Identifying Conserved Gene Clusters in the Presence of Homology Families

Xin He
Department of Computer Science
Univ. of Illinois at Urbana-Champaign
Urbana, IL 61801
USA

Michael H. Goldwasser
Dept. of Math. and Computer Science
Saint Louis University
220 North Grand Blvd
St. Louis, MO 63103-2007
USA

Keywords: cluster of orthologous genes, comparative genomics, conserved gene cluster, gene team, homology

Abstract

The study of conserved gene clusters is important for understanding the forces behind genome organization and evolution, as well as the function of individual genes or gene groups. In this paper, we present a new model and algorithm for identifying conserved gene clusters from pairwise genome comparison. This generalizes a recent model called "gene teams". A gene team is a set of genes that appear homologously in two or more species, possibly in a different order yet with the distance of adjacent genes in the team for each chromosome always no more than a certain threshold. We remove the constraint in the original model that each gene must have a unique occurrence in each chromosome, and thus allow the analysis on complex prokaryotic or eukaryotic genomes with extensive paralogs. Our algorithm analyzes a pair of chromosomes in $O(mn)$ time and uses $O(m + n)$ space, where m and n are the number of genes in the respective chromosomes. We demonstrate the utility of our methods by studying two bacterial genomes, *E. coli* K-12 and *B. subtilis*. Many of the teams identified by our algorithm correlate with documented *E. coli* operons, while several others match *predicted* operons, previously suggested by computational techniques. Our implementation and data are publicly available at `http://euler.slu.edu/~goldwasser/homologyteams/`.

1 Introduction

A fundamental question in genomics involves the relationship between a gene's position in a chromosome and its function. The current evidence suggests that genes are not randomly distributed in the chromosomes and that genes which are physically close to each other, even if not adjacent, tend to represent groups of genes with a functional relationship. In prokaryotic genomes, functionally-related genes are often organized into "operons," clusters of genes that are regulated as one transcriptional unit. If a set of genes remain as relative neighbors in multiple species, it may be that evolutionary constraints maintain this arrangement and thus these genes are more likely to represent a functionally-significant group. Such a gene group is often termed a *conserved gene cluster* in the literature. In this paper, we consider two distinct issues in comparative genomics. Our primary result is the development of an algorithm for *identifying* conserved gene clusters. A secondary issue is the subsequent *evaluation* of the significance of identified clusters.

In a recent series of papers, Béal et al. (Béal et al., 2004; Bergeron et al., 2002) introduce a concept denoted as a *gene team*. A gene team is a set of genes that appear homologously in two or more species, possibly in a different order yet with the distance between adjacent genes in the team for each chromosome always no more than a certain threshold. They provide an $O(n \log^2 n)$-time, $O(n)$-space algorithm which identifies the gene teams common to a pair of chromosomes comprised of n common genes. However, their model includes a restriction that each gene has at most one occurrence in each chromosome. This severely limits the impact of the model, as many genomes contain a large number of multi-gene families known as homologs, which descended from a common ancestor via a combination of duplication and speciation events. This is especially so in higher organisms. In this paper, we extend the notion of "gene teams" to allow more general consideration of such homologs. We refer to these generalizations as "homology teams," for reasons explained in Section 3. We provide a divide-and-conquer algorithm to find such homology teams for *pairwise* chromosome comparison. The algorithm has $O(mn)$ running time and $O(m+n)$ space requirement, where m and n are the number of genes in the two chromosome for which there exists at least one homologous gene in the opposite chromosome.

When a gene cluster has been identified, a subsequent issue is to evaluate its possible biological significance. It has been shown that gene clusters whose order are conserved have a strong tendency to encode proteins that have physical interaction (Dandekar et al., 1998). Recent studies also indicate that in some eukaryotic genomes, neighboring genes tend to have similar expression patterns (Spellman and Rubin, 2002). The functional coupling of genes in conserved clusters has been demonstrated computationally in studies involving multiple prokaryotic genomes (Fujibuchi et al., 2000; Overbeek et al., 1999). However, the evaluation of a cluster's significance must likely be based upon greater knowledge than solely the combinatorial structure of the two compared genomes. A gene cluster may be conserved simply because the two genomes being compared have not sufficiently diverged from their common ancestor. A non-functional association might presumably be broken only after continued shuffling of the genomes. Conversely, it may be that a set of genes were not located close to each other in an ancestor genome, yet come together during evolution due to coincidence. In this case, an identified gene cluster should not truly be qualified as conserved.

Though the problem of evaluating biological significance of clusters is beyond our primary scope, we wish to demonstrate the practical use of our algorithmic techniques in identifying potential conserved gene clusters. We describe preliminary experiments based upon the comparison of two bacterial genomes, *E. coli* K-12 and *B. subtilis*. Because this pair is well studied, we can compare our identified teams to known or predicted operons as benchmarks. Furthermore, these two genomes are sufficiently divergent to assume that genes would no longer be in proximity to each other, if not for the forces of selection. This criteria has been used in a previous study by Fujibuchi et al. (Fujibuchi et al., 2000), who used phylogenetic distance of 20%, measured by the percent difference in small rRNA sequences, as a cutoff value for any significant conserved clusters. In Appendix A we develop a statistical test to assess the possibility that an identified gene cluster is due to chance.

The remainder of the paper is structured as follows. We discuss several closely related research efforts in Section 2. Our own concept of a homology team is formalized in Section 3. We present the algorithm for identifying homology teams of two chromosomes in Section 4, together with an analysis of running time and space usage. In Section 5, we demonstrate the practical use of this algorithm by performing analysis on genomic data, successfully reconstructing known and predicted operons from pairwise genome comparison.

2 Related Work

Various computational methods have been proposed to identify conserved gene clusters by comparison of multiple genomes, both experimentally and formally. Models of a conserved gene cluster vary across several axes: whether the genes in a cluster are required to be immediate neighbors on a chromosome; whether gene order is allowed to vary among chromosomes; whether paralogs are allowed on a single chromosome, and thus a many-to-many homologous relationship among chromosomes; whether comparison is pairwise between two genomes, or generally between three or more genomes.

If disallowing paralogs and requiring immediate neighboring relationships, each chromosome can be modelled as a permutation over the set of genes. The problem of locating gene clusters is then related to a previously studied combinatorial problem, that of finding common intervals in two or more permutations. Uno and Yagiura present an $O(n + K)$-time, $O(n)$-space algorithm to find all K common intervals between two permutations of n items, where $K \leq \binom{n}{2}$ (Uno and Tagiura, 2000). Heber and Stoye generalize this algorithm to an $O(cn + K)$-time, $O(cn)$-space algorithm to find all K common intervals between $c \geq 2$ permutations (Heber and Stoye, 2001). Recently, Didier extends this approach to include paralogs by considering a more general sequence definition than a strict permutation, providing an $O(n^2 \log n)$-time and linear-space algorithm to enumerate all such intervals (Didier, 2003). Schmidt and Stoye show how to solve this problem in $O(n^2)$-time and linear-space (Schmidt and Stoye, 2004). Our algorithm achieves these results as a simple special case where no gap is allowed within a homology team.

A drawback of each of the above approaches is that a single misplaced gene disqualifies the commonality between two otherwise similar intervals. To remedy this drawback, gene clusters can be defined so long as the gaps between adjacent genes in the cluster are less than a given threshold. This is the approach taken by Overbeek et al. (Overbeek et al., 1999), by Béal et al.(Béal et al., 2004), as well as by us. We note that the work of Overbeek et al. considers only pairwise comparison of chromosomes, as does ours. Béal et al. do observe that their algorithm can be generalized to find common teams among a set of c chromosomes, resulting in an $O(cn \log^2 n)$-time, $O(cn)$-space algorithm. However, the insistence on one-to-one homologous relationships severely limits the utility of that generalization. For pairwise comparison, a common technique to ensure one-to-one homologous relationships on experimental data is to rely on the bidirectional best hit method, as in (Luc et al., 2003; Overbeek et al., 1999). The generalization of this criteria to three or more chromosomes likely precludes the inclusion of many likely homologs.

3 Model and Definitions

We begin with a brief review of terminology associated with phylogenetic homology. *Homologs* are genes that have descended, usually with divergence, from a common ancestral gene. The homology relationship is symmetric and transitive and thus a gene and all its homologs form (disjoint) equivalence classes we will term *homology families*. Often, a homology between a pair of genes is further categorized based upon the type of event which caused the earliest divergence. Fitch introduces the terms *ortholog* and *paralog* in this regard (Fitch, 1970). A pair of homologs whose original divergence followed a speciation are termed *orthologs*; a pair of homologs whose original divergence followed a gene duplication are termed *paralogs*. Based upon these definitions, a pair of orthologs must always lie in distinct genomes. Paralogs are often viewed as duplicated genes which lie within a single genome, though it is possible for a gene to have a paralog in another genome, e.g., if a speciation occurs following an earlier gene duplication (Fitch, 2000). Without

gene duplications, the homologous relationship between a pair of genomes would be one-to-one; yet because of gene duplications there commonly exist many-to-one or many-to-many homologous relationships between a pair of genomes. For the remainder of our paper, we assume many-to-many homologous relationships may exist. We then have no further need to distinguish between orthologous and paralogous relationships.

We note that the (phylogenetic) homology relationship is defined entirely by common ancestry, a property which is unlikely to be verifiable, rather inferred based upon evidence. We therefore distinguish between this (phylogenetic) homology and other notions such as *structural homology* or *functional homology* which strive to classify groups of genes based on some form of similarity (Fitch, 2000). Classification by various similarity measures often suffer from a lack of symmetry, transitivity or completeness in identifying homology. Tatusov et al. introduce the widely-used notion of a *Cluster of Orthologous Groups (COG)* (Tatusov et al., 1997; Tatusov et al., 2001). Each COG consists of a group of genes which are presumed to be homologous based upon sequence similarity across multiple genomes. In essence, a COG is an experimental approximation to a homology family. In the experiments described in Section 5, we rely on data from the COG database (National Center for Biotechnology Information, 2004) in this regard. Yet, the algorithm described in Section 4 is independent of the method used to infer homology.

Definition 1 *A* chromosome *is a pair $C = (\Sigma, X)$ where Σ is a set of homology families and where X is an ordered sequence of genes. Each gene of X is denoted as a pair (p, f) with $p \in \mathbb{R}$ representing the physical position of the gene in the chromosome and $f \in \Sigma$ the homology family to which the gene belongs. We will refer to Σ as the* alphabet *for C*

Definition 2 *Given a chromosome containing two genes $g_i = (p_i, f_i)$ and $g_j = (p_j, f_j)$, the distance between g_i and g_j is defined as $\Delta(g_i, g_j) = |p_i - p_j|$.*

The sequence of genes is ordered such that the position values are monotonically increasing. However, the positions need not be consecutive nor integral. In practice, this value represents a physical measure of the gene position, such as base pair distance. Since we are interested in performing comparative genomics on a pair of chromosomes, genes which do not have homologs in the opposite chromosome should be omitted in the representation, thereby reducing the computational costs for our algorithm.

Example 3 *We consider a chromosome C over alphabet $\Sigma = \{a, b, c, d, e\}$, with genes $(1, c), (4, e), (5, d), (6, a), (8, b), (10, e), (11, d)$. Throughout this paper, examples with integral positions are illustrated using a textual representation, in this case $C =$* `<c**eda*b*ed>`*. We use asterisks to represent gaps between genes of the relevant genes. In a pairwise chromosome comparison, such gaps may represent genes which do not have homologs in the second chromosome.*

Definition 4 *Given chromosome $C = (\Sigma, X)$ and any $X' \subseteq X$, we use the notation $\Sigma(X') \subseteq \Sigma$ to denote the subset of families occurring in X'.*

Definition 5 *Given chromosome $C = (\Sigma, X)$, we denote as a* subsequence *any pair $(\Sigma(X'), X')$ where $X' \subseteq X$.*

For example, $X' = \{(4, e), (6, a), (10, e), (11, d)\}$, forms a subsequence of chromosome C given in Example 3.

Definition 6 *Given chromosome $C = (\Sigma, X)$ and subsequence $C' = (\Sigma', X')$, we say that C' is a* subchromosome *if it is also the case that X' is a* contiguous *subset of X.*

For example, $X' = \{(5, d), (6, a), (8, b), (10, e)\}$, when paired with $\Sigma(X') = \{a, b, d, e\}$, forms a subchromosome of chromosome C given in Example 3.

3.1 Homology Teams

In this section, we introduce a series of definitions to capture the notion of neighboring genes and conserved gene clusters. Two genes g_i and g_j are considered neighboring whenever $\Delta(g_i, g_j) \leq \delta$ for a chosen parameter $\delta > 0$. Most of our definitions are generalizations of those originally made by Béal et al. in an earlier model which assumed a one-to-one homologous relationship, (Béal et al., 2004). In generalizing these concepts, we now carefully distinguish between a collection of genes as opposed to a collection of families. A deeper comparison between our model and this earlier model is discussed in Section 3.2.

Definition 7 *Given chromosome $C = (\Sigma, X)$, a subsequence C' is termed a* δ-subsequence *of C if every pair of adjacent genes of C' are separated by distance of at most δ. We further say that C' is a* δ-run *if it is a maximal δ-subsequence with respect to inclusion.*

Definition 8 *Given chromosome $C = (\Sigma, X)$, we say that $\Sigma' \subseteq \Sigma$ is a* δ-chain *of C if there exists some δ-subsequence C' of C such that $\Sigma' = \Sigma(C')$. We say that C' is a* witness *to such a δ-chain.*

Example 9 *When $\delta = 2$, the chromosome of Example 3 is comprised of two δ-runs, $\{(1, c)\}$ and $\{(4, e), (5, d), (6, a), (8, b), (10, e), (11, d)\}$. Examples of δ-chains include $\{d, e\}$ as witnessed by $\{(4, e), (5, d)\}$, $\{a, b, d, e\}$ as witnessed by $\{(4, e), (5, d), (6, a), (8, b), (10, e), (11, d)\}$, as well as $\{d, a, b\}$ as witnessed by $\{(5, d), (6, a), (8, b)\}$.*

Definition 10 *Let Σ be the set of common homology families for two chromosomes C and D. A subset $\Sigma' \subseteq \Sigma$ is a* δ-set *of C and D if Σ' is a δ-chain of both C and D. We say that a pair (C', D')* witnesses *the δ-set if C' (resp. D') is a witness to the δ-chain of C (resp. D).*

Definition 11 *For two chromosomes C and D, a δ-set Σ' is a* δ-team *if it has a witness pair (C', D') such that no other δ-set $\Sigma^* \supset \Sigma$ has a witness of the form (C^*, D^*) with $C^* \supseteq C'$ and $D^* \supseteq D'$.*

Example 12 *Consider chromosomes $C =$* `<c**eda*b*ed>` *and $D =$* `<ab*ba**c*dee*a>`*. With $\delta = 2$, the non-trivial* δ-sets *of C and D are $\{a, b\}$, $\{a, e\}$, $\{d, e\}$ and $\{a, d, e\}$. Notice that $\{a, d\}$ is not a δ-set, even though $\{a, d, e\}$ is. The non-trivial* δ-teams *of C and D are: $\{a, b\}$, $\{d, e\}$ and $\{a, d, e\}$. Notice that $\{d, e\}$ qualifies as a δ-team due to witness $C' = \{(10, e), (11, d)\}$ and $D' = \{(10, d), (11, e), (12, e)\}$.*

A δ-team is then a maximal δ-set with respect to subset inclusion of a witness. As it is a set of homology families, we informally call this a homology team. We note that the genes in the witnesses of a δ-team may occur in different orders in the underlying chromosomes.

3.2 Comparison to a Previous Model

Our model is a direct generalization of a previous model for gene teams which assumes a one-to-one homologous relationship (Béal et al., 2004). Our definitions of δ-chain, δ-set and δ-team reduce precisely to those original definitions when applied to a pair of chromosomes with a one-to-one homologous relationship. Unfortunately, many convenient properties in the restricted setting do not apply in general. Most significantly, the following properties are proven in the restricted setting (Béal et al., 2004). If Σ_1 and Σ_2 are δ-chains of C and $\Sigma_1 \cap \Sigma_2 \neq \emptyset$, then $\Sigma_1 \cup \Sigma_2$ is also a δ-chain. A similar statement follows for δ-sets, and thus it is shown that the collection of δ-teams forms a disjoint partition of the underlying genes. These properties no longer hold in the general setting. If we consider chromosome D from Example 12, we see that both $\{a, b\}$ and $\{a, e\}$ are δ-chains with $\delta = 2$, yet $\{a, b, e\}$ is not such a chain. We see that a is in δ-team $\{a, b\}$ as well as $\{a, d, e\}$. In general, these vastly different structural properties complicate the algorithmic search for δ-teams.

4 Algorithmic Approach

We consider two chromosomes C and D represented as sequences over a common alphabet Σ, as in Definition 1. We let m and n denote the number of genes in C and D, respectively, excluding genes which are not represented in the common alphabet. In this section, we present a divide-and-conquer algorithm which discovers each δ-team for a given δ in $O(mn)$ time and $O(m+n)$ space.

Our presentation is as follows. We begin in Section 4.1, with a comparison to the previous algorithm which applies only for one-to-one homologous relationships (Béal et al., 2004). Section 4.2 contains an overview of our algorithm followed by a detailed presentation. A proof of correctness is given in Section 4.3, followed by an analysis of the time and space usage in Section 4.4. There are several possible forms of desired output which are discussed in Section 4.5 and an extension to circular chromosomes is given in Section 4.6.

4.1 Comparison to A Previous Algorithm

Assuming one-to-one homologous relationships, gene teams between two chromosomes can be identified by an $O((m+n)\log^2(m+n))$-time algorithm (Béal et al., 2004). Both that algorithm and ours rely on a divide-and-conquer approach which begins by breaking one of the chromosomes into subchromosomes separated by a gap greater than δ between the position of relevant genes. The algorithm of Béal et al. further relies on the fact that each gene in the first chromosome has at most one homolog in the second chromosome. Thus, the second chromosome is partitioned into two disjoint subsequences (though not necessarily subchromosomes) in step with the division of the first chromosome. Thereafter, they recurse on independent subproblems.

When a gene may have multiple homologs, the situation is quite different as noted in Section 3.2. The homology teams do not necessarily form a partition of the homology families, so we cannot assume a linear bound on the cumulative cardinality of all such teams. Algorithmically, our approach differs from that of (Béal et al., 2004) at the division into subproblems. Though we will divide the first chromosome at gaps of greater than δ, we are unable to divide the second chromosome in step with the first; genes from the second chromosome may have homologs on several pieces of the first chromosome.

4.2 Algorithm Presentation

Our algorithm for identifying common teams between chromosomes C and D begins with the top-level routine, FindTeams(C, D), detailed in Figure 1 at the end of this section. Our approach is to split chromosome C into its δ-runs, based upon gaps of size greater than δ between genes of common families. Then, a recursive subproblem can be defined for each such δ-run, when paired with the *subsequence* of the entire other chromosome formed by taking all genes with homologs in the δ-run. The recursion is formulated by the FindTeamsRecurse(A, B) routine, detailed in Figure 2, to identify common teams between (sub)chromosomes A and B. We also define two additional utility functions. Figure 3 describes the MarkCommonAlphabet(A, B), used to compute the common alphabet of (sub)chromosomes A and B. Figure 4 describes the FindFirstRun$(A, \text{TIMESTAMP})$ routine, used to identify the first δ-run of a (sub)chromosome. A straightforward implementation of this recursion leads to an $O(mn)$ time bound. However much care is needed to achieve this bound with overall linear space usage. In the remainder of this section, we discuss the more subtle algorithmic issues.

The recursive formulation has a depth bounded by $O(m+n)$. If we then wish to maintain overall space usage of $O(m+n)$ we must take great care in succinctly representing the subproblems. Specifically, we use only constant space to represent the parameters and local variables of

the recursion. We note the distinction made in Definition 5 between a *subsequence* and a *subchromosome*. Conceptually, each subproblem is defined based upon a pair of *subsequences* of the original chromosomes, since genes which are not common to the two subsequences would have been omitted at a higher level of the recursion. Representing each of those subsequences directly would require space proportional to the cardinality of such subsequences, at each recursive level. Instead, we represent the subproblem by the pair of *subchromosomes* which contain the *subsequences*. Each subchromosome can be represented in constant space, as a triple (*chrom*,*left*,*right*), where *chrom* is a reference to the original chromosome and *left* and *right* are references to the leftmost and rightmost genes of the chromosome, respectively.

Another subtlety arises due to the need to discern the underlying subsequences within the represented subchromosomes. This seems to require a representation of the locally common alphabet at each recursive step, whether recomputed or sent as a parameter. Again, we must be careful to achieve overall linear space usage. Our solution is to implicitly represent all locally common alphabets of the active recursion simultaneously, through the use of the following *global* structures. There exists an integer GLOBALTIME, and two integer arrays STAMP[ ] and TEMPMARK[ ], with entries for each member of Σ. Since the local alphabet at one level of recursion is a subset of the local alphabet at the parent level, the timestamps for common alphabet symbols can be incremented to mark the local alphabet, while not violating the parent's ability to discern its own alphabet.

Finally, we note one additional subtlety in our approach. It is important in the analysis that we break a subchromosome into *all* appropriate δ-runs at a given level of our recursion. The algorithm of Béal et al. breaks off only the first run of a chromosome at a given recursive level; the remainder is left within a single subproblem, though likely to be broken at deeper levels of recursion. Our time analysis could not be applied if we tried a similar approach here due to our higher overhead cost at each level in recomputing the effective local alphabet. By instead insisting that a subproblem be broken into all of its δ-runs, we are assured that the following recursive level can only result in further subproblems by a split of the *opposite* chromosome. That is, when invoking FINDTEAMSRECURSE(A, B), we rely on the invariant that B is known to contain a single δ-run with respect to the common alphabet of A and B. Therefore, if A is found to containse a single δ-run then a team has been identified. Otherwise, A is fully decomposed into its δ-runs and another level of the recursion is started for each segment of A, serving as the *second* parameter, and B serving as the *first* parameter. Note that because the common alphabet at the new recursive level may be a strict subset of that at the parent level, B may no longer be a single δ-run. This leads us to the alternating, double recurrence, analyzed in Section 4.4.

4.3 Algorithm Correctness

We begin with several lemmas involving the use of the global arrays in representing common alphabets during the recursion.

Lemma 13 *At the onset of* MARKCOMMONALPHABET, STAMP$[f] \leq$ TEMPMARK$[f] \leq$ GLOBALTIME, *for each alphabet symbol* f.

Proof: STAMP$[f]$, TEMPMARK$[f]$, and GLOBALTIME are initialized to 0 for each c in FINDTEAMS. After that, they are only modified within MARKCOMMONALPHABET. The invariant is evident from Figure 3. ∎

Lemma 14 *A call to* MARKCOMMONALPHABET(A, B) *effects array* STAMP[ ] *as follows:*

```
FindTeams(C, D)
    // Initialize Global Data
    globaltime ← 0
    for each f in Σ
      Stamp[f] ← 0; TempMark[f] ← 0

    // Decompose problem based on initial runs of C
    localtime ← MarkCommonAlphabet(C, D)
    C_rest ← C
    repeat
      C_first ← FindFirstRun(C_rest, localtime)
      C_rest ← C_rest \ C_first
      FindTeamsRecurse(D, C_first)
    until (C_rest = ∅)
```

Figure 1: The FindTeams procedure.

```
FindTeamsRecurse(A, B)
    localtime ← MarkCommonAlphabet(A, B)
    A_first ← FindFirstRun(A, localtime)
    A_rest ← A \ A_first
    if A_rest = ∅ then                          // common alphabet forms a δ-team
      ReportTeam(A, B)
    else                                        // decompose for each δ-run of A
      repeat
        FindTeamsRecurse(B, A_first)            // note well: reversal of parameters
        A_first ← FindFirstRun(A_rest, localtime)
        A_rest ← A_rest \ A_first
      until (A_rest = ∅)
      FindTeamsRecurse(B, A_first)
```

Figure 2: The FindTeamsRecurse procedure.

```
// Relies on use of global data: globaltime, TempMark[ ], Stamp[ ]
MarkCommonAlphabet(A, B)
    globaltime = globaltime + 1
    for each g in A
      Let f be the homology family of g
      TempMark[f] ← globaltime
    for each g in B
      Let f be the homology family of g
      if TempMark[f] = globaltime then
        Stamp[f] ← globaltime
    return globaltime
```

Figure 3: The MarkCommonAlphabet procedure.

FINDFIRSTRUN(A, TIMESTAMP)
  endRun ← first gene in A with STAMP[f] ≥ TIMESTAMP
  nextGene ← gene which follows *endRun* in A
  while (*nextGene* is well defined AND Δ(*endRun*, *nextGene*) ≤ δ) **do**
    Let f be the homology family of *nextGene*
    if STAMP[f] ≥ TIMESTAMP **then**
      // if *nextGene* is in common family, extend the run
      endRun ← *nextGene*
    // in either case continue advancing
    nextGene ← gene which follows *nextGene* in A
  return subchromosome up to and including *endRun*

Figure 4: The FINDFIRSTRUN procedure.

$$\text{STAMP}[f] = \begin{cases} \text{GLOBALTIME} & \text{if } f \in \Sigma(A) \cap \Sigma(B) \\ \text{left unchanged} & \text{otherwise} \end{cases}$$

Proof: Evident from Figure 3 and Lemma 13. ∎

Lemma 15 *A call to* FINDTEAMSRECURSE(A, B) *effects* STAMP[ ] *as follows. The value* STAMP[f] *remains unchanged for all $f \notin \Sigma(A) \cap \Sigma(B)$, and is not decreased for $f \in \Sigma(A) \cap \Sigma(B)$.*

Proof: The array STAMP[ ] is only modified within calls to MARKCOMMONALPHABET. The call to MARKCOMMONALPHABET made from the first line of the FINDTEAMSRECURSE routine respects our claim. This follows directly from Lemmas 13 and 14 with the observation that GLOBALTIME increases.

If the call to FINDTEAMSRECURSE invokes recursion from within the "repeat" loop, we can prove this lemma by induction on the maximum recursive depth. By inspection of Figure 4, we can see that each assignment to A_{first} is a subchromosome of A. Thus, $\Sigma(A_{\text{first}}) \cap \Sigma(B) \subseteq \Sigma(A) \cap \Sigma(B)$. By induction, this means that STAMP[f] remains unchanged within the recursion for $f \notin \Sigma(A) \cap \Sigma(B) \supseteq \Sigma(A_{\text{first}}) \cap \Sigma(B)$. ∎

Lemma 16 *Whether invoked from* FINDTEAMS(X, Y) *or* FINDTEAMSRECURSE(X, Y), *it must be that upon onset of* FINDFIRSTRUN,

$$\begin{array}{lcll} \text{STAMP}[f] & \geq & \text{TIMESTAMP} & \text{if } f \in \Sigma(X) \cap \Sigma(Y) \\ \text{STAMP}[f] & < & \text{TIMESTAMP} & \text{if } f \notin \Sigma(X) \cap \Sigma(Y) \end{array}$$

Proof: After the first call to MARKCOMMONALPHABET, the condition is true due to Lemmas 13 and 14 with the observation that GLOBALTIME increases. It remains true throughout the "repeat" loop due to Lemma 15. ∎

Lemma 17 *For a call to* FINDFIRSTRUN(A, TIMESTAMP), *let A' be the returned subchromosome. Define A^* to be the subsequence of A containing those genes for which* STAMP[f] ≥ TIMESTAMP. *If $A^* \neq \emptyset$ then $A' \cap A^*$ is the first δ-run of A^*.*

Proof: We refer to the description of FINDFIRSTRUN from Figure 4. If $A^* = \emptyset$, the claim is trivial. Otherwise, we designate the first δ-run of A^* as genes $g_1, g_2, \ldots, g_k$ for some k, noting that $\Delta(g_i, g_{i+1}) \leq \delta$ for all $1 \leq i < k$. Let f_i denote the family for each gene g_i.

(*first δ-run of A^**) $\subseteq$ ($A' \cap A^*$)*:* We show by induction that for each $1 \leq i \leq k$, *endRun* is at some point equal to g_i. *endRun* is initially set to g_1. If we assume *endRun* $= g_i$ at some point for each $1 \leq i < k$, we show that it is so for g_{i+1}. Since $\Delta(g_i, g_{i+1}) \leq \delta$, the while loop will advance until *nextGene* is equal to g_{i+1}. Then, since STAMP$[f_{i+1}] \geq$ TIMESTAMP, *endRun* will be set to g_{i+1}.

($A' \cap A^*$) $\subseteq$ (*first δ-run of A^**)*:* By the previous paragraph, there is a point at which *endRun* is equal to g_k. We claim that *endRun* cannot be updated after that point. The only way in which *endRun* is updated would be if there exists some g_ℓ with $\Delta(g_k, g_\ell) \leq \delta$ and with STAMP$[f_\ell] \geq$ TIMESTAMP. Yet if such a g_ℓ existed, then g_k could not end a δ-run of A^*. ∎

Lemma 18 *For each call to* FINDTEAMSRECURSE(A, B), $\Sigma(A) \cap \Sigma(B)$ *is a δ-chain of B.*

Proof: For a call to FINDTEAMSRECURSE(A, B), we consider the routine from which it was invoked. From that caller there must exists subchromosomes which we will denote as A_{caller} and B_{caller} such that the following are true. $A = B_{\text{caller}}$; B is the first δ-run of A_{caller} when restricted to the common alphabet $\Sigma(A_{\text{caller}}) \cap \Sigma(B_{\text{caller}})$. This second property follows from Lemma 17. This means that $\Sigma(A_{\text{caller}}) \cap \Sigma(B_{\text{caller}}) = \Sigma(A_{\text{caller}}) \cap \Sigma(A)$ is a δ-chain of B. Furthermore, since B is a subset of A_{caller}, the witness to the above δ-chain cannot contain any symbols from $\Sigma(A_{\text{caller}}) \setminus \Sigma(B)$, and thus it must be that $\Sigma(B) \cap \Sigma(A)$ is a δ-chain of B. ∎

Lemma 19 *Given (sub)chromosomes A and B, if A consists of δ-runs $A_1, A_2, \ldots, A_k$, then*

$$\{\Sigma' : \Sigma' \text{ is a } \delta\text{-team of } A \text{ and } B\} = \bigcup_{1 \leq i \leq k} \{\Sigma' : \Sigma' \text{ is a } \delta\text{-team of } A_i \text{ and } B\}$$

Proof: Leftside $\subseteq$ Rightside: Assume Σ' is a δ-team of A and B. From Definition 11, we consider any witness pair A' and B' which are δ-subsequences of A and B respectively, with $\Sigma' = \Sigma(A') = \Sigma(B')$. By Definition, if A' is not itself a δ-run of A, it must be a subset of some δ-run A_i of A. Thus Σ' is a δ-set of A_i and B, as witnessed by pair (A', B'). It must be that Σ' is a δ-team of A_i and B. Assume for contradiction there exists some $\Sigma^* \supset \Sigma'$ with witness pair A^* and B^* such that $A' \subseteq A^* \subseteq A_i$ and $B' \subseteq B^*$. Such a Σ^* must be a δ-set of A and B, yet this contradicts the original assumption that Σ' is a δ-team of A and B.

Rightside $\subseteq$ Leftside: Assume Σ' is a δ-team of A_i and B, and witnessed by A' and B'. This same witness pair demonstrates that Σ' is a δ-set of A and B. It must be that Σ' is a δ-team of A and B. Assume for contradiction that there exists some $\Sigma^* \supset \Sigma'$ with witness pair A^* and B^* such that $A' \subseteq A^* \subseteq A$ and $B' \subseteq B^*$. Since there is a gap of greater than δ between genes from separate δ-runs, it must be that such $A^* \subseteq A_i$ and thus Σ^* a δ-set of A_i and B. This contradicts the original assumption that Σ' is a δ-team of A_i and B. ∎

Lemma 20 *For the recursion which results from a given call to* FINDTEAMSRECURSE(A', B')*:*

(1) If REPORTTEAM(A, B) *is called,* $\Sigma(A) \cap \Sigma(B)$ *forms a δ-team of A' and B'.*

(2) If Σ' is a δ-team for A' and B', REPORTTEAM(A, B) *will be called for some A and B such that $\Sigma' = \Sigma(A) \cap \Sigma(B)$.*

Proof: We prove this claim by induction on the depth of recursion. If the call does not invoke any recursion, then it is because the initial call to FindFirstRun determined, as shown in Lemma 17, that A' is δ-run relative to the alphabet $\Sigma(A') \cap \Sigma(B')$. Since B' is known to be a δ-chain as well, by Lemma 18, then $\Sigma(A') \cap \Sigma(B')$ is a δ-set of A' and B'. Since the witnesses include all of A' and B', this must in fact be a δ-team. Furthermore, there could not possible be any other δ-team for A' and B' as its witnesses would necessarily be a subset of A' and B' which we have just seen as witnesses to a δ-team.

If recursion is invoked, FindTeamsRecurse(B', A'_i) is called for each δ-run of A'. Applying induction together with Lemma 19 completes the claim. ∎

Theorem 21 *For a given call to* FindTeams(C, D)*:*

(1) If ReportTeam(A, B) *is called,* $\Sigma(A) \cap \Sigma(B)$ *forms a* δ*-team of* C *and* D*.*

(2) If Σ' *is a* δ*-team for* C *and* D*,* ReportTeam(A, B) *will be called for some* A *and* B *such that* $\Sigma' = \Sigma(A) \cap \Sigma(B)$*.*

Proof: The call to FindTeams(C, D) results in a call to FindTeamsRecurse(D, C_i) for each δ-run of C. The claim is a direct result of Lemmas 19 and 20. ∎

4.4 Analysis of Time and Space Usage

In this section, we let A and B denote subchromosomes of the original chromosomes C and D (though not necessarily respectively), and a and b the respective number of genes in A and B.

We begin by examining the running time of a call to FindTeamsRecurse(A, B). We let Σ' denote $\Sigma(A) \cap \Sigma(B)$. We let k denote the number of δ-runs of A over Σ', and $a_1, \ldots, a_k$ the cardinalities of those δ-runs. It is easily seen that the running time for this procedure satisfies the following recurrence, for some constant $c > 0$.

$$T(a,b) \leq \begin{cases} c(a+b) & \text{when } k = 1 \\ c(a+b) + \sum_{1 \leq i \leq k} T(b, a_i) & \text{when } k \geq 2 \end{cases}$$

The overhead cost of $c(a+b)$ is due to the call made to MarkCommonAlphabet(A, B) and the cumulative time spent on the k calls to FindFirstRun. When $k \geq 2$, the summation is due to the recursion.

Lemma 22 $T(a,b) = O(ab)$

Proof: We prove a more specific claim, that there exists a constant $c' > 0$ such that $T(a,b) \leq c'ab - 2ca - 3cb$. We prove this by induction on $(a+b)$.

As a base case, when $a = b = 1$, k must be 1 and thus the running time is bounded by a constant. By choosing c' to be sufficiently large, the claim can be made true. In general, we consider larger values of $(a+b)$. For a given invocation of the procedure, we consider two possible cases, depending on whether $k = 1$. When $k = 1$, then the recurrence dictates that $T(a,b) \leq c(a+b)$. By choosing $c' \geq 8c$, we see that

$$\begin{aligned} T(a,b) &\leq c(a+b) = (3ca - 2ca) + (4cb - 3cb) \\ &\leq 4c(a+b) - 2ca - 3cb \leq 4c(2ab) - 2ca - 3cb \\ &\leq c'ab - 2ca - 3cb \end{aligned}$$

When $k > 1$, we apply induction, to see that

$$\begin{aligned} T(a,b) &\leq c(a+b) + \sum_{1 \leq i \leq k} T(b, a_i) \\ &\leq c(a+b) + \sum_{1 \leq i \leq k} (c'ba_i - 2cb - 3ca_i) \\ &\leq c(a+b) - 2kcb + (c'b - 3c)a \\ &= c'ba - 2ca - (2k-1)cb \\ &\leq c'ba - 2ca - 3cb \end{aligned}$$

The third inequality holds since $(c'b - 3c) > 0$ and $\sum_i a_i \leq a$. The final inequality holds as $k \geq 2$. ∎

Theorem 23 FINDTEAMS(C, D) *runs in* $O(mn)$ *time.*

Proof: The call to FINDTEAMS(C, D) results in a call to FINDTEAMSRECURSE(D, C_i) for each of k δ-runs of C, with m_i the size of C_i. Therefore, by Lemma 22, the overall time is $O(m_1 n + m_2 n + \cdots + m_k n) = O(mn)$. ∎

Theorem 24 FINDTEAMS(C, D) *uses* $O(m + n)$ *space.*

Proof: Global arrays STAMP[ ] and TEMPMARK[ ] use space proportional to $\Sigma = O(m + n)$. The only additional space we must account for is that required for the recursion. The parameters and local variables within a single level of the recursion requires constant space. Since at least one gene is removed at each recursive level, then can be at most $O(m + n)$ active calls at any one time. ∎

4.5 Variants on Algorithm Output

Theorem 21 guarantees that for each δ-team, the procedure REPORTTEAM(A, B) will be called, where A and B are subchromosomes whose common alphabet defines the δ-team. As of yet, we have not explicitly declared such a routine. Our running time analysis of Section 4.4 holds true so long as the time spent within a call to REPORTTEAM(A, B) is $O(a + b)$. By using a technique similar to MARKCOMMONALPHABET, the δ-team as well as its witness, can be output within $O(a + b)$ time.

In Section 4.1, we noted that homology teams do not necessarily form a partition, and thus that the cumulative cardinality may be super-linear. If the output is to be considered as part of the memory usage, then an $O(m + n)$ space bound no longer applies. In fact, the number of teams is not necessarily bounded by $O(m + n)$.

One remedy is to flush the output to disk as it is generated, in which case it need not be considered as part of the memory usage. If a strict linear space bound is desired, an alternative is to output information for the most "meaningful" δ-teams. For example, for each family f, we could track the maximal-cardinality δ-team which includes c. To keep the linear bound on the space usage, a candidate δ-team can be stored via the two triples which define the containing subchromosomes; the resulting δ-teams could then be reconstructed at the conclusion.

4.6 Circular Chromosomes

We note briefly that our algorithm easily extends to the case of circular chromosomes with appropriate modification to our representation of a subchromosome and the procedure FINDFIRSTRUN. If a gap of greater than δ is found, a circular chromosome can be viewed linearly.

5 Experiments

As a proof of concept of our model and algorithm, we have performed experiments demonstrating the use of our techniques in identifying conserved gene clusters. We describe the experiments in this section, exploring several practical issues in the application of our techniques, such as the initial classification of genes into homology families, the effect of the parameter δ on the results, and the statistical significance of the results. Yet, we feel that our experiments provide results of biological significance on their own accord.

Specifically, we perform analysis on two prokaryotic genomes: *E. coli* K12 and *B. subtilis*. Because these genomes are well-studied, many of the genes and their presumed homologs have been previously identified and annotated. Furthermore, over 50% of the genes in each genome are identified from homology families common to both genomes, with more than two-thirds of those displaying one-to-many or many-to-many homologous relationships. Thus, these are genomes that could not be effectively analyzed by the previous gene team model, which was restricted to one-to-one orthologous relationships. This highlights the importance of considering more general orthologous relationships when identifying conserved gene clusters.

The assessment of our identified teams is based upon a comparison with known *E. coli* operons. While operons and conserved gene clusters are two different concepts, they are closely-related in prokaryotic genomes. First, one of the known evolutionary forces behind the conserved gene clusters is the need of coordinated transcription, which is essentially how operon is defined. Therefore, a conserved gene cluster is likely to suggest an operon in the context of a single genome. Second, if an operon is conserved during evolution because of the functional constraint, it will become a conserved gene cluster in the context of pairwise genome comparison. Therefore, the known operons serve as reference data in our experiment, as we would expect to find them correlated with our identified homology teams (and indeed, do).

5.1 Experiment Methodology

The first step in the analysis process is the identification of homologous relationships among the genes. This is known to be a difficult problem. The difficulty lies in the fact that the sequence similarities of genes do not correspond exactly with their evolutionary relationship (Fitch, 2000). For the ease of experimentation, we took advantage of existing data from NCBI's Cluster of Orthologous Groups (COG) database (National Center for Biotechnology Information, 2004). The genes in the downloaded files are already annotated with a COG number, which is uniquely assigned to each of 4873 COGs currently compiled at NCBI. All genes with the same COG number will be considered as homologs, and thus each COG serves as our operational definition of a homology family. This method of classifying homologs is different from the one used in previous papers (Luc et al., 2003; Overbeek et al., 1999), namely the bidirectional best hit method which guarantees one-to-one homologous relationships, though at the possible expense of ignoring some true homologs.

For *E. coli* K12 and *B. subtilis*, a significant percentage of genes are from COGs common to these two genomes. The precise statistics are given in Table 1. Furthermore, the distribution of COGs based on the arity of the homologous relationship is shown in Table 2, demonstrating the predominance of one-to-many or many-to-many relationships.

With the data drawn from the NCBI database, we can apply the algorithm given in Section 4. Rather than analyze each reported team,, we restricted the results only to the maximal-cardinality team containing each individual COG (see Section 4.5). The remaining decision in the algorithm application is the choice of parameter, δ, which controls the size of the allowable gap between genes witnessing a team. As δ grows, the reported teams will have larger cardinality and would eventually

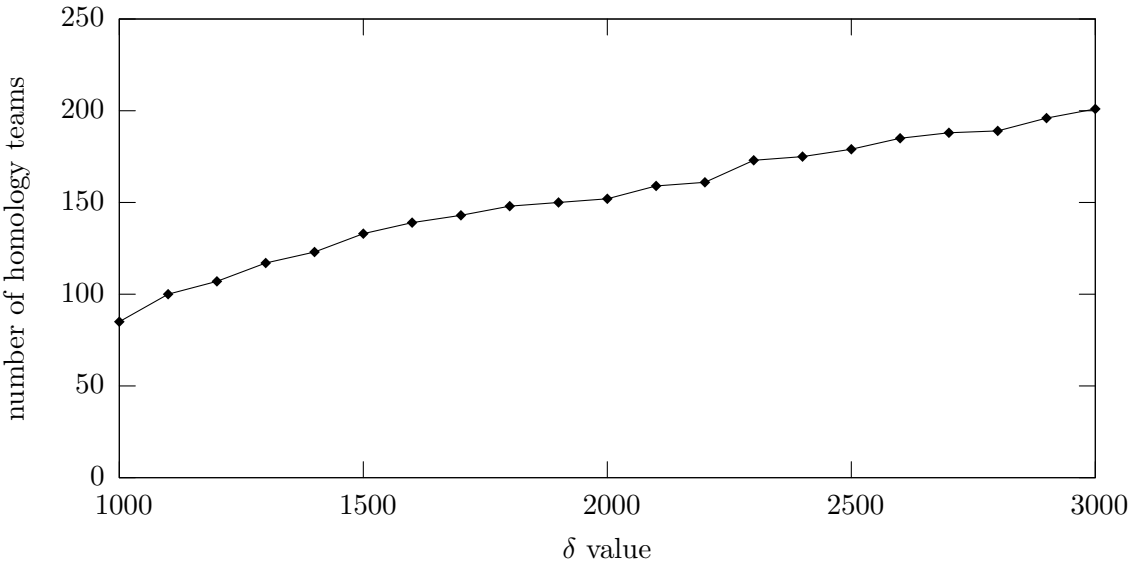

Figure 5: The number of teams identified for varying values of δ (measured in bp).

include many false positives, in terms of predicted conserved clusters. On the other hand, if δ is too small, we may miss many potential conserved clusters. Figure 5 shows the number of such teams identified as δ ranges from 1000bp to 3000bp. Roughly speaking, this base pair distance translates to a separation of one or two genes between neighboring genes in a witness. There is no clear approach to picking the "ideal" value of δ for a given analysis. In this experiment, we instead chose a few of the independently known operons conserved in both species to serve as "benchmarks" in calibrating our choice of δ. Specifically, we examined one ribosomal protein cluster, ATP synthase operon, tryptophan biosynthesis operon and ABC ribose transport operon. We chose $\delta = 1900$bp, as this was roughly the minimum value at which those benchmark operons were reconstructed. For this value, we detected 150 corresponding COG teams; the distribution of these teams, based on cardinality, is shown in Figure 6. Though we only used four of the known operons as benchmarks in calibrating δ, the reported 150 teams seems reasonable in the following context. In a study of *B. subtilis* by Okuda (Okuda et al., 2002), 467 operons are predicted by cotranscription study, 201 of which are multi-transcribed. The number of multi-gene operons that are common to both species should not be too much smaller than this, as many operons have functions that are important to organisms and therefore conserved.

5.2 Analysis of Identified Clusters

In this section, we examine many of the 150 homology teams which were identified by our algorithm during the pairwise comparison of *E. coli* and *B. subtilis*, comparing them manually to known *E. coli* operons. For this comparison we consider the annotations of the COGs, as taken from the NCBI database (National Center for Biotechnology Information, 2004), as well as the witnessing *E. coli* gene name. We then compare those names to the annotated operons taken from the RegulonDB database (Salgado et al., 2004).

Of the 150 identified teams, we selected only a subset for this manual analysis. First we excluded ribosomal clusters because they are widespread in the genome and less interesting. There are more than 50 ribosomal protein operons in *E. coli*, and indeed many teams we identified belong to this

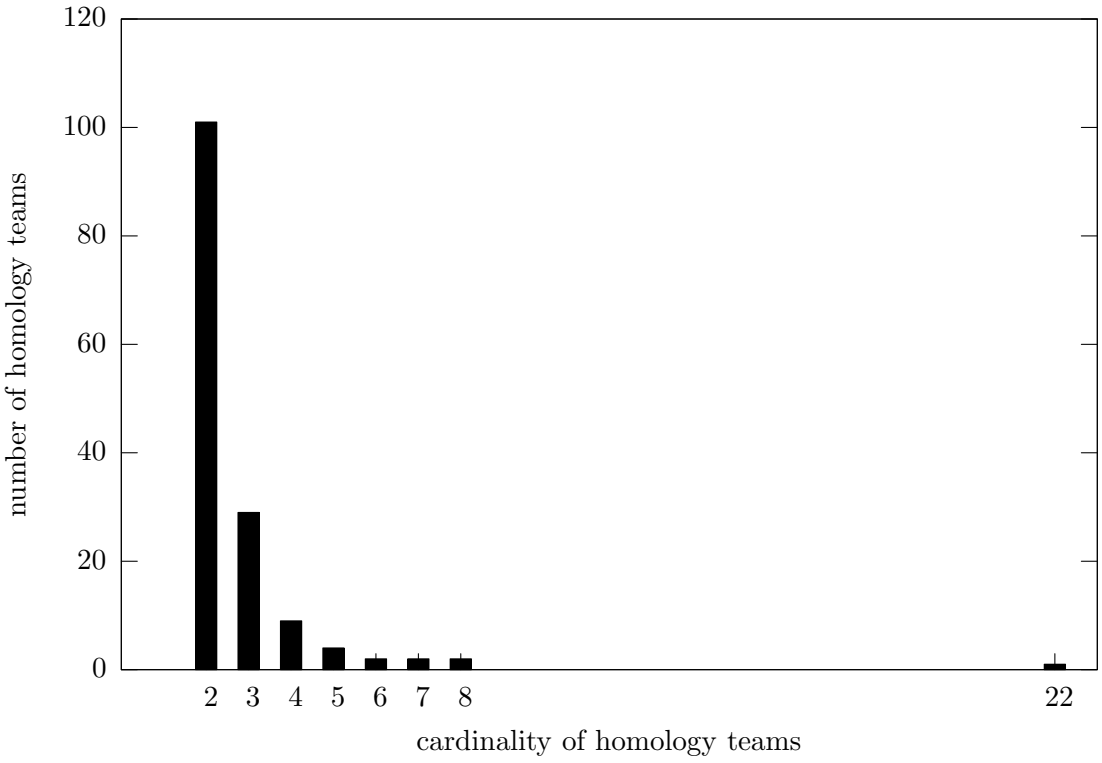

Figure 6: The cardinality of the 150 reported teams, for $\delta = 1900$bp

category. The largest team, with 22 genes, is a union of two ribosomal protein operons of *E. coli*. Second, we did not analyze the teams of size two, as these may not be statistically significant and are less interesting from a biological perspective because of the small cardinality.

The remaining 43 teams are manually analyzed in this way. We further apply a test of statistical significance to each team, in the style of Durand and Sankoff (Durand and Sankoff, 2003); details are described in Appendix A. Our results for the ten most significant teams are shown in Tables 3–5. In some cases, a team may match several operons. In this case, all of them will be given in the tables, but the functional annotation refers only to the major operon. The data for all 43 teams can be accessed at `http://euler.slu.edu/~goldwasser/homologyteams/data`.

We can generally classify the identified teams into four categories, based upon their correlation with operons: those that match exactly with some documented operons; those that match most or part of some documented operons or contain additional genes; those that match some predicted operons; and those that do not match any operons. From all 43 teams we analyzed, there are 10, 20, 11 and 2 teams in each category, respectively. The examples of the first category include operons of ATP synthase, Menaquinone (vitamin K2) biosynthesis and Histidine biosynthesis. These are clearly the most desired type of clusters. The second category is somewhat mixed. In some cases, there is only slight exception, e.g., one team matches all of flagellar biosynthesis operon, fliFGHIJK, but has one additional gene fliE, which is functionally related but not shown by experiments to be co-expressed with other fli-genes. In other cases, there is a larger difference. For example, cytochrome o ubiquinol oxidase subunit operon is composed of cyoABCDE, but we only observe a team consisted of cyoBCD. We think there are a few explanations for this. First of all, the gene-to-COG assignment in the NCBI database is imperfect, with presumed homologs sometimes assigned

Genome	Number of genes		
	overall	from an identified COG	from an identified COG common to both genomes
E. coli	4290	3289 (76.7%)	2332 (54.4%)
B. subtilis	4101	2818 (68.7%)	2339 (57.0%)

Table 1: Percentage of overall genes identified in clusters of orthologous groups

COG type	No. COGs	Genes assigned to such COGs	
		E. coli	*B. subtilis*
one-to-one	578	578	578
one-to-many	299	517	568
many-to-many	260	1237	1193

Table 2: Distribution of COGs based on the type of homologous relationship in the compared genomes

homology team: witnessing genes' names in *E. coli* (with COG annotation from NCBI)		
Size	Matched operon(s) [functional annotation of major operon]	Statistical Significance
F0F1-type ATP synthase, epsilon subunit (COG0355-atpC), F0F1-type ATP synthase, beta subunit (COG0055-atpD), F0F1-type ATP synthase, gamma subunit (COG0224-atpG), F0F1-type ATP synthase, alpha subunit (COG0056-atpA), F0F1-type ATP synthase, delta subunit (COG0712-atpH), F0F1-type ATP synthase, subunit b (COG0711-atpF), F0F1-type ATP synthase, subunit c/Archaeal/vacuolar-type H+ATPase, subunit K (COG0636-atpE), F0F1-type ATP synthase, subunit a (COG0356-atpB)		
8	atpIBEFHAGDC [ATP synthase]	$4.0e-17$
Flagellar hook-basal body protein (COG1677-fliE), Flagellar biosynthesis/type III secretory pathway lipoprotein (COG1766-fliF), Flagellar motor switch protein (COG1536-fliG), Flagellar biosynthesis/type III secretory pathway protein (COG1317-fliH), Flagellar biosynthesis/type III secretory pathway ATPase (COG1157-fliI), Flagellar biosynthesis chaperone (COG2882-fliJ), Flagellar hook-length control protein (COG3144-fliK)		
7	fliE, fliFGHIJK [Flagellar biosynthesis]	$5.0e-15$
Imidazoleglycerol-phosphate dehydratase (COG0131-hisB), Glutamine amidotransferase (COG0118-hisH), Phosphoribosylformimino-5-aminoimidazole carboxamide ribonucleotide (ProFAR) isomerase (COG0106-hisA), Imidazoleglycerol-phosphate synthase (COG0107-hisF), Phosphoribosyl-ATP pyrophosphohydrolase (COG0140-hisI)		
5	hisGDCBHAFI [Histidine biosynthesis]	$1.2e-10$

Table 3: Identified homology teams and matched operons, with $\delta = 1900bp$.

homology team: witnessing genes names in *E. coli* (with COG annotation from NCBI)		
Size	Matched operon(s) [functional annotation of major operon]	Statistical Significance
Acyl-CoA synthetases (AMP-forming)/AMP-acid ligases II (COG0318-menE), O-succinylbenzoate synthase (COG1441-menC), Dihydroxynaphthoic acid synthase (COG0447-menB), Predicted hydrolases or acyltransferases (COG0596-yfbB), 2-succinyl-6-hydroxy-2,4-cyclohexadiene-1-carboxylate synthase (COG1165-menD), Isochorismate synthase (COG1169-menF)		
6	menFD-yfbB-menBCE [Menaquinone (vitamin K2) biosynthesis]	$1.2e-10$
Nucleotide-binding protein implicated in inhibition of septum formation (COG0424-yhdE), Cell shape-determining protein (COG2891-mreD), Cell shape-determining protein (COG1792-mreC), Actin-like ATPase involved in cell morphogenesis (COG1077-mreB)		
4	mreBCD-yhdE-cafA [shape-determining genes]	$4.8e-08$
Adenosylmethionine-8-amino-7-oxononanoate aminotransferase (COG0161-bioA), Biotin synthase and related enzymes (COG0502-bioB), 7-keto-8-aminopelargonate synthetase and related enzymes (COG0156-bioF), Dethiobiotin synthetase (COG0132-bioD)		
4	bioA, bioBFCD [biotin biosynthesis]	$9.6e-08$
UDP-N-acetylmuramyl pentapeptide phosphotransferase (COG0472-mraY), UDP-N-acetylmuramoylalanine-D-glutamate ligase (COG0771-murD), Bacterial cell division membrane protein (COG0772-ftsW), UDP-N-acetylglucosamine:LPS N-acetylglucosamine transferase (COG0707-murG)		
4	Predicted: murE-murF-mraY-murD-ftsW-murG-murC-ddlB	$9.6e-08$

Table 4: Continuation of Table 3

homology team: witnessing genes names in *E. coli* (with COG annotation from NCBI)		
Size	Matched operon(s) [functional annotation of major operon]	Statistical Significance
3-isopropylmalate dehydratase small subunit (COG0066-leuD), 3-isopropylmalate dehydratase large subunit (COG0065-leuC), Isocitrate/isopropylmalate dehydrogenase (COG0473-leuB), Isopropylmalate/homocitrate/citramalate synthases (COG0119-leuA)		
4	leuLABCD [leucine biosynthesis]	$9.6e-08$
ABC-type ribose transport system, auxiliary component (COG1869-rbsD), ABC-type sugar transport system, ATPase component (COG1129-rbsA), Ribose/xylose/arabinose/galactoside ABC-type transport systems, permease components (COG1172-rbsC), ABC-type sugar transport system, periplasmic component (COG1879-rbsB), Sugar kinases, ribokinase family (COG0524-rbsK), Transcriptional regulators (COG1609-rbsR)		
6	rbsDACBK, rbsR [ABC-type ribose transport system]	$1.2e-07$
Aryl carrier domain (COG3433-entB), Peptide arylation enzymes (COG1021-entE), Isochorismate synthase (COG1169-entC), Dehydrogenases with different specificities (COG1028-entA)		
4	entCEBA [enterobactin biosynthesis]	$7.7e-07$

Table 5: Continuation of Table 4

to different COGs. For example, while *E. coli* trpD is associated with COG0512, *B. subtilis* trpD is associated with COG0547. This prevents us from fully reconstructing this operon. The second reason might be that our chosen value of δ was too small. The third reason is simply that some operons are not fully conserved in both species, which is what we indeed expect. There is another interesting possibility, that is, there are some unknown or unsuspected operon structures. While the authors cannot claim in which cases this is true, we suggest only one example here. One team is composed of fepGDC, fepB and ybdA. While fepGDC is a known operon of ferric enterobactin transport system, ybdA is a "putative transport" gene according to NCBI annotation. Therefore, it seems conceivable that fepGDC and ybdA are indeed localized together during evolution due to functional constraint. Whether the fepGDC operon should include ybdA is not known to us, but it may be interesting to investigate.

We find those teams that match predicted operons very interesting. One example of this is a team matching the predicted operon hisP-hisM-hisQ-hisJ. The genes of this predicted operon, according to their COG annotation, are involved in one process, ABC-type amino acid transport, therefore, seems to be a legitimate candidate of a genuine operon. Since the computational prediction from RegulonDB is based on the information on single genome, our result, from the evolutionary point of view, clearly strengthens the hypothesis.

It is not clear whether there is any function implication for the two teams which do not match any known or predicted operon. For this and other uncertain situations, such as the fepGDC operon discussed earlier, further information might be gained with comparison to more genomes. If a gene cluster is confirmed in more species, it is likely to be a genuine conserved gene cluster that carries functional meaning. In general, while the agreement of conserved gene clusters with operons can validate a computational approach, we believe that it is the discrepancy of these two sets of data that has biological significance, and can serve as a source of new hypothesis on genes and their relations.

Finally, we note that the ordering of the genes witnessing some teams is not necessarily conserved in the two species, for example with the ABC ribose transport operon. There are also occasions where insertions or deletions occur in only one or two of the genomes, for example the tryptophan biosynthesis operon.

6 Conclusion

The identification of conserved gene clusters is important to many biological problems, such as the study of transcriptional regulation, genome evolution, etc. We developed a general mathematical model and algorithm for the problem of pairwise conserved gene cluster identification. We used these techniques to perform analysis on two bacterial genomes, *E. coli* K-12 and *B. subtilis*, and correctly reconstructed many known or predicted operons.

We would like to point out several avenues for further research. Our algorithm was presented for pairwise genome comparison. The algorithm of Béal et al., with one-to-one homologous relationships, was efficiently generalizable for the simultaneous comparison of more than two chromosomes. Our algorithm does not appear to have such an efficient generalization. The straightforward generalization of our algorithm has time complexity $O(n_1 n_2 \dots n_k)$, where n_j is the number of genes of the j^{th} genome.

Secondly, there would be great use for a statistical test to distinguish between coincidental homology teams and those with true biological relevance. Such a test was recently developed by Durand and Sankoff (Durand and Sankoff, 2003), however using a differing combinatorial model for conserved gene clusters. In Appendix A, we discuss a preliminary model we have developed in

the context of homology teams, however more work needs to be done.

Thirdly, for teams that are identified but not known to be a transcription unit, we ask if the transcriptions are indeed coregulated. This can be tested experimentally or computationally by studying the gene expression profiles. In case the hypothesis of coregulation is disproved, one natural question is then what causes the conservation of gene clustering in multiple species.

Finally, the systematic recognition of conserved gene clusters can be used to study the large-scale genome reorganization during evolution. The number and distribution of clusters with respect to size or other criteria can be very helpful information to address this question.

7 Acknowledgments

Much of this research took place while the authors were at Loyola University Chicago. Michael Goldwasser's research is supported, in part, by the National Science Foundation through grants CCR-0208987 and CCR-0417368. The authors are grateful to Howard Laten and Bryan Picket for helpful conversations, to Mathieu Raffinot for helpful comments, and to the anonymous referees for increasing our awareness of the issues raised by Fitch (Fitch, 2000).

References

Béal, M.-P., Bergeron, A., Corteel, S., and Raffinot, M. (2004). An algorithmic view of gene teams. *Theoretical Computer Science*, 320(2–3):395–418.

Bergeron, A., Corteel, S., and Raffinot, M. (2002). The algorithmic of gene teams. In Guigó, R. and Gusfield, D., editors, *Proc. Second Annual Workshop on Algorithms in Bioinformatics (WABI)*, volume 2452 of *Lecture Notes in Computer Science*, pages 464–476, Rome, Italy. Springer-Verlag.

Dandekar, T., Snel, B., Huynen, M., and Bork, P. (1998). Conservation of gene order: a fingerprint of proteins that physically interact. *Trends in Biochemical Sciences*, 23(9):324–328.

Didier, G. (2003). Common intervals of two sequences. In Benson, G. and Page, R. D. M., editors, *Proc. Third Annual Workshop on Algorithms in Bioinformatics (WABI)*, volume 2812 of *Lecture Notes in Computer Science*, pages 17–24, Budapest, Hungary. Springer-Verlag.

Durand, D. and Sankoff, D. (2003). Tests for gene clustering. *Journal of Computational Biology*, 10:453–482.

Fitch, W. M. (1970). Distinguishing homologous from analogous protiens. *Systematic Zoology*, 19(2):99–113.

Fitch, W. M. (2000). Homology: a personal view on some of the problems. *Trends in Genetics*, 16(5):227–231.

Fujibuchi, W., Ogata, H., Matsuda, H., and Kanehisa, M. (2000). Automatic detection of conserved gene clusters in multiple genomes by graph comparion and P-quasi grouping. *Nucleic Acids Research*, 28(20):4029–4036.

Heber, S. and Stoye, J. (2001). Algorithms for finding gene clusters. In Gascuel, O. and Moret, B. M. E., editors, *Proc. First Annual Workshop on Algorithms in Bioinformatics (WABI)*, volume 2149 of *Lecture Notes in Computer Science*, pages 252–263, Aarhus, Denmark. Springer-Verlag.

Luc, N., Risler, J.-L., Bergeron, A., and Raffinot, M. (2003). Gene Teams: a new formalization of gene clusters for comparative genomic. *Computational Biology and Chemistry*, 27:59–67.

National Center for Biotechnology Information (2004). Phylogenetic classification of proteins encoded in complete genomes. `http://www.ncbi.nih.gov/COG`.

Okuda, S., Kawashima, S., and Kanehisa, M. (2002). Database of operons in *Bacillus subtilis*. *Genome Informatics*, 13:496–497.

Overbeek, R., Fonstein, M., D'Souza, M., Pusch, G. D., and Maltsev, N. (1999). The use of gene clusters to infer functional coupling. *Proceedings of the National Academy of Sciences USA*, 96(6):2896–2901.

Salgado, H., Gama-Castro, S., Martínez-Antonio, A., Díaz-Peredo, E., Sánchez-Solano, F., Peralta-Gil, M., Garcia-Alonso, D., Jiménez-Jacinto, V., Santos-Savaleta, A., Bonavides-Martínez, C., and Collado-Vides, J. (2004). RegulonDB (version 4.0): transcriptional regulation, operon organization and growth conditions in Escherichia coli K-12. *Nucleic Acids Research*, 32(Database issue):D303–306. `http://www.cifn.unam.mx/Computational_Genomics/regulondb`.

Schmidt, T. and Stoye, J. (2004). Quadratic time algorithms for finding common intervals in two and more sequences. In Sahinalp, S. C., Muthukrishnan, S., and Dogrusoz, U., editors, *Proc. 15th Annual Symp. on Combinatorial Pattern Matching (CPM)*, volume 3109 of *Lecture Notes in Computer Science*, pages 347–359, Istanbul, Turkey. Springer-Verlag.

Spellman, P. T. and Rubin, G. M. (2002). Evidence for large domains of similarly expressed genes in the drosophila genome. *Journal of Biology*, 1:5.

Tatusov, R. L., Koonin, E. V., and Lipman, D. J. (1997). A genomic perspective on protein families. *Science*, 278:631–637.

Tatusov, R. L., Natale, D. A., Garkavtsev, I. V., Tatusova, T. A., Shankavaram, U. T., Rao, B. S., Kiryutin, B., Galperin, M. Y., Fedorova, N. D., and Koonin, E. V. (2001). The COG database: new developments in phylogenetic classification of proteins from complete genomes. *Nucleic Acids Research*, 29(1):22–28.

Tucker, A. (2002). *Applied Combinatorics*. John Wiley & Sons.

Uno, T. and Tagiura, M. (2000). Fast algorithms to enumerate all common intervals of two permutations. *Algorithmica*, 26(2):290–309.

A Statistical Significance

In this section, our objective is to develop a test to assess the statistical significance of identified clusters against null hypothesis of random gene order. This is an approach that was recently considered by Durand and Sankoff (Durand and Sankoff, 2003), based upon a differing model for conserved gene clusters. In their paper, k genes are considered to be a cluster if they are localized within a window of size r. Our formulation of gene clusters as δ-teams requires a different approach.

The problem we consider is, if we observe k genes clustered in two genomes, what is the probability that this occurs by chance alone? In general, this probability depends on how this cluster was found, e.g., by reference region, window sampling or whole-genome comparison (Durand and

Sankoff, 2003). We assume a pessimistic scenario in our analysis to avoid overestimation of the significance of clusters. Specifically, we examine a region of one genome and select a k-gene cluster from this region (called reference region in (Durand and Sankoff, 2003)). We consider the probability that this set of genes will also occur as a cluster in the second genome, if the order of the second genome is uniformly random. Suppose we have k given genes in a genome of size n, and the genes of this genome are given but their order is random. The k genes are considered to witness a team if the distance of any neighboring two genes of this set is less than a given threshold. In our following analysis, the distance is measured by the number of genes inserted between these two (as opposed to the number of base pairs). Let this distance threshold be d. Hence $d = 1$ means a single insertion is allowed.

To begin, we consider a special case where there is only one-to-one homologous relationship, as was the case in the original gene team model. We consider the probability that a set of k distinct given genes form a team, with d insertions allowed between any two adjacent genes of this set, in the genome of n distinct genes. There are clearly $n!$ ways of permuting the genome. To count the number of ways of creating a genome with the specified k genes as a cluster, we divide this into three steps. First, we generate a k-cluster. There are $k!$ ways of permuting the k genes. In addition, we need to consider the different ways of adding spaces to the cluster because insertion is allowed. Let the number of spaces within this cluster be i, then $0 \leq i \leq (k-1)d$, and the number of ways of generating a k-cluster with i spaces be $s(k, d, i)$. The problem of finding $s(k, d, i)$ is equivalent to the following well-known problem in combinatorics: find the number of integer solutions of the equation:

$$x_1 + x_2 + \ldots + x_{k-1} = i \qquad 0 \leq x_j \leq d, \forall j \in \{1, 2, \ldots k-1\}$$

This problem can be solved by finding the coefficient of x^i of the polynomial $(1 + x + x^2 + x^3 \ldots + x^d)^{k-1}$. The details can be found in standard combinatorics texts, for instance, see (Tucker, 2002). Since there is no simple closed formula for $s(k, d, i)$, we will assume this function has already been solved. Second, we place this cluster as a whole to the n slots in the genome. The size of the cluster is $k + i$, hence there are equivalently $n - (k + i) + 1$ slots in the genome to place this cluster. Last, we place all the rest of genes on the rest of slots. There are $(n - k)!$ ways to do so. In summary, the probability of forming a k-cluster in a genome of size n is:

$$P(n, k, d) = \frac{(n-k)!k! \sum_{i=0}^{(k-1)d} s(k, d, i) \cdot (n - k - i + 1)}{n!} = \frac{\sum_{i=0}^{(k-1)d} s(k, d, i) \cdot (n - k - i + 1)}{\binom{n}{k}} \quad (1)$$

Next, we consider the general homologous relationship, as is the case in our generalized gene-team model. Suppose there are also m families in the genome, which we will denote as $M = \{f_1, f_2, \ldots, f_m\}$. Each gene must belong to one family. Let ϕ_j be the number of member genes that the family f_j has. We also assume that the k given genes belong to distinct families to simplify the mathematical analysis. This is justified in most cases since the paralogous genes seldom neighbor with each another. The cluster is now defined differently. If the underlying families of the given k genes form a cluster in the genome, then we will consider these k genes as a cluster. Let the index of the family that the j-th gene belongs to be i_j. The only difference between the current problem and the previous one is that we allow more ways of generating a cluster. In other words, for the j-th gene, we have now ϕ_{i_j} possible selections from the family f_{i_j}, instead of one. Once we have selected k genes from k families, the previous procedure can be applied without change to the remaining steps. Since there are $\phi_{i_1} \cdot \phi_{i_2} \ldots \phi_{i_k}$ ways of selecting k genes, the new probability is now:

$$Q(M, n, k, d) = P(n, k, d) \cdot \prod_{j=1}^{k} \phi_{i_j} \tag{2}$$

We try to give an approximation of this probability in practice. Generally, the cluster size would be less than 10, and we also suppose that a reasonably small δ is chosen, say 1 to 3. The number of genes in a genome, n, tends to be one to several thousand, which is much larger than k, d and i, with $0 \leq i \leq (k-1)d$. Therefore the number of ways of placing a k-cluster with i spaces in n slots is roughly independent of i. The total number of ways of adding spaces to k genes is $(d+1)^{k-1}$ since there are $(d+1)$ choices of adding spaces between any two adjacent genes (0-space, 1-space, up to d-space). So the numerator of $P(n, k, d)$ in Equation 1 is approximately $(n-k+1) \cdot (d+1)^{k-1}$. Then we have:

$$P(n, k, d) \approx \frac{(n-k+1) \cdot (d+1)^{k-1}}{\binom{n}{k}} \tag{3}$$

For *E. coli* genome, $n \approx 2000$ (note that we only consider the genes that have orthologs in both genomes). Our choice of $\delta = 1900$bp in Section 5, is roughly equivalent to $d = 1$. Since the majority of COGs have one to three paralogs in *E. coli*, we can estimate the product term in Equation 2 as 2^k. Then by Equation 2, we have: $k = 2, Q(2) \approx 8 \cdot 10^{-3}$; $k = 3, Q(3) \approx 4.8 \cdot 10^{-5}$; $k = 4, Q(4) \approx 3.8 \cdot 10^{-7}$. It can been seen that the identified teams of size greater or equal to 3 should be statistically significant. However, the probability given by Equation 2 depends on the content of a specific cluster. Since the number of genes of families can vary greatly, the actual probabilities can differ significantly from these estimates. In our analysis, we use the estimate of $P(n, k, d)$ for all clusters, but calculate the probability for each individual cluster.