define the cost representing a protein using only domains by

$$cost(P_i) \quad = \quad \lceil \log |\mathcal{D}| \rceil \cdot |P_i|, \tag{1}$$

where $|S|$ denotes the number of elements in the (multi) set S, and $\lceil x \rceil$ is the minimum integer no smaller than x. Then, a proteome without compression is represented with size $\sum_{P_i \in \mathcal{P}} cost(P_i)$.

Adler and Mitzenmacher [25] considered the cost generating a web page using another page as follows. A web page consists of links to other web pages. A new page is often created by copying some links from an existing page to itself. This is similar to gene duplication that a new gene is created by copying an existing gene. They used a 0-1 vector each element of which

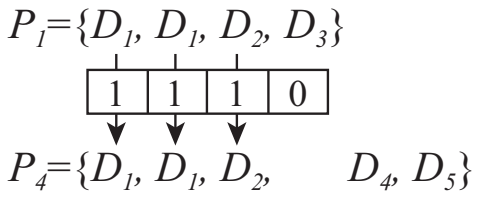

Figure 1: Illustration of $cost(P_i, P_j)$ for $P_1 = \{D_1, D_1, D_2, D_3\}$, $P_4 = \{D_1, D_1, D_2, D_4, D_5\}$, $\mathcal{P} = \{P_1, P_2, P_3, P_4\}$, and $\mathcal{D} = \{D_1, D_2, D_3, D_4, D_5\}$. The rectangle denotes a 0-1 vector representing absence (0) or presence (1) of domains.

represents absence or presence of a web page linked from the existing page. The cost includes the size of the vector, the length representing the existing page, and new links not included in the existing one. For our purpose, the cost generating a protein P_i from another protein P_j by deleting and/or adding domains appropriately, $P_j \to P_i$, is defined by

$$cost(P_i, P_j) \;=\; \lceil \log |\mathcal{P}| \rceil + |P_j| + \lceil \log |\mathcal{D}| \rceil \cdot |P_i - P_j|, \tag{2}$$

where $|P_i - P_j|$ denotes the number of domains of P_i that are not included in P_j, and $|P_j|$ means the size of the 0-1 vector representing whether or not each domain in P_j is included in P_i. If the gene coding P_i is duplicated from that coding P_j, $cost(P_i, P_j)$ can be smaller than $cost(P_i)$ and costs for duplication and fusion from other genes. Fig. 1 illustrates the cost generating $P_4 = \{D_1, D_1, D_2, D_4, D_5\}$ from $P_1 = \{D_1, D_1, D_2, D_3\}$ in a proteome $\mathcal{P} = \{P_1, P_2, P_3, P_4\}$ with $\mathcal{D} = \{D_1, D_2, D_3, D_4, D_5\}$. The rectangle denotes a 0-1 vector representing absence (0) or presence (1) of domains of P_1 in P_4, two D_1s and D_2 of P_1 remain in P_4, and D_3 disappears in P_4. Furthermore, D_4 and D_5 are added. Thus, $cost(P_4, P_1) = \lceil \log 4 \rceil + 4 + \lceil \log 5 \rceil \cdot 2 = 12$.

In addition to gene duplication events, we consider gene fusion events that two different genes in an organism are fused into one gene in a descendent organism [21]. Then, we consider to generate a protein P_i using two proteins P_j and P_k. Since the number of combinations of three proteins is large for an actual proteome, for instance, the number is $\binom{1000}{3} = 166167000$ if $|\mathcal{P}| = 1000$, we consider only the case that P_i completely contains both proteins, that is, $P_j \cup P_k \subseteq P_i$. Thus, we define the cost generating P_i from P_j and P_k by adding domains appropriately, $P_j + P_k \to P_i$, by

$$cost(P_i, P_j, P_k) \;=\; 2 \cdot \lceil \log |\mathcal{P}| \rceil + \lceil \log |\mathcal{D}| \rceil \cdot |P_i - P_j - P_k|. \tag{3}$$

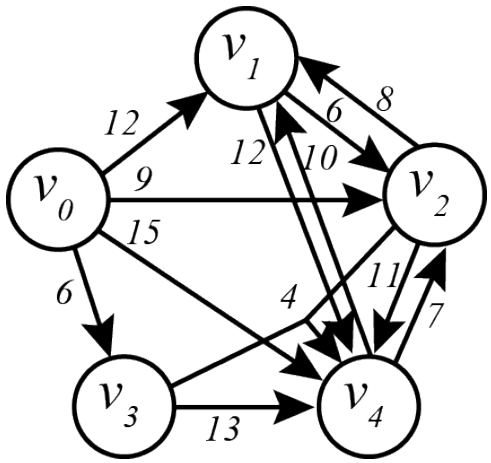

Figure 2: Example of the hypergraph generated from $\mathcal{P} = \{P_1, P_2, P_3, P_4\}$, where $\mathcal{D} = \{D_1, D_2, D_3, D_4, D_5\}$, $P_1 = \{D_1, D_1, D_2, D_3\}$, $P_2 = \{D_1, D_1, D_2\}$, $P_3 = \{D_4, D_5\}$, and $P_4 = \{D_1, D_1, D_2, D_4, D_5\}$. A number denotes the cost for the corresponding hyperedge.

In this equation, the first term of the right-hand side means the length representing P_j and P_k, and the second term means the length representing domains newly appeared in P_i. It should be noted that the 0-1 vector in the case of gene duplication is not needed because all the domains in P_j and P_k are used in P_i.

Let $G(V, E)$ be a directed hypergraph with a set V of vertices and a set E of hyperedges, where each hyperedge e $(= (L, h) \in E)$ has tail vertices L $(\subset V)$ and a head vertex h $(\in V)$, $h \notin L$ holds, and is weighted by w_e. In terms of compressing a proteome, the set V consists of vertices v_i $(\in V)$ corresponding to proteins P_i $(\in \mathcal{P})$ and a root vertex v_0 that means a special protein without any domains, then, $|V| = |\mathcal{P}| + 1$. The set E consists of $(\{v_0\}, v_i)$ with weight $cost(P_i)$, $(\{v_j\}, v_i)$ with weight $cost(P_i, P_j)$ if $cost(P_i, P_j) < cost(P_i)$, and $(\{v_j, v_k\}, v_i)$ with weight $cost(P_i, P_j, P_k)$ if $P_j \cup P_k \subseteq P_i$. Fig. 2 shows an example of the hypergraph generated from four proteins $\mathcal{P} = \{P_1, P_2, P_3, P_4\}$, where $\mathcal{D} = \{D_1, D_2, D_3, D_4, D_5\}$, $P_1 = \{D_1, D_1, D_2, D_3\}$, $P_2 = \{D_1, D_1, D_2\}$, $P_3 = \{D_4, D_5\}$, and $P_4 = \{D_1, D_1, D_2, D_4, D_5\}$. In this figure, each number denotes the cost for the corresponding hyperedge, for instance, for the hyperedge $(\{v_2, v_3\}, v_4)$, the weight is $cost(P_4, P_2, P_3) = 2 \cdot \lceil \log 4 \rceil + \lceil \log 5 \rceil \cdot 0 = 4$.

Since the hypergraph constructed as above contains many redundant hyperedges for representing a proteome, it can be considerably compressed. In order to obtain the minimum compressed size, the number of incident

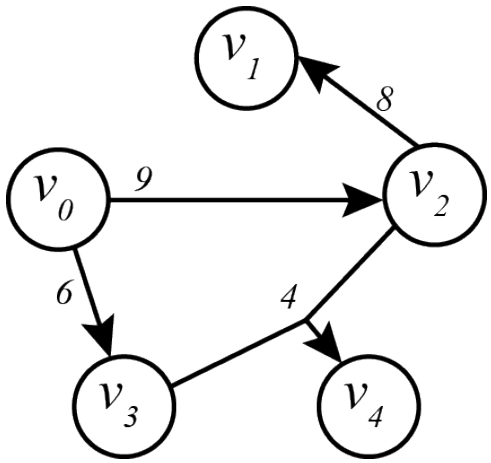

Figure 3: The minimum spanning directed hypertree for the example of Fig. 2.

hyperedges must be at most one for each vertex. It means that one rule is enough to generate the set of domains in a protein. In addition to this condition, to extract the original domain compositions from the hypergraph, the compressed hypergraph must not have a cycle. For instance, in Fig. 2, if hyperedges, $(\{v_1\}, v_2)$, $(\{v_4\}, v_1)$, and $(\{v_2\}, v_4)$, are selected, then P_2 is generated from P_1, P_1 is generated from P_4, P_4 is generated from P_2, and we cannot determine the sets of domains in P_1,P_2, and P_4. Hence, we can compress a proteome $\mathcal{P}$ by finding the minimum spanning directed hypertree $T(V, F)$ of $G(V, E)$ such that $F \subseteq E$, $|F| = |\mathcal{P}|$, and T has no cycle.

Problem 1. *Given a set of proteins $\mathcal{P}$ with domain compositions, find the minimum spanning directed hypertree for the hypergraph constructed from $\mathcal{P}$.*

Fig. 3 shows the minimum spanning directed hypertree for the example of Fig. 2. Then, the compressed size is 27.

2.2. Integer linear programming-based method

It has been shown that the problem of finding the minimum spanning directed hypertree $T(V, F)$ in the hypergraph $G(V, E)$ is NP-hard even if each hyperedge has at most three vertices, and an integer linear programming (ILP) formulation has been proposed [24]. Based on their formulation, we introduce the following ILP formulation by utilizing the set $\mathcal{S}$ of strongly connected components in G.

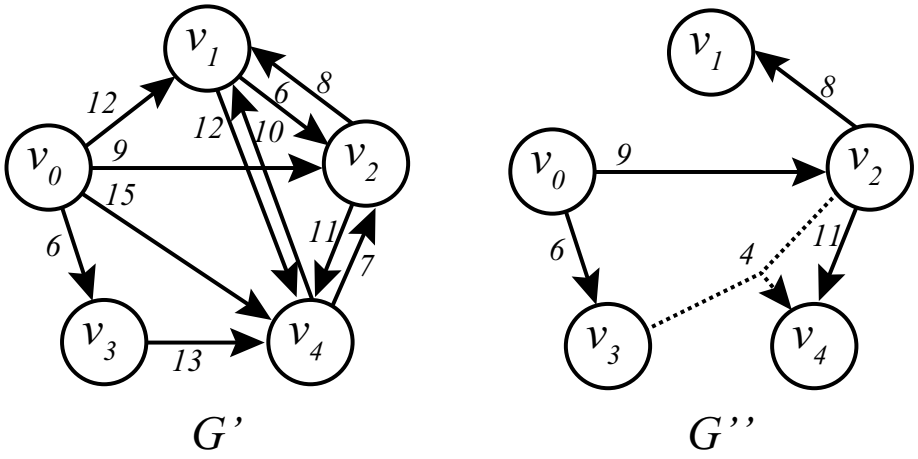

Figure 4: Illustration of our heuristic approach for the example of Fig. 2. Left) The graph G', each of whose edges consists of exactly two vertices. Right) The hypergraph G''. The solid arrows in G'' denote the edges F of the minimum spanning tree U for G' in Step 1. The dotted arrow denotes a hyperedge to be added to E'' in Step 2, whose weight, 4, is less than the weight, 11, of the edge pointing to v_4 in U.

$$
\begin{aligned}
&\text{minimize } \sum_{e \in E} w_e x_e, \\
&\text{subject to} \\
&\quad \sum_{\{e \in E | e = (L, v_i)\}} x_e = 1 \quad \text{for all } v_i \in V - \{v_0\}, \\
&\quad x_e \leq y_{j,i} \quad \text{for all } e = (L, v_i) \in E \text{ and } v_j \in L, \\
&\quad y_{i,j} + y_{j,i} = 1 \quad \text{for all } S \in \mathcal{S} \text{ and } v_i, v_j \in S, \\
&\quad y_{i,j} + y_{j,k} + y_{k,i} \leq 2 \quad \text{for all } S \in \mathcal{S} \text{ and } v_i, v_j, v_k \in S, \\
&\quad x_e, y_{i,j} \in \{0, 1\}.
\end{aligned}
$$

Here, $x_e = 1$ if the hyperedge e is selected as the optimal solution, otherwise $x_e = 0$. $y_{i,j} = 1$ means that v_i is on the upstream of v_j, otherwise $y_{i,j} = 0$ or $y_{j,i} = 1$. The fourth constraint forbids that all of $y_{i,j}$, $y_{j,k}$, and $y_{k,i}$ become 1 at the same time. It means that any cycle is not allowed. Although this ILP provides the exact solution, the execution time might be too long to obtain the solution because the number of pairs of proteins in an organism is large.

2.3. Heuristic approach

We propose a heuristic approach by reducing the number of hyperedges to obtain a smaller compressed size. If each hyperedge consists of exactly two

vertices, the minimum spanning tree can be found deterministically in time $O(|V| \log |V| + |E|)$ [26, 27, 28]. Hence, by making use of this polynomial time algorithm, we propose the following procedure for hypergraph $G(V, E)$.

Input: hypergraph $G(V, E)$ transformed from a proteome $\mathcal{P}$.
Output: hypertree representing the minimum grammar for the proteome.
Step 1. Find the minimum spanning tree $U(V, F)$ for $G'(V, E')$ in polynomial time, where E' is the set of all edges having exactly two vertices in E.
Step 2. Construct E'' that consists of the edges F and some hyperedges having three vertices, which is defined by $E'' = F \cup \{(\{v_j, v_k\}, v_i) \mid e = (\{v_j, v_k\}, v_i) \in E, w_e < w_f \text{ for } f = (\{v_l\}, v_i) \in F\}$.
Step 3. Find the set $\mathcal{S}$ of strongly connected components of hypergraph $G''(V, E'')$.
Step 4. Solve the ILP for $G''(V, E'')$ and $\mathcal{S}$.

Fig. 4 illustrates the procedure for the example of Fig. 2. The left figure shows the graph G' generated from the original hypergraph G by deleting hyperedges having three or more vertices, that is, $(\{v_2, v_3\}, v_4)$ in this case. The right figure shows the hypergraph G'', which contains the edges of the minimum spanning tree U for G' and hyperedges having three vertices with a smaller weight than the edge in U having the same head vertex, where the number of inedges for each vertex in U is one except v_0. In this example, there is only one hyperedge $(\{v_2, v_3\}, v_4)$. This hyperedge is included in E'' because the weight, 4, is less than the weight, 11, of the edge pointing to v_4 in U. By solving an ILP instance corresponding to $G''(V, E'')$, we obtain the solution shown in Fig. 3.

It should be noted that the procedure is not guaranteed to output the minimum spanning directed hypertree for G, and the time complexity can be exponential in the worst case because ILP instances in Step 4 include hypergraphs with hyperedges having at most three vertices. The source code in C++ implementation is available at our supplementary web page, http://sunflower.kuicr.kyoto-u.ac.jp/morihiro/pdcomp/.

3. Results

We used the UniProt database (release 2013_03) [29] to get the multiset of domains included in each protein, and used two databases, Pfam [30] and SMART [31], to define domains. We examined fourteen proteomes of *D. discoideum*, *E. coli*, *S. cerevisiae*, *S. pombe*, *C. elegans*, *D. melanogaster*, *A.*

Table 1: Results on the compressed size using Pfam domains for *D. discoideum*, *E. coli*, *S. cerevisiae*, *S. pombe*, *C. elegans*, *D. melanogaster*, *A. thaliana*, *D. rerio*, *X. laevis*, *M. musculus*, and *H. sapiens*.

organism	# prot	# dom	original	dup	both	$\|E''\|$	+
D. discoideum	3253	1772	52393	45688	45653	7786	24
E. coli	3979	2455	65784	58645	58605	5384	58
S. cerevisiae	4756	2707	77460	69936	69902	5345	31
S. pombe	4318	2751	70776	64969	64939	4550	26
C. elegans	2854	1735	44814	40665	40637	3042	21
D. melanogaster	2868	1916	46475	42484	42451	3054	24
A. thaliana	10836	2085	189936	154425	154401	25202	47
O. sativa	2872	926	41910	35509	35503	4742	7
D. rerio	2464	1600	37763	34784	34765	2730	14
X. laevis	2959	1664	46123	41776	41761	3139	12
G. gallus	2066	1423	35167	31510	31488	2410	18
M. musculus	14234	4353	282698	230511	—	53859	—
P. troglodytes	639	428	8181	7529	7527	644	2
H. sapiens	16204	4450	324428	260392	—	92494	—

'# prot' and '# dom' denote the numbers of proteins and domains, respectively. 'original' denotes the original size, 'dup' and 'both' denote the compressed sizes with duplication rules, and with both duplication and fusion rules, respectively. '—' denotes that the execution was not finished within ten hours. 'E''' denotes the set of hyperedges in G'' to be applied to the ILP in Step 4. '+' denotes the number of proteins that fusion rules are selected by our approach.

thaliana, *O. sativa*, *D. rerio*, *X. laevis*, *G. gallus*, *M. musculus*, *P. troglodytes*, and *H. sapiens*. We used CPLEX (version 12.5) to solve ILP instances.

Tables 1 and 2 show, for each organism, the numbers of proteins and Pfam/SMART domains, the original size, the compressed sizes with duplication rules, and with both duplication and fusion rules, the number of hyperedges applied to the ILP, and the number of proteins that fusion rules are selected, where proteins without any known domains were removed, and we dealt with P_i as a set, not a multiset. We can see from the tables that the compressed size with duplication rules was smaller than the original size for each organism, and furthermore the compressed size with both duplication

Table 2: Results on the compressed size using SMART domains for *D. discoideum*, *E. coli*, *S. cerevisiae*, *S. pombe*, *C. elegans*, *D. melanogaster*, *A. thaliana*, *D. rerio*, *X. laevis*, *M. musculus*, and *H. sapiens*.

organism	# prot	# dom	original	dup	both	$\|E''\|$	+
D. discoideum	1425	338	16857	15500	15499	2827	13
E. coli	769	242	8064	7378	7377	1010	5
S. cerevisiae	1662	449	19107	17874	17870	4192	16
S. pombe	1525	453	18225	16900	16894	3918	17
C. elegans	1339	439	17802	16018	16008	2233	26
D. melanogaster	1379	480	18711	16806	16787	2210	32
A. thaliana	4372	387	49716	46370	46363	5171	4
O. sativa	1296	176	13256	12283	12282	1305	1
D. rerio	1145	358	15192	13607	13604	1761	15
X. laevis	1645	404	19998	18292	18285	1992	16
G. gallus	1186	428	15966	14192	14186	1786	17
M. musculus	8153	753	123840	103632	103586	33007	77
P. troglodytes	315	154	3512	3253	3250	329	2
H. sapiens	9554	757	146970	124659	—	20661	—

'# prot' and '# dom' denote the numbers of proteins and domains, respectively. 'original' denotes the original size, 'dup' and 'both' denote the compressed sizes with duplication rules, and with both duplication and fusion rules, respectively. '—' denotes that the execution was not finished within ten hours. 'E''' denotes the set of hyperedges in G'' to be applied to the ILP in Step 4. '+' denotes the number of proteins that fusion rules are selected by our approach.

and fusion rules was slightly smaller than that with duplication rules. For *M. musculus* with Pfam domains and *H. sapiens*, we could not obtain the compressed size with both rules within ten hours under Xeon 2.67GHz CPU. In particular, we could not obtain the compressed size for *H. sapiens* using SMART domains even though the number of hyperedges E'' for *H. sapiens* was less than that for *M. musculus* (see Table 2). On the other hand, the number of variables $y_{i,j}$ in the ILP for *H. sapiens* was 406630, which was more than that for *M. musculus*, 144204. It means that the hypergraph $G''(V, E'')$ for *H. sapiens* was more complicated than that for *M. musculus*. For other organisms, the decrease of the compressed size with both rules was small. It

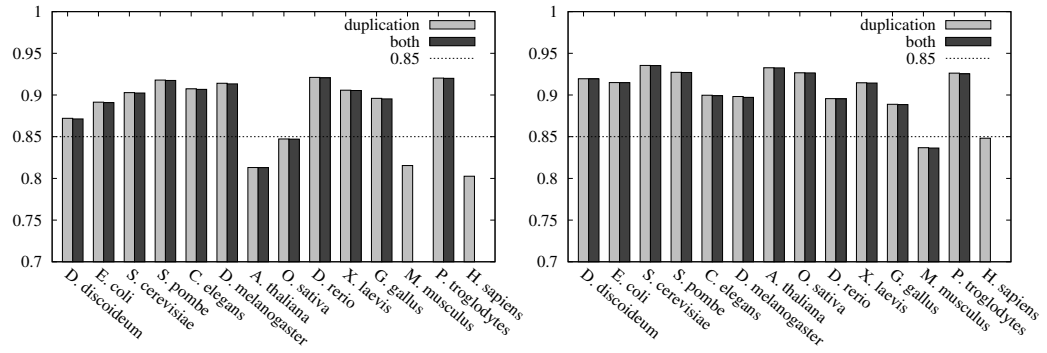

Figure 5: Results on the compression ratio by duplication rules and by both duplication and fusion rules. Left) Pfam domains. Right) SMART domains.

is considered because some hyperedges belonging to the minimum spanning directed hypertree for G were reduced before the ILP was applied.

It is intractable in many cases to solve the ILP directly for the original hypergraph $G(V, E)$ without reducing the number of hyperedges as in Steps 1 and 2. For evaluation of our method, we however tried to find the optimal solution for each dataset. Then, we obtained the optimal solution of the ILP only for the original hypergraph $G(V, E)$ of *P. troglodytes* with SMART domains, where the number of hyperedges $|E|$ was 761, and the minimum cost was 3249. We can see that the minimum cost was closed to the cost solved by our method, 3250, and our method was able to compress the proteome well although further evaluation using organisms with more domains and without restriction to the fusion rule is needed.

Fig. 5 compares the compression ratios by duplication rules and by both duplication and fusion rules using Pfam and SMART domains, which were calculated from Tables 1 and 2. For Pfam domains, the ratios of the compressed size with duplication rules to the original size in *A. thaliana*, *O. sativa*, *M. musculus*, and *H. sapiens*, 80% to 85%, were lower than those in other organisms, 87% to 92%. It is considered that gene duplication in *A. thaliana*, *O. sativa*, *M. musculus*, and *H. sapiens* tends to occur more frequently than in other organisms. For SMART domains, the ratios of the compressed size with duplication rules to the original size in *M. musculus*, and *H. sapiens* were still lower than 85%, and those in other organisms were higher than 88%. These results suggest that the same domain would be frequently utilized in higher organisms because the compression ratio was higher

Table 3: Fusion rules selected for the proteome of *C. elegans* with Pfam domains.

SDK_CAEEL	+	VRK1_CAEEL	→	**UNC22_CAEEL**
YNG2_CAEEL	+	**UNC22_CAEEL**	→	UNC89_CAEEL

Table 4: Fusion rules selected for the proteome of *A. thaliana* with Pfam domains.

BCCP2_ARATH	+	ACCC_ARATH	→	**MCCA_ARATH**
MCCA_ARATH	+	ACCD_ARATH	→	ACC1_ARATH
XB31_ARATH	+	**AKT5_ARATH**	→	AKT2_ARATH
XB34_ARATH	+	KAT1_ARATH	→	**AKT5_ARATH**

in higher organisms whose proteomes have been sufficiently investigated. It should be noted that the numbers of proteins and domains, and the compression ratio of *P. troglodytes* were largely different from those of *H. sapiens* although *P. troglodytes* is considered to be similar to *H. sapiens*. It implies that the number of known domains stored in these databases varies with organisms. In addition, the numbers of SMART domains for the organisms in this study were quite fewer than those of Pfam domains, and for our purpose, the Pfam database is more suitable. In addition to the domain databases, we examined InterPro database [32], which is an integrated database of multiple, diverse databases with reduction of redundancy. We could not find the solution by our method with InterPro database for more organisms than with Pfam database (see Table S1 at the supplementary web page). The compressed size using both of duplication and fusion was smaller than that using only duplication, which suggests the importance of fusion events in evolution of a proteome. At least, assuming the fusion events contributes to reducing the entropy of biological sequences.

Tables 3 and 4 show some fusion rules selected by our approach for the proteome of *C. elegans* and *A. thaliana* with Pfam domains, respectively, where each protein is specified by the UniProt identity (see also results on the selected rules for each organism at the supplementary web page). In *C. elegans*, protein SDK_CAEEL appeared three times, contained Pfam domains, PF00041 and PF07679, and was used to generate proteins DIG1_CAEEL, UNC22_CAEEL, and LAR_CAEEL. In particular, UNC22_CAEEL was used for another protein UNC89_CAEEL. In *A. thaliana*, fused proteins MCCA_ARATH and AKT5_ARATH were used for generating other proteins using fusion rules, respectively. These results imply that fused genes can be itera-

tively taken by gene fusion events.

4. Conclusion

We proposed a heuristic approach to compress a proteome based on protein domain compositions. The proteome was transformed to a hypergraph having hyperedges with at most three vertices on the basis of evolutionary mechanisms of gene duplication and fusion. We applied our approach to fourteen proteomes of *D. discoideum*, *E. coli*, *S. cerevisiae*, *S. pombe*, *C. elegans*, *D. melanogaster*, *A. thaliana*, *O. sativa*, *D. rerio*, *X. laevis*, *G. gallus*, *M. musculus*, *P. troglodytes*, and *H. sapiens*. As a result, the compressed size using both duplication and fusion rules was smaller than that using only duplication and the original size. Furthermore, we observed the difference of gene duplication rates between organisms. It is considered that gene duplication in *M. musculus* and *H. sapiens* tends to occur more frequently than other organisms examined in this study. In addition, we observed the phenomenon in several organisms that a fused gene was used in another gene fusion event again. For correlation between the compression ratio of each proteome and the phylogenetic tree, further analysis is needed.

However, the decrease of the compressed size was still small because our method might reduce hyperedges belonging to the minimum spanning hypertree. We need to improve our method with respect to both aspect of the efficiency and accuracy. For the efficiency, one possible way is to develop polynomial-time approximation algorithms instead of use of the integer linear programming. Another future work is to obtain the compressed size by dealing with a protein as a multiset of domains. The proteome compression using domain compositions in this study can be applied to compression of protein amino acid sequences and DNA base sequences, and the compression ratio may be improved by making use of sequences included in domains as reference.

Acknowledgment

This work was partially supported by Grants-in-Aid #22240009 and #245 00361 from MEXT, Japan. We would like to thank SuperComputer System, Institute for Chemical Research, Kyoto University, Japan.

References

[1] I. Prigogine, G. Nicolis, Self-Organization in Non-Equilibrium Systems, Wiley, 1977.

[2] S. Grumbach, F. Tahi, in: Information Processing & Management, pp. 875–886.

[3] A. Lempel, J. Ziv, IEEE Trans. Inform. Theory 22 (1976) 75–81.

[4] J. Ziv, A. Lempel, IEEE Trans. Inform. Theory 23 (1976) 337–343.

[5] E. Rivals, J. paul Delahaye, M. Dauchet, O. Delgrange, E. Rivals, J. p. Delahaye, M. Dauchet, O. Delgrange, S. Informatique, A Guaranteed Compression Scheme for Repetitive DNA Sequences, Technical Report, LIFL Lille I University, technical report IT-285, 1996.

[6] X. Chen, M. Li, B. Ma, J. Tromp, Bioinformatics 18 (2002) 1696–1698.

[7] F. Willems, Y. Shtarkov, T. Tjalkens, IEEE Trans. Inform. Theory IT-41 (1995) 653–664.

[8] T. Matsumoto, K. Sadakane, H. Imai, Genome Informatics 11 (2000) 43–52.

[9] M. Cao, T. Dix, L. Allison, C. Mears, in: Proc. Data Compression Conference (DCC '07), pp. 43–52.

[10] Z. Zhu, J. Zhou, Z. Ji, Y.-H. Shi, IEEE Transactions on Evolutionary Computation 15 (2011) 643–658.

[11] S. Kuruppu, B. Beresford-Smith, T. Conway, J. Zobel, IEEE/ACM Transactions on Computational Biology and Bioinformatics 9 (2012) 137–149.

[12] A. Cannane, H. Williams, Journal of American Society for Information Science and Technology 52 (2001) 430–437.

[13] V. Mäkinen, G. Navarro, J. Sirén, N. Välimäki, in: Proc. 13th Annual International Conference on Research in Computational Molecular Biology (RECOMB '09), pp. 121–137.

[14] S. Kuruppu, S. Puglisi, J. Zobel, in: Proc. 16th International Symposium on String Processing and Information Retrieval (SPIRE '10), pp. 201–206.

[15] A. Pinho, D. Pratas, S. Garcia, Nucleic Acids Research 40 (2012) e27.

[16] S. Deorowicz, A. Danek, S. Grabowski, Bioinformatics 29 (2013) 2572–2578.

[17] X.-M. Zhao, Y. Wang, L. Chen, K. Aihara, Proteins 72 (2008) 461–473.

[18] R. Doolittle, Annual Review of Biochemistry 64 (1995) 287–314.

[19] X.-C. Zhang, Z. Wang, X. Zhang, M. Le, J. Sun, D. Xu, J. Cheng, G. Stacey, BMC Evolutionary Biology 12 (2012) 6.

[20] J. Zhang, Trends in Ecology and Evolution 18 (2003) 292–298.

[21] S. Kummerfeld, S. Teichmann, Trends in Genetics 21 (2005) 25–30.

[22] J. C. Nacher, M. Hayashida, T. Akutsu, BioSystems 101 (2010) 127–135.

[23] A. Moore, Å.K. Björklund, D. Ekman, E. Bornberg-Bauer, A. Elofsson, Trends in Biochemical Sciences 33 (2008) 444–451.

[24] B. Brejová, D. G. Brown, T. Vinař, in: WABI 2003: Algorithms and Bioinformatics: 3rd International Workshop, volume 2812 of Lecture Notes in Bioinformatics, Springer, 2003, pp. 78–94.

[25] M. Adler, M. Mitzenmacher, in: Proceedings of the Data Compression Conference (DCC'01).

[26] R. Tarjan, Networks 7 (1977) 25–35.

[27] P. Camerini, L. Fratta, F. Maffioli, Networks 9 (1979) 309–312.

[28] H. Gabow, Z. Galil, T. Spencer, R. Tarjan, Combinatorica 6 (1986) 109–122.

[29] The UniProt Consortium, Nucleic Acids Research 40 (2012) D71–D75.

[30] M. Punta, P. C. Coggill, R. Y. Eberhardt, J. Mistry, J. Tate, C. Boursnell, N. Pang, K. Forslund, G. Ceric, J. Clements, A. Heger, L. Holm, E. L. L. Sonnhammer, S. R. Eddy, A. Bateman, R. D. Finn, Nucleic Acids Research 40 (2012) D290–D301.

[31] I. Letunic, T. Doerks, P. Bork, Nucleic Acids Research 40 (2012) D302–D305.

[32] S. Hunter, P. Jones, A. Mitchell, R. Apweiler, T. K. Attwood, A. Bateman, T. Bernard, D. Binns, P. Bork, S. Burge, E. de Castro, P. Coggill, M. Corbett, U. Das, L. Daugherty, L. Duquenne, R. D. Finn, M. Fraser, J. Gough, D. Haft, N. Hulo, D. Kahn, E. Kelly, I. Letunic, D. Lonsdale, R. Lopez, M. Madera, J. Maslen, C. McAnulla, J. McDowall, C. McMenamin, H. Mi, P. Mutowo-Muellenet, N. Mulder, D. Natale, C. Orengo, S. Pesseat, M. Punta, A. F. Quinn, C. Rivoire, A. Sangrador-Vegas, J. D. Selengut, C. J. A. Sigrist, M. Scheremetjew, J. Tate, M. Thimmajanarthanan, P. D. Thomas, C. H. Wu, C. Yeats, S.-Y. Yong, Nucleic Acids Research 40 (2011) D306–D312.