Gene Teams with Relaxed Proximity Constraint

Sun Kim
School of Informatics
Center for Genomics and Bioinformatics
Indiana University
IN 47408, USA
sunkim2@indiana.edu

Jeong-Hyeon Choi
School of Informatics
Indiana University
IN 47408, USA
jeochoi@indiana.edu

Jiong Yang
EECS
Case Western Reserve Univ.
OH 44106, USA
jiong@eecs.cwru.edu

Abstract

Functionally related genes co-evolve, probably due to the strong selection pressure in evolution. Thus we expect that they are present in multiple genomes. Physical proximity among genes, known as gene team, is a very useful concept to discover functionally related genes in multiple genomes. However, there are also many gene sets that do not preserve physical proximity.

In this paper, we generalized the gene team model, that looks for gene clusters in a physically clustered form, to multiple genome cases with relaxed constraint. We propose a novel hybrid pattern model that combines the set and the sequential pattern models. Our model searches for gene clusters with and/or without physical proximity constraint. This model is implemented and tested with 97 genomes (120 replicons). The result was analyzed to show the usefulness of our model. Especially, analysis of gene clusters that belong to B. subtilis and E. coli demonstrated that our model predicted many experimentally verified operons and functionally related clusters. Our program is fast enough to provide a sevice on the web at http://platcom.informatics.indiana.edu/platcom/. Users can select any combination of 97 genomes to predict gene teams.

1. Introduction

As the number of completely sequenced genomes is increasing rapidly due to the recent advance in genome sequencing technology, interpreting the content of genomes becomes more important. One of the most effective methods is to compare multiple genomes. Genomes can be compared on either genome sequence or protein sequence level, or both. In this paper, we assume a situation where genomes are compared on the protein level and their predicted proteins are already well classified into families of sequences. In particular, we will use the COG database [20] where predicted proteins in 66 completely sequenced genomes are classified into 4873 families. Thus the problem of our interest in this paper is to mine sets of protein families that occur in multiple genomes.

1.1. Related work: gene sets as patterns

There has been a significant amount of research on this problem. Predicting functionally correlated gene sets is a very broad topic utilizing mutliple data sources, such as gene expression data or literature, as well as sequence data. Since our goal is to generalize the gene team model based on the sequence information, our survey is limited to those utilizing sequence information. [5] and [11] studied *gene fusion events*. A gene fusion event involves two genomes, G_1 and G_2, and at least three genes, $g_{1.i} \in G_1$ and $g_{2.j}, g_{2.k} \in G_2$. A gene fusion event is an observation where $g_{1.i}$ is homologous both to $g_{2.j}$ and $g_{2.k}$, the overlap regions between $g_{1.i}$ and $g_{2.j}$ and between $g_{1.i}$ and $g_{2.k}$ with respect to $g_{1.i}$ are disjoint, and there is no observable sequence similarity between $g_{2.j}$ and $g_{2.k}$. In other words, two distinct possible domains exist in a single gene $g_{1.i}$, which is called *a composite gene*. Two genes $g_{2.j}$ and $g_{2.k}$ are called *component genes* in the fusion event. From a gene fusion event, we may conjecture that two genes $g_{2.j}$ and $g_{2.k}$ are functionally related based on the evidence of a single gene $g_{1.i}$ that has the two sequences as components. Note that such a conclusion is only possible by comparing two genomes. The work by [14] studied gene clusters to infer functional coupling. Genes are matched between two genomes using two concepts, pairs of *close* bidirectional best hits (PCBBHs) and pairs of *close* homologs (PCHs), where the term *close* means the physical proximity, say within 300 bp. Then PCBBHs and PCHs are clustered together to infer gene clusters. They successfully predicted *de novo* purine biosynthesis and glycolysis metabolic pathways. There are also interesting approaches that do not con-

sider physical proximity. [15] compared 16 genomes and successfully predicted sets of functionally linked proteins related to two structural complexes and a general amino acid metabolism by comparing only protein phylogenetic profiles. For each protein, a vector of 16 bits was created, each bit denoting the presence or absence of the protein in the corresponding genome. Then they clustered vectors to predict functionally linked protein sets. The main underlying hypothesis was that functionally linked proteins evolve in a correlated fashion and they have homologs in the same subset of organisms. Following these seminal works, there has been a significant research on methods and discovery of multiple genomes comparisons [2, 4, 22, 9]. Prediction of functionally linked gene sets are undoubtedly important for biological applications. Furthermore, these techniques can lead to development of novel techniques to solve hard computational problems in biology. For example, [12] used the three methods for predicting functionally linked genes to predict regulons, a set of genes that are regulated by a single transcription mechanism, and their regulatory motifs in 22 prokaryotic genomes using the motif-discovery program AlignACE. Note that motif (promoter site) discovery in a genome scale is far from being solved and their success was possible by restricting motif search in a small fraction of the genome, the upstream regions of predicted gene sets.

The problem of discovering sequential and set motifs (patterns) has been studied in the field of data mining. In [1], the authors investigated the problem of finding a set of items whose number of occurrences is larger than some threshold in a set of sequences or transactions. One of the most important properties of frequent set in this work is the Apriori property. The occurrences of a given set of items (e.g., a set of genes) is less than or equal to the occurrences of any of its subsets. Due to this property, the authors proposed a level-wise search algorithm. First shorter (smaller) patterns are search. If the number of occurrences of a shorter pattern does not satisfy the occurrence threshold, then it is not necessary to consider the super-patterns of the shorter pattern. We only need to consider it pattern if all its sub-patterns satisfy the occurrence threshold.

The field of sequential pattern/motif is also very active in the data mining community. Unlike the set patterns, a sequential pattern takes into account of the relative position of the elements (e.g., genes) in a pattern. Much work, e.g., asynchronous patterns [23], periodical patterns [8], etc., are proposed to discover the sequential patterns (with different constraints) whose occurrences exceed a threshold. These approaches all utilize the Apriori property. There are previous work on set pattern and distance constrained set pattern (i.e., run). However, to the best of our knowledge, our hybrid model in Section 3 is the first one to combine both type of patterns.

1.2. Motivation: Proximity Constraint or Not

If genomes are compared simply by counting genes common in two genomes, it is not very meaningful. In genomes, especially prokaryote genomes, functionally related genes are tend to be physically clustered. Thus finding out physically clustered genes is an effective way to produce functionally related gene sets. For example, [9, 2] developed an efficient algorithm to enumerate all physically clustered gene sets, termed as *gene team* in a "pair" of genomes. To produce meaningful gene teams, the choice of genome pair is very important. For example, a well known *lac* operon of four genes, i.e., gene team, *lacI*, *lacZ*, *lacY* and *lacA*, in *Escherichia coli K12* cannot be detected by comparing it to *Yersinia pestis KIM*, since these genes scatter around in *Yersinia pestis KIM*: lacZ is at base position 1995374, lacY at 2842618, and *lacI* at 3562787. On the other hand, two genomes, *Mycoplasma genitalium* and *Mycoplasma pneumoniae*, are very close so comparing them produces gene teams, each with a large number of genes. This difficulty might be alleviated if we compare a large number of genomes, say 100, simultaneously, since different gene teams may appear in different genome pairs. There are two challenges in comparing many genomes simultaneously:

- Challenge 1: It is not trivial to extend the gene team model to multiple genome cases since strictly enforcing the physical proximity constraint in "all" genomes may result in the failure of detecting gean teams. Thus we need a new model for multiple genome comparison in search of gene teams. In this paper, we propose a *hybrid pattern model* that combines the sequential pattern model (gene team model) and the set pattern model (gene team without proximity constraint).

- Challenge 2: Since we cannot enforce a constraint – whatever it is – on "all" genomes due to the abnormality in the nature, we need to consider subsets of the input genome sets. For example, suppose that we compare 100 genomes and we enforce a constraint, say based on the hybrid model, to a set of three genomes. Then we have to consider 161,700 three genome subsets (= $\binom{100}{3}$). How can we systematically enumerate these? We need an efficient algorithm for this. We had developed an algorithm based on the level-wise search techniques, which have been used to investigate the market basket data in the data mining community.

2. Parameters

To study co-occurrences of functionally related genes with or without the proximity constraint, there are four important parameters.

1. How many genes should co-occur to be biologically meaningful? We will call this *the gene set size constraint*. Let T_z be the threshold value for the gene set size constraint.

2. What is the distance between two adjacent genes if we enforce the physical proximity constraint? We will call this *the physical proximity distance constraint* or *distance constraint*. Let T_δ be the threshold value for the distance constraint.

3. How many genomes should the set of genes be physically clustered in? We will call this *distance constraint set pattern constraint* or *dset pattern constraint* in short (see Section 3). Let T_p be the threshold value for the dset pattern constraint.

4. Given a set of genes, in how many genomes should they occur as a whole, with or without the physical proximity constraint? We will count the number of genomes with and without the proximity constraint separately. We will call this *the set pattern constraint* (see Section 3). Let T_s be the threshold value for the set pattern constraint.

Although there has been a significant development on predicting functionally linked genes by comparing multiple genomes, these four issues have not been seriously studied in a combined form. This paper will explore this issue by formulating a formal problem, *the hybrid gene pattern mining problem*, in the next section we present an algorithm for the problem. We call gene teams with relaxed proximity constraint as *gene clusters* where the context is clear.

3. Model of Hybrid Patterns

In most of the previous research, there are two common models of genome patterns. One is the distance constrained sequential patterns while the other is set patterns. Before giving the formal definitions of the patterns, we will first describe some common terminology. Let Θ_G be a set of genes and Θ_F be a set of gene families. There exists a function X such that X maps a gene (in Θ_G) into a gene family (in Θ_F). In this paper, we will use COG that classifies genes in 66 genome into 4873 families.

A gene g_i in a genome G can be represented by two numbers, its transcription starting and ending positions denoted by $start(g_i)$ and $end(g_i)$ respectively. The **transcription direction**, $direction(g_i)$, of g_i is defined as $\oplus$ if $start(g_i) < end(g_i)$ and as $\ominus$ otherwise. The distance between two genes g_i and g_j, $\boldsymbol{distance}(g_i, g_j)$, is defined as $start(g_j) - end(g_i)$ (assuming $start(g_i) < start(g_j)$) if $direction(g_i) = direction(g_j)$ or as ∞ otherwise.

A **gene-family sequence** F is a sequence of gene-families, i.e., $F =< f_1, d_1, f_2, d_2, \ldots, f_m >$ where $f_i \in \Theta_F$ and $d_i = distance(g_j, g_{j+1})$ for $1 \leq i < m$ where $f_i = X(g_j)$. Thus, each genome can be transformed into a gene-family sequence via the mapping function X. A **δ-run** is the subsequence[1] of a gene-family sequence, where distances between every adjacent genes are within δ. We will omit comma and distance in family sequences for short, i.e., $f_1 f_2 \cdots f_m$ such that $f_i = X(g_i), distance(g_i, g_{i+1}) \leq \delta$ for all $1 \leq i \leq m$.

We consider the following example. A genome consists of 6 genes, g_1, g_2, g_3, g_4, g_5, and g_6, and distance between adjacent genes are 100, 200, 300, 400, and 100 base respectively. Assume that gene g_1, g_2 are mapped into family f_1, gene g_3, g_5 into family f_2, g_4 into f_3, and g_6 into f_4. The genome can be represented as the following gene-family sequence $F =< f_1, 100, f_1, 200, f_2, 300, f_3, 400, f_2, 100, f_4 >$. When $\delta = 401$, there are two runs: $r_1 = f_1 f_1 f_2 f_3$ and $r_2 = f_2 f_4$. In this paper, we are interested in discovering patterns in the gene-family sequences.

A **set pattern** of a gene-family sequence F is a set of families in F, i.e., $P_{set}(F) = \{p_1, \ldots, p_n\}$ where $p_i \in \Theta_F$ for $1 \leq i \leq n$ and $p_i \neq p_j$ for $i \neq j$. We call F supports the set pattern $\{p_1, \ldots, p_k\}$ if $\forall i \in [1, k], \exists f_j \in F$ such that $f_j = p_i$, i.e., any subset pattern of $P_{set}(F)$. We also call F supports the **d-set pattern** $P_{dset} = \{\delta, p_1, \ldots, p_k\}$, which stands for a distance constrained set pattern, if there exists a run r of F such that $P_{set}(r) - P_{dset} = \emptyset, P_{dset} - P_{set}(r) = \{\delta\}$. If a gene-family sequence F meets distance constrain, i.e., a run, then $P_{dset}(F)$ is the set pattern of families in F and $P_{dset}(F) = P_{set}(F)$ except δ. In other words, the d-set pattern can be viewed as a specialization of the set pattern. Each element in a set pattern may occur at different portion of a gene-family sequence. On the other hand, the elements in a d-set pattern should occur nearby. The distance threshold δ is used for this purpose. Let $P_{dset} = \{\delta, p_1, p_2, \ldots, p_m\}$ be a d-set pattern. Then the **twin pattern** of P_{dset} is the set pattern $\{p_1, p_2, \ldots, p_m\}$. The twin patten of a d-set pattern P_{dset} is a set pattern and it is generated by removing the distance constraint threshold δ.

For the previous example, $P_{set}(F) = \{f_1, f_2, f_3, f_4\}$, $P_{dset}(r_1) = \{401, f_1, f_2, f_3\}$, and $P_{dset}(r_2) = \{401, f_2, f_4\}$. And F supports $\{f1, f2, f4\}$, but it does not support $\{f_1, f_2, f_3, f_5\}$. Similarly, F supports d-set pattern $\{401, f_2, f_3, f_4\}$, but does not supports d-set pattern $\{401, f_1, f_3, f_4\}$ since the shortest distance between f_1 and f_3 is more than 401.

Given a support threshold k and a set of gene-family sequences, $\mathcal{D}$, a d-set pattern P is called a **frequent d-set pattern** if the support of P in $\mathcal{D}$ is at least k where the **support** of a pattern (either set pattern or d-set pattern) P in $\mathcal{D}$ is the number of sequences in D that supports P. P is called a **maximal frequent d-set pattern** if P is a frequent d-set

[1] In this paper, substring and subsequence are interchangeable.

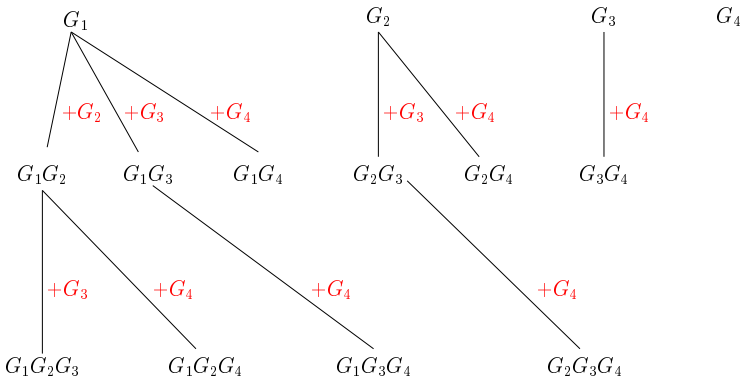


Figure 1. The levelwise enumeration tree of four genomes.

pattern and any of super-pattern of P is not a frequent sequential pattern. In the previous example, $\{401, f_2, f_3, f_4\}$ is a maximal d-set pattern while $\{401, f_2, f_3\}$ is not.

Problem Statement:
HybridGenePatternMining($T_\delta, T_p, T_z, T_s, G$):
For a set of genomes G, discover all maximal frequent d-set patterns P_{dset} for a pre-specified distance threshold T_δ such that $|P_{dset}| \geq T_z$ and the support of P_{dset} is at least T_p and the support of the twin pattern of P_{dset} is at least T_s where T_p and T_s are two support thresholds.

There are algorithms that compute clusters of COG families for a *pair* of genomes [9, 2] and the problem is termed as *COG teams*. Thus our problem is a generalized version of the COG team problem in search of hybrid patterns in unknown subsets of genomes.

4. Mining Hybrid Gene Patterns

Our algorithm recursively refines *distance constrained family sequences* or *runs* in the literature [9, 2, 14] that defined as follows. These definitions are needed for the implementation of the hybrid model.

δ-runs will be denoted using either $\boldsymbol{r}$, $\boldsymbol{s}$, or $\boldsymbol{t}$. A run match $\boldsymbol{R^k}$ is a k-tuple $(r_1, \ldots, r_k)$ such that $P_{dset}(r_1) = \cdots = P_{dset}(r_k)$, and $P_{dset}(R^k)$ is the same as $P_{dset}(r_i)$ for any $r_i \in R^k$. We simplify the notation in $\boldsymbol{R^k}$ by omitting commas whenever the context is clear; $(r_1 r_2 r_3)$ instead of (r_1, r_2, r_3). See Table 1 for examples of r and $\boldsymbol{R^k}$. $\boldsymbol{\mathcal{R}}$ denotes a set of k-tuples (run matches), $\{R^k\}$. A run match is computed iteratively using an operation, **maximal dset match**, denoted as $\wedge$. The maximal dset match of a pair of runs, $\boldsymbol{r} \wedge \boldsymbol{t}$, generates a set of 2-tuples, i.e., $\{R^2 = (r', t') \mid r'$ and t' are substrings of r and t, respectively, and $P_{dset}(r') = P_{dset}(t')\}$. While computing $r \wedge t$, r and t are split into families, $f_i \in P_{dset}(r) - P_{dset}(t)$ and $f_j \in P_{dset}(t) - P_{dset}(r)$, respectively. The maximal dset match of a run match R^k and a run s, $\boldsymbol{R^k} \wedge \boldsymbol{s}$, generates a set of $(k+1)-$tuples, i.e, $(r_1, \ldots, r_k) \wedge s = \{R^{k+1} = (r'_1, \ldots, r'_k, s') \mid s'$ and r'_i are substrings of s and r_i for $1 \leq i \leq k$, respectively, and $P_{dset}(r'_1) = \cdots = P_{dset}(r'_k) = P_{dset}(s')\}$.

The **hybrid pattern mining algorithm** is a two-step *generate-and-refine* algorithm that (1) first generates a set $\mathcal{R}$ of run matches such that for all $R \in \mathcal{R}$, R is a T_p-tuple, $|P_{dset}(R)|$ is not less than a threshold T_z, and $\forall r_i \in R$ is a T_δ-run, and then (2) refine $\mathcal{R}$ to ensure that for all $R \in \mathcal{R}$, the support of the twin patten of $P_{dset}(R)$ in the complement of a set of genomes is at least a threshold T_s.

The algorithm for distance constrained pattern mining dSetPatternMining(T_δ, T_p, T_z, $G = \{G_1, \ldots, G_n\}$) is explained using Example 1 with the description of related functions. A complete description of the algorithm is in supplementary material 1 on [18].

Example 1 There are four genomes, $\{G_1, G_2, G_3, G_4\}$. Suppose that we are looking for P_{dset} with support 3 and of size 2 or greater, i.e., $T_p = 3, T_z = 2$. Given the support constraint $T_p = 3$, we need to examine all 3 combinations of 4 genomes. To do that, we will begin levelwise enumeration of genomes. The levelwise enumeration of genomes are illustrated in Figure 1.

For the level one, we just need to compute runs, i.e., distance constrained gene-family sequences, in each genome. Let us assume runs in four genomes as shown Table 1.

We begin level 2 enumeration of genomes $\mathcal{R}^2$. There are six pairs of four genomes, $\binom{4}{2}$: (G_1, G_2), (G_1, G_3), (G_1, G_4), (G_2, G_3), (G_2, G_4), and (G_3, G_4). To distinguish which genome pairs, we will use $\mathcal{R}^2(G_i, G_j)$. The dset matches of the first pair is computed by Eq. 1.

$$\mathcal{R}^2(G_1, G_2) = (r_1 \wedge r_3) \bigcup (r_1 \wedge r_4) \bigcup (r_2 \wedge r_3) \bigcup (r_2 \wedge r_4). \quad (1)$$

Note that we do not compute $r_1 \wedge r_2$ since they are from the same genome G_1. $r_i \wedge r_j$ results in a set of 2-tuples, $\mathcal{R}^2$,

Table 1. The gene teams at each level where ? denotes a gene of unknown family.

Level	Clusters
1	$\mathcal{R}^1(G_1) = \{(r_1 = f_9f_3f_2f_4f_5f_8f_6f_7), (r_2 = f_5f_6)\}$, $\mathcal{R}^1(G_2) = \{(r_3 = f_9f_2f_3f_7f_6f_7), (r_4 = f_4f_5)\}$, $\mathcal{R}^1(G_3) = \{(r_5 = f_4f_5f_2?f_3f_9)\}$, $\mathcal{R}^1(G_4) = \{(r_6 = f_9f_6f_7)\}$.
2	$\mathcal{R}^2(G_1, G_2) = \{(r_{1.1}, r_{3.1}), (r_{1.2}, r_{3.2}), (r_{1.3}, r_4)\}$ $= \{(f_9f_3f_2, f_9f_2f_3), (f_6f_7, f_7f_6f_7), (f_4f_5, f_4f_5)\}$ $\mathcal{R}^2(G_1, G_3) = \{(r_{1.4}, r_{5.1}), (r_{1.5}, r_{5.2})\} = \{(f_9f_3f_2f_4f_5, f_4f_5f_2?f_3f_9), (f_4f_5, f_4f_5)\}$, $\mathcal{R}^2(G_1, G_4) = \{(r_{1.6}, r_{6.1})\} = \{(f_6f_7, f_6f_7)\}$, $\mathcal{R}^2(G_2, G_3) = \{(r_{3.3}, r_{5.3}), (r_4, r_{5.4})\} = \{(f_9f_2f_3, f_2?f_3f_9), (f_4f_5, f_4f_5)\}$, $\mathcal{R}^2(G_2, G_4) = \{(r_{3.4}, r_{6.2})\} = \{(f_7f_6f_7, f_6f_7)\}$, $\mathcal{R}^2(G_3, G_4) = \emptyset$.
3	$\mathcal{R}^3(G_1, G_2, G_3) = \{(f_9f_2f_3, f_9f_3f_2, f_2?f_3f_9), (f_4f_5, f_4f_5, f_4f_5)\}$ $\mathcal{R}^3(G_1, G_2, G_4) = \{(f_6f_7, f_6f_7, f_6f_7)\}$, $\mathcal{R}^3(G_1, G_3, G_4) = \emptyset$, $\mathcal{R}^3(G_2, G_3, G_4) = \emptyset$.

which will be computed by calling dSetMatch(T_z, r_i, r_j), see the algorithm in the supplementary material on [18]. Note also that dSetMatch(T_z, r_i, r_j) is recursively called in line 9 if $P_{dset}(r_i) \neq P_{dset}(r_j)$. Computing $(r_1 \wedge r_3)$ in Eq. 1 starts breaking up r_1 where families are not present in r_3, i.e, $P_{dset}(r_1) - P_{dset}(r_3) = \{f_4, f_5, f_8\}$, thus r_1 becoming $(r_{1.1} = f_9f_3f_2)$ and $(r_{1.2} = f_6f_7)$[2]. Then we need to compute $r_{1.1} \wedge r_3$ and $r_{1.2} \wedge r_3$. $r_{1.1} \wedge r_3 = \{(r_{1.1} = f_9f_3f_2, r_{3.1} = f_9f_2f_3)\}$ after r_3 is split where families are not present in $r_{1.1}$, i.e., $P_{dset}(r_3) - P_{dset}(r_{1.1}) = \{f_7, f_6, f_7\}$. Similarly, $r_{1.2} \wedge r_3 = \{(r_{1.2} = f_6f_7, r_{3.2} = f_7f_6f_7)\}$. So $(r_1 \wedge r_3)$ = $\{(r_{1.1} = f_9f_3f_2, r_{3.1} = f_9f_2f_3), (r_{1.2} = f_6f_7, r_{3.2} = f_7f_6f_7)\}$. The computation procedure for $(r_1 \wedge r_3)$ is illustrated in Figure 2. Note that the result is a set of 2-tuples. Similarly, computing $(r_1 \wedge r_4)$ in Eq. 1 starts breaking up r_1 and then $\{(r_{1.3} = f_4f_5, r_4 = f_4f_5)\}$. The dset matches of $(r_1 \wedge r_3)$ and of $(r_1 \wedge r_4)$ in Eq. 1 are discarded since the number of families in runs after splitting, f_6, is only one and $T_z = 2$ (see line 5 of dSetMatch(T_z, r_i, r_j)). Table 1 shows the dset matches of the remain pairs of genomes.

We begin level 3 enumeration of genomes $\mathcal{R}^3$ by adding a new genome to the result from the level 2 enumeration. There are 4 = $\binom{4}{3}$ combination of 3 genomes out of 4 genomes, i.e., $\mathcal{R}^3(G_1, G_2, G_3)$, $\mathcal{R}^3(G_1, G_2, G_4)$, $\mathcal{R}^3(G_1, G_3, G_4)$, and $\mathcal{R}^3(G_2, G_3, G_4)$. This can be enumerated by adding a genome G_k to $\mathcal{R}^2(G_i, G_j)$, $i < j$, only for $j < k$, by calling Compute_DMSET$_3$(T_z, R^2 ,G_k) for $R^2 \in \mathcal{R}^2$ (see lines 10 and 11 of dSetPatternMining(T_δ, T_p, T_z, $G = \{G_1, \ldots, G_n\}$)). Let us compute $\mathcal{R}^3(G_1, G_2, G_3)$ which will be computed by Eq. 1.

$$\begin{aligned}\mathcal{R}^3(G_1, G_2, G_3) &= \mathcal{R}^2(G_1, G_2) \wedge \mathcal{R}^1(G_3) \\ &= (r_{1.1}, r_{3.1}) \wedge r_5 \bigcup (r_{1.2}, r_{3.2}) \wedge r_5 \\ &\quad \bigcup (r_{1.3}, r_4) \wedge r_5. \qquad (2)\end{aligned}$$

The first term $(r_{1.1}, r_{3.1}) \wedge r_5$ in Eq. 1 is computed by calling ComputeMdsetMatch(T_z, $(r_{1.1}, r_{3.1})$, r_5). Since $P_{dset}((r_{1.1}, r_{3.1})) \neq P_{dset}(r_5)$, we need to compute $(r_{1.1} \wedge r_5) \wedge r_{3.1}$, which will be computed in two steps. Firstly, $r_{1.1} \wedge r_5$ = $\{(r_{1.1}, r_{5.5})\}$ = $\{(f_9f_2f_3, f_2?f_3f_9)\}$ by calling dSetMatch(T_z, $r_{1.1}$, r_5) (line 2). Secondly, $\{(r_{1.1}, r_{5.5})\} \wedge r_{3.1} = (r_{1.1}, r_{5.5}) \wedge r_{3.1}$ is computed by calling ComputeMdsetMatch(T_z, $(r_{1.1}, r_{5.5})$, $r_{3.1}$) (line 5). Since $P_{dset}(r_{1.1}, r_{5.5})) = P_{dset}(r_{3.1})$, the result is a set of 3-tuples $\{(r_{1.1}, r_{3.1}, r_{5.5})\}$. Note that splitting did not occur for this case. In other words, $(r_{1.1}, r_{3.1}) \wedge r_5 = (r_{1.1} \wedge r_5) \wedge r_{3.1} = \{(r_{1.1}, r_{5.5})\} \wedge r_{3.1} = \{(r_{1.1}, r_{3.1}, r_{5.5})\} = \{(f_9f_2f_3, f_9f_3f_2, f_2?f_3f_9)\}$. Similarly, we can calculate the second and the third terms in Eq. 1, i.e., $(r_{1.4}, r_4) \wedge r_5$ = $\{(r_{1.4}, r_4, r_{5.6})\}$ = $\{(f_4f_5, f_4f_5, f_4f_5)\}$, and $(r_{1.2}, r_{3.2}) \wedge r_5 = \emptyset$ since $r_{1.2} \wedge r_5 = \emptyset$. Finally, $\mathcal{R}^3(G_1, G_2, G_3)$ = $\{(f_9f_2f_3, f_9f_3f_2, f_2?f_3f_9), (f_4f_5, f_4f_5, f_4f_5)\}$. Table 1 shows the dset matches of the remain pairs of genomes. □

The algorithm for set pattern mining twinPatternMining($R^{T_p}, T_p, T_s, G = \{G_1, \ldots, G_n\}$) is explained with Example 2 with the description of related functions. A complete description of the algorithm is in supplementary material 1 on [18].

Example 2 Let us assume another genome G_5 $=< f_9, 2k, f_6, 1k, f_8, 2k, f_3, 3k, f_7 >$ in addition to four genomes $\{G_1, G_2, G_3, G_4\}$ in Example 1. There were

[2]We use notations $r_{1.x}$ to explain the computation procedure temporarily.

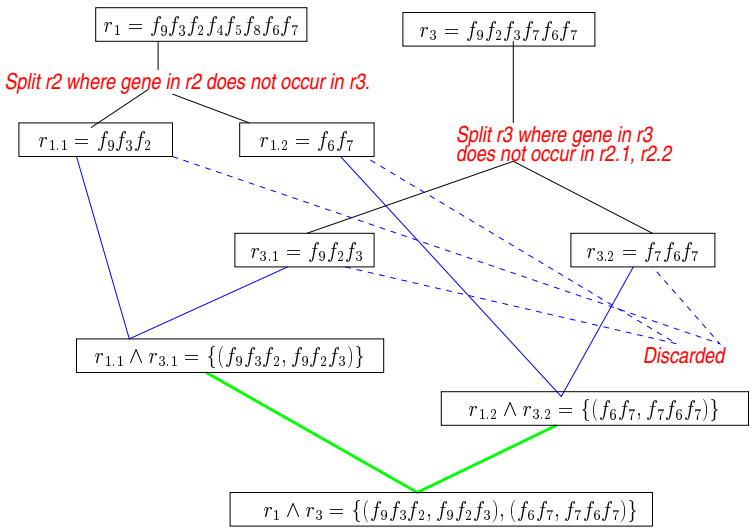


Figure 2. Illustration of how to compute $r_1 \wedge r_3$. r_1 is split into $r_{1.1}$ and $r_{1.2}$ where families in r_1 do not present in r_3, i.e., f_4, f_5 and f_8. Each of the resulting two substrings, $r_{1.1}$ and $r_{1.2}$, is tried to dset matched with r_3. To compute $r_3 \wedge r_{1.1}$, r_3 is split into $r_{3.1}$ and then the result becomes a set of 2-tuple $\{(r_{1.1}, r_{3.1})\}$. To compute $r_3 \wedge r_{1.2}$, r_3 is split into $r_{3.2}$ and then the result becomes a set of 2-tuple $\{(r_{1.2}, r_{3.2})\}$. The final result is then $\{(r_{1.1}, r_{3.1}), (r_{1.2}, r_{3.2})\}$.

three dset matches with proximity constraint $T_p = 3$ from the four genomes in Example 1, whose dset pattern are $P_{dset_1} = \{\delta, f_9, f_2, f_3\}$, $P_{dset_2} = \{\delta, f_4, f_5\}$, and $P_{dset_3} = \{\delta, f_6, f_7\}$. Mining a hybrid pattern is done by calling twinPatternMining($P_{dset_i}, T_p, T_s, G = \{G_5\}$) for each P_{dset_i}. Assume the set constraint $T_s = 1$. No family in P_{dset_2} occurs in G_5, so second dset match cannot be a hybrid pattern (by calling twinOccurrence($P_{dset_2}, \{G_5\}$) and checking line 2 of twinPatternMining()). P_{dset_3} occurs as a whole in G_5, so the set pattern $P_{set_3} = \{f_6, f_7\}$ becomes a twin pattern of P_{dset_3}. P_{set_3} and P_{dset_3} become a hybrid pattern.

For P_{dset_1}, some families in P_{dset_1} do not occur in G_5. So we need to shrink P_{dset_1} by one (calling shrinkPDSETbyOne($P_{dset_1}, T_p = 3$)). Each run of a 3-tuple $R^3 = (f_9f_2f_3, f_9f_3f_2, f_2?f_3f_9)$ will be shrunk by one and tested for their co-occurrences in G_1, G_2, and G_3. Let us try the first run $f_9f_2f_3$. Its left end is shrunk, i.e, f_2f_3, and tested for its co-occurrence in G_5 (see line 5 of shrinkPDSETbyOne()). Since $\emptyset$ is returned, the right end of the run $f_9f_2f_3$ will be shrunk by one and f_9f_2 is tested for its co-occurrence in G_5 (see line 8 of shrinkPDSETbyOne()). f_9f_2 occurs as a whole in G_5. However, it does not occur as a whole with distance constraint in G_1 and G_3, so we need to try each of the two remaining runs, $f_9f_3f_2$ and $f_2?f_3f_9$. None of the two generates sub-patterns that support G_1, G_2 and G_3. So there is no twin patterns of P_{dset_1}. Note that a sub-pattern f_3f_9 of $f_2?f_3f_9$ does support G_5 as a set pattern, but it fails to support G_2 as a distance constrained pattern. □

5. Experiment

The hybrid model is implemented in C++ using STL. It inputs `ptt` files from NCBI and produces family patterns or *conserved gene clusters*[3]. All experiments are performed on a dual Pentium IV 2.0 Ghz machine with 4GB main memory. We used 97 genomes (120 replicons) whose genes are classified with COG assignment. The 97 genome data were downloaded from NCBI. See [18] for genome list used to experiment.

5.1. Analysis of 97 genomes

Our algorithm was able to perform correlated gene set mining with 97 genomes (120 replicons) with $T_\delta = 200, T_z = 2, T_p = 2, T_s = 1$. Our algorithm exhaustively searched for gene clusters up to a support value 120, i.e., all power set enumeration of 120 replicons. However, this comprehensive analysis took only 5 hours and 40 minutes due to the *Apriori property* [1] described below.

Genome Apriori property: *If a family sequence s does not appear in a set of genomes $\mathcal{G}$, then s or its super sequence*

[3] We will use *gene clusters* and *family patterns* interchangeably.

cannot appear in $\mathcal{G}'$ where $\mathcal{G} \subset \mathcal{G}'$.

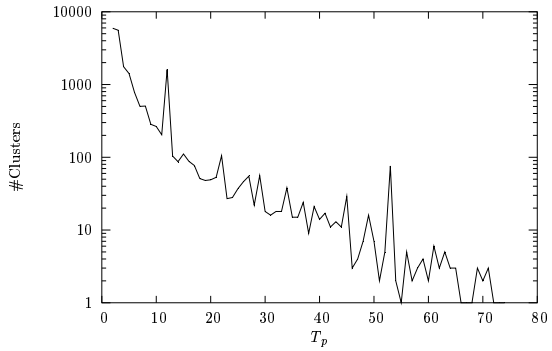
There were 20345 gene clusters with two families or more at different support levels (T_p and T_s). There are 36 clusters that are present in 60 replicons or more, $T_p \geq 60$. Among these are 30 ribosomal protein clusters, 5 RNA polymerase subunit clusters, and 1 transporter cluster. There are 5578 clusters that are present in 3 replicons or more. The largest clusters with 20 or more families, $T_z \geq 20$, are all ribosomal protein clusters (40 clusters). The data is summarized in Figure 3 and available on [18].

5.2. Parameter settings for T_p and T_s

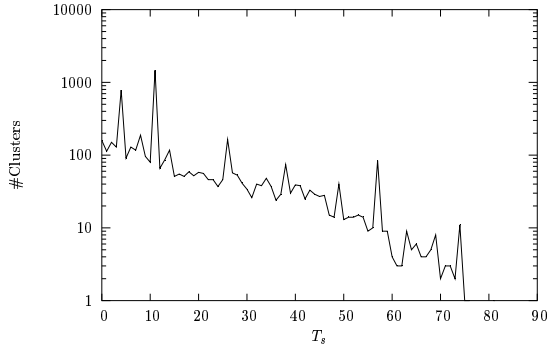
The main point of this paper is that the hybrid model is effective in mining correlated gene sets. Suppose that we mine gene clusters in n genomes. Then the question is in how many genomes gene clusters should occur with or without the proximity contraint, *i.e.,* T_s and T_p. To explore this question, we used operons in *B. subtilis* and *Escherichia coli K12* [10, 13] since they are well studied organisms. We selected 12 genomes including *B. subtilis* in *Gramplus* group and 10 genomes including *Escherichia coli K12* in *Gamma* group, and then mined gene sets in 5 genomes. To explore the effect of the proximity contraint, various combinations of T_p and T_s were used. Note that $T_p + T_s = 5$. Note also that the gene sets for $T_p = i$ and $T_s = k-i$ contain those for $T_p = j$ and $T_s = k-j$ if $T_p + T_s = k$ and $i \leq j$, *i.e.,* $HybridGenePatternMining(T_\delta, T_p{=}3, T_z, T_s{=}2, G) \subseteq HybridGenePatternMining(T_\delta, T_p{=}2, T_z, T_s = 3, G)$.

Table 2 (a) and (b) show the number of operons detected by the gene sets, the number of genes occurring in operons, and the number of genes occurring outside operons in *B. subtilis* and *Escherichia coli K12*, respectively. For the *B. subtilis* data, the number of detected operons decreased for higher T_p values. This is intuitive since the proximity contraint became enforced stronger for higher T_p values. However, the number of detected operons were relatively stable for the *Escherichia coli K12* data up to $T_p = 4$. This is beacuse the *Escherichia coli K12* data contained four different strains of *Escherichia coli K12* while all genomes were diferent for the the *B. subtilis* data. Thus it shows that the characteristics of the input data is important for mining correlated gene sets. Note that the $T_p = 1$ constraint requires gene clusters only in one genome with the proximity constraint, which is probably too weak to produce meaningful gene clusters. However, the $T_p = 1$ constraint together with $T_s = 4$ detected 158 genes in 40 experimentally verified operons in *B. subtilis* with only 80 genes outside the operons.[4] Thus our hybrid model may detect gene clusters even

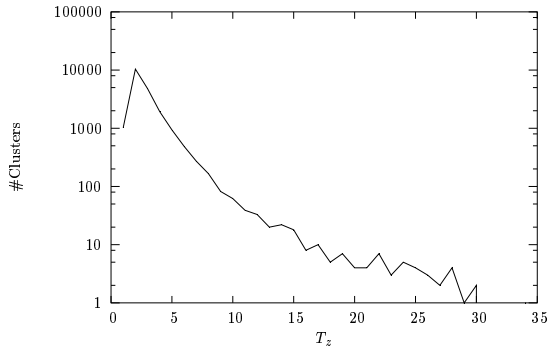
[4]We cannot rule out the possibility of some of the 80 genes being in true operons since there might be unclassfied operons.



(a)



(b)



(c)

Figure 3. Summary of gene clusters in 120 replicons. The dset support values (T_p) vs. the number of clusters (plot a), the set support values (T_s) for a fixed $T_p = 3$ vs. the number of clusters (plot b), and and the size of clusters (T_z) vs. the number of clusters (plot c).

Table 2. The number of operons detected by predicted gene sets

T_p	T_s	#Operons detected	#Genes in operons	#Genes outside operons
5	0	18	63	3
4	1	22	85	13
3	2	25	110	20
2	3	33	134	45
1	4	40	158	80

(a) *B. subtilis* among *Gramplus*

T_p	T_s	#Operons detected	#Genes in operons	#Genes outside operons
5	0	29	104	37
4	1	40	141	58
3	2	43	153	78
2	3	45	160	83
1	4	47	167	92

(b) *Escherichia coli K12* among *Gamma*

Table 3. The number of genes in clusters, the number of clusters, and the coverage ratio of genes in clusters to the total number of genes in *B. subtilis*.

T_p with $T_s = 1$	T_s with $T_p = 1$	# genes	# clusters	coverage
2	5	2679	557	0.65
2	10	2465	531	0.60
2	15	2353	510	0.57
3	5	2495	561	0.60
3	10	2048	529	0.49
3	15	2121	497	0.52
4	5	2394	570	0.58
4	10	2048	528	0.50
4	15	1881	490	0.46
5	5	2322	567	0.56
5	10	1941	515	0.47
5	15	1746	470	0.42

when relatively distant genomes are available.

5.3. Post-processing for Operon Prediction

A simple post-processing of predicted gene clusters produced quite accurate operon predictions: many gene clusters matched in entirety or included experimentally verified operons. Note that these predictions were made without performing any testing for regulation promoters or terminators, yet matched many experimentally verified operons. Table 3 summarizes the number of genes, the number of clusters, and the coverage ratio of genes in clusters to the total number of genes in *B. subtilis*. The predicted gene clusters cover 42% to 65% of the entire protein coding genes in *B. subtilis* for different support values.

Operons detected by gene clusters with ($T_p = 4, T_s = 1$) and ($T_p = 1, T_s = 15$) in Table 4. Among 48 experimentally verified operons with gene clusters, 26 operons are with their exact boundaries. Many extra genes in the clusters, those outside known operons, are indeed functionally related. For example, a predicted gene cluster contains *glpD glpF glpK glpP* where *glpF glpK* are an operon known as *glpFK*. Two surrounding proteins, *glpD* and *glpP* are leader and antiterminator proteins [7]. Another predicted gene cluster contains *dacF spoIIAA spoIIAB sigF spoVAA spoVAB spoVAC spoVAD spoVAE spoVAF* where *spoVAA spoVAB spoVAC spoVAD spoVAE* are known as an operon. However, it was shown that *dacF* and *spoIIA* operons are autoregulated [16]. In addition, there were predicted gene clusters that did not contain any of known operons in [13]. Note that functionally related gene cluster is a more general concept than operon, thus many gene clusters that do not contain operons can be functionally related. We were not able to verify these clusters since many of them are not characterized. However, there are clusters that we were able to verify in the literature. For example, a gene cluster that did not contain known operons had six genes, *ssuA ssuB ssuC ssuD ygaN yhzA*. Among these, the first five genes (*ssuA ssuB ssuC ssuD ygaN*) are known to have operon-like structure that utilizes sulfur from aliphatic sulfonate [21]

This method can be used for predicting putative operon of any bacterial genome and is available on [18].

5.4. Comparison with results in the literature

We compared pairwise gene cluster prediction with *Escherichia coli K12* and *B. subtilis* in [9]. Our pairwise prediction algorithm is simply to split recursively two runs r and s (see dSetMatch(T_z, r, s) in supplementary material on the web [18]). This simple algorithm can produce the result consistent with those in [9] without the time and space problem; it only took 3 seconds with 2.5MB memory usage.

We also compared gene clusters from our algorithm with gene order conservation study in [19]. Our gene clusters with respect to *Escherichia coli K12* were quite consistent with those in [19]; in a total of 18 groups, predicted gene sets exactly matched group 1, 3, 6, 9, 10, 12, 13, 14, 15, and 16 and partially matched group 4, 5, 7, 8, and 11. Our method were able to detect the longest groups of 14 and 28 genes.

yjbA **appC appB appA appF appD** yjaZ fabF fabHA
purD purH purN purM purF purL purQ purS purC purB purK purE yebG yebE yebD yebC
ndk hepT menH hepS mtrB mtrA hbs spoIVA yphF yphE gpsA yphC seaA yphA
pyrR pyrP pyrB pyrC pyrAA pyrAB pyrK pyrD pyrF pyrE
ypkP **dfrA thyB** ypjQ ypjP
comC folC valS ysxE spoVID **hemL hemB hemD hemC hemX hemA**
comGA comGB comGC comGD comGE comGF comGG yqzE
pstS pstC pstA pstBA pstBB
acuA acuB acuC
qcrC qcrB qcrA ypiF ypiB ypiA aroE tyrA hisC **trpA trpB trpF trpC trpD trpE**
spoVAF **spoVAE spoVAD spoVAC spoVAB spoVAA** sigF spoIIAB spoIIAA dacF
acpS ydcC alr ydcD ydcE **rsbR rsbS rsbT rsbU rsbV rsbW sigB rsbX** ydcF ydcG
ywtB ywtA ywsC **rbsR rbsK rbsD rbsA rbsC rbsB**
minD minC mreD mreC mreB radC maf spoIIB
lonA lonB clpX tig ysoA **leuD leuC leuB leuA ilvC ilvH ilvB**
atpC atpD atpG atpA atpH atpF atpE atpB atpI

argC argJ argB argD carA carB argF yjzC
gcvT gcvPA gcvPB
glnA glnR ynbB ynbA
kapB kinB patB
pbpE racX yveF yveG
ureA ureB ureC
glgP glgA glgD glgC glgB
yfkQ **treP treA treR**
qoxA qoxB qoxC qoxD
glnH glnM glnP glnQ
nrgA nrgB ywoA
adaA adaB
motA motB
phoP phoR
ftsA ftsZ
tagG tagH

oppA oppB oppC oppD oppF yjbB yjbC yjbD
gbsA gbsB yuaD
feuA feuB feuC
sdhC sdhA sdhB ysmA gerE
mutL mutS cotE ymcA ymcB
cgeC cgeD cgeE
glpD **glpF glpK** glpP
opuBA opuBB opuBC opuBD
ecsA ecsB ecsC
hemE hemH hemY
dnaG sigA
spoIVFA spoIVFB
glpQ glpT
sacX sacY
pbuX xpt
alsD alsS

Table 4. Operons detected by gene clusters with ($T_p = 4, T_s = 1$) and ($T_p = 1, T_s = 15$). Genes in bold font are those in known operons. Among 48 experimentally verified operons with gene clusters, 26 operons are with their excat boundaries. Many extra genes, those outside known operons, in other clusters are indeed functionally related (see the main text).

5.5. Phylogenetic relationship using conserved gene clusters

Traditionally, species phylogenies have been acquired by comparisons of a specific gene, *i.e., 16S rRNA*. However, they are rarely consistent with each other, due to horizontal gene transfer and highly variable rates of evolution. [17] have developed a creative distance-based phylogeny constructed on the basis of gene content of 13 completely sequence genomes. The evolutionary *distance* between two genomes is defined as (1 − similarity), and the *similarity* is the fraction of the number of their common genes to the number of genes in smallest genome. The common genes between two genomes are considered only when the score of two genes is above cutoff value (say, $E = 0.01$) according to Smith-Waterman comparison.

We show that common gene clusters predicted by our method can produce accurate phylogenetic relationship among different organisms. Note that our method do not have to align sequences using the pairwise and multiple sequence alignment methods. We collected 13 genomes used in [17]: *H.influenzae*, *M.genitalium*, *Synechocystis*, *M.jannaschii*, *E.coli*, *M.thermoautotrophicum*, *H.pylori*, *A.fulgidus*, *B.subtilis*, *B.burgdorferi*, *S.cerevisiae*, *A.aeolicus*, and *HP.horikoshii* (Figure 4). Gene teams were computed with $T_p = 2$ and $T_s = 2$. The evolutionary distance between two genomes is defined in the same way as in [17]. The only difference between our approach and the one by [17] is how to count common genes between genomes. Common genes in our approach is those in predicted gene clusters.

Figure 4 compares three phylogenetic trees generated using *16S rRNA*, common genes, and common gene clusters. Plot (a) and (b) come from [17], and plot (c) is constructed using the neighborhood joining method in `phylip` package [6] and visualized using `PhyloDRAW` [3]. It is interesting that all trees are the same except *Synechocystis* and *S.cerevisiae*. As shown in the figure, predicted gene clusters can be used to produce an accurate phylogenetic tree without aligning sequences.

6. Conclusion

In this paper, we proposed a novel hybrid pattern mining model and showed that this model can generate biologically meaningful gene clusters in 97 genomes (120 replicons) based on COG classification. The hybrid pattern mining model is new in that it combines two widely used pattern models, sequential pattern model and set pattern model.

The hybrid model was successful in predicting 20,345 possibly functionally related gene clusters on a wide range of genome combinations in 5 hours and 40 minutes on a Pentium 2.0 Ghz machine. Analysis of gene clusters in *B.*

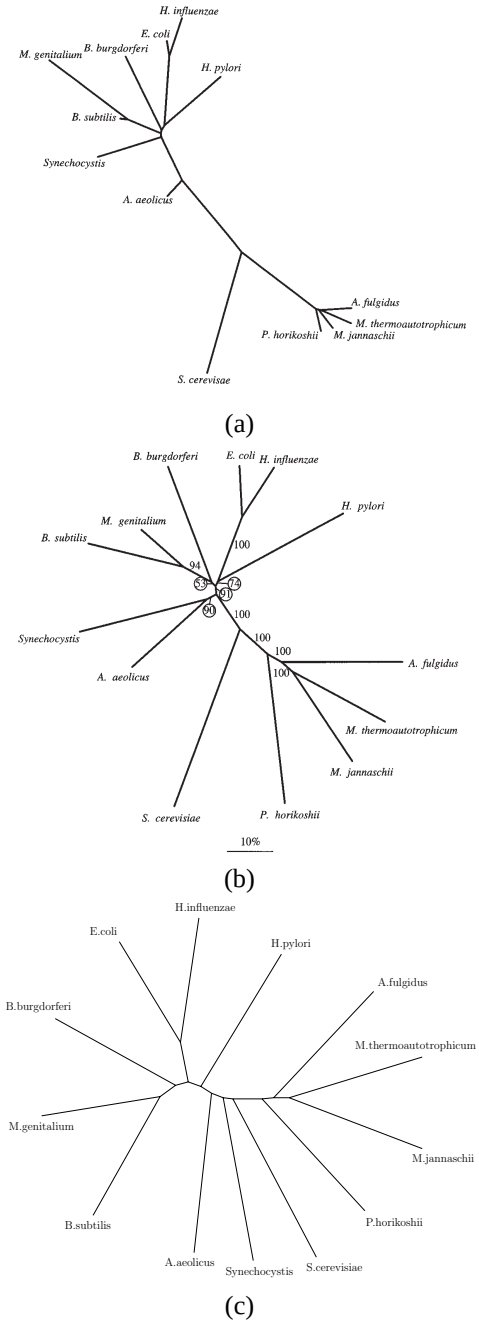


Figure 4. Comparison of phylogenetic trees for 13 genomes used in [17]. Plot (a) is generated using *16S rRNA*, plot (b) using common genes [17], and plot (c) using common genes in clusters predicted by our method. See main text for more detail. All plots are the same except *Synechocystis* and *S.cerevisiae*. The predicted gene clusters using our method can be used to produce an accurate phylogenetic tree without aligning sequences.

subtilis were quite close to experimentally verified operons and functionally related gene sets in the literature. It is also shown that our prediction result is consistent with those in the literature [19, 9].

The current COG database do not classify multi-domain sequences, thus fusion event was not utilized. Our model can easily accommodate the multi-domain sequences by generating gene or family sequences by producing two adjacent families with the distance of 0 base. Once the multidomain data is available, we will report the analysis results. In addition to the COG assignment, there are other family classification schemes, e.g., GO-term- or domain-based family classifications. Our hybrid model and algorithm do not depend on a particular classification scheme, thus users can easily utilize other family classification schemes, which may produce different results from those reported in this paper. We plan to explore other family classification schemes. In addition to the family classification scheme issue, genomes can be compared at several different levels. In particular, we may need to consider one single family string of multiple replicons in a single genome, rather than a separate string for each replicon. Users can easily convert multiple family strings into one either by choosing only one of the multiple replicons or by concatenating all replicons into a single family string, so it will not affect the design of our model and algorithm. However, it will certainly affect the parameter settings, especially for T_s and T_p. This paper is mainly to present a new hybrid model and an algorithm for the model, not to study the behaviour of gene team occurrences with or without proximity constraint in terms of biological implications. We plan to perform an in-depth study of biological implication on gene team occurrences with or without proximity constraint.

Aside from the issues that we discussed above, there are several interesting questions that remain to be explored. Our dset pattern model, or gene team model in the literature [9, 2], does not allow an arbitrary distance between genes. For example, suppose that two genes, g_i and g_j, are separated by 5000bp. Then there may or may not be other genes between g_i and g_j. In this case, it is not clear how to interpret this gene sequence whether it is a dset pattern or a set pattern. Another question is how we can utilize the hybrid pattern model for classifying genes of unknown function or for correcting annotations. There is a growing evidence of the importance of gene context analysis; for example, a recently developed graphical tool [4] demonstrated that annotation errors for *moeB*, *tenI* and *goxB* genes in several genomes by generating gene context around COG0352. The hybrid pattern model provide such context of genes, which will be more accurate, if correctly used, than pairwise relationships generated by FASTA or BLAST.

Acknowledgment

This work is partially supported by INGEN (Indiana Genomics Initiatives) and NSF DBI-0237901. Thanks also to the two anonymous reviewers for their comments.

References

[1] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In J. B. Bocca, M. Jarke, and C. Zaniolo, editors, *Proc. 20th Int. Conf. Very Large Data Bases, VLDB*, pages 487–499. Morgan Kaufmann, 12–15 1994.

[2] A. Bergeron, S. Corteel, and M. Raffinot. The algorithmic of gene teams. In R. Guigó and D. Gusfield, editors, *Proceedings of the 2nd Workshop in Bioinformatics*, number 2452 in Lecture Notes in Computer Science, pages 464–476, Rome, Italy, 2002. Springer-Verlag, Berlin.

[3] J.-H. Choi, H.-Y. Jung, H.-S. Kim, and H.-G. Cho. PhyloDRAW: A phylogenetic tree drawing system. *Bioinformatics*, 16(11):1056–1058, 2000.

[4] R. Ciria, C. Abreu-Goodger, E. Morett, and E. Merino. GeConT: gene context analysis. *Bioinformatics*, 20(14):2307–2308, 2004.

[5] A. J. Enright, I. Iliopoulos, N. C. Kyrpides, and C. A. Ouzounis. Protein interaction maps for complete genomes based on gene fusion events. *Nature*, 402(6757):86–90, Nov 1999.

[6] J. Felsenstein. *PHYLIP: Phylogeny Inference Package. Version 3.5.* University of Washington, Seattle, WA, 1993.

[7] E. Glatz, A. Farewell, and B. Rutberg. The *Bacillus subtilis glpD* leader and antiterminator protein *glpP* provide a target for glucose repression in *Escherichia coli*. *FEMS Microbiology Letters*, 162(1):93–96, May 1998.

[8] J. Han, G. Dong, and Y. Yin. Efficient mining of partial periodic patterns in time series database. In *Fifteenth International Conference on Data Engineering*, pages 106–115, Sydney, Australia, 1999. IEEE Computer Society.

[9] X. He and M. H. Goldwasser. Identifying conserved gene clusters in the presence of orthologous groups. In *Proceedings of the 8th Annual International Conference on Computational Molecular Biology (RECOMB)*, pages 272–280. ACM Press, 2004.

[10] T. Itoh, K. Takemoto, H. Mori, and T. Gojobori. Evolutionary instability of operon structures disclosed by sequence comparisons of complete microbial genomes. *Mol. Biol. Evol.*, 16(3):332–346, 1999.

[11] E. M. Marcotte, M. Pellegrini, H.-L. Ng, D. W. Rice, T. O. Yeates, and D. Eisenberg. Detecting protein function and protein-protein interactions from genome sequences. *Science*, 285(5428):751–753, 1999.

[12] A. M. McGuire and G. M. Church. Predicting regulons and their *cis*-regulatory motifs by comparative genomics. *Nucleic Acids Res.*, 28(22):4523–4530, 2000.

[13] Operon Data Library, 1999.

[14] R. Overbeek, M. Fonstein, M. D'Souza, G. D. Pusch, and N. Maltsev. The use of gene clusters to infer functional coupling. *Proc. Natl. Acad. Sci. U.S.A.*, 96(6):2896–2901, 1999.

[15] M. Pellegrini, E. M. Marcotte, M. J. Thompson, D. Eisenberg, and T. O. Yeates. Assigning protein functions by comparative genome analysis: Protein phylogenetic profiles. *Proc. Natl. Acad. Sci. U.S.A.*, 96(8):4285–4288, 1999.

[16] R. Schuch and P. J. Piggot. The *dacF-spoIIA* operon of *Bacillus subtilis*, encoding sigma f, is autoregulated. *J Bacteriol.*, 176(13):4104–4110, Jul 1994.

[17] B. Snel, P. Bork, and M. A. Huynen. Genome phylogeny based on gene content. *Nat. Genet.*, 21(1):66–67, 1999.

[18] Supplementary material, 2005.

[19] J. Tamames. Evolution of gene order conservation in prokaryotes. *Genome Biology*, 2(6):research0020.1–research0020.11, 2001.

[20] R. Tatusov, N. Fedorova, J. Jackson, A. Jacobs, B. Kiryutin, E. Koonin, D. Krylov, R. Mazumder, S. Mekhedov, A. Nikolskaya, B. S. Rao, S. Smirnov, A. Sverdlov, S. Vasudevan, Y. Wolf, J. Yin, and D. Natale. The COG database: An updated version includes eukaryotes. *BMC Bioinformatics*, 4(1):41, 2003.

[21] J. van der Ploeg, N. Cummings, T. Leisinger, and I. Connerton. *Bacillus subtilis* genes for the utilization of sulfur from aliphatic sulfonates. *Microbiology*, 144(9):2555–2561, 1998.

[22] I. Yanai, A. Derti, and C. DeLisi. Genes linked by fusion events are generally of the same functional category: A systematic analysis of 30 microbial genomes. *Proc. Natl. Acad. Sci. U.S.A.*, 98(14):7940–7945, 2001.

[23] J. Yang, W. Wang, and P. S. Yu. Mining asynchronous periodic patterns in time series data. In *Knowledge Discovery and Data Mining*, pages 275–0279, 2000.