Using PQ Trees for Comparative Genomics

Gad M. Landau[⋆,1,2], Laxmi Parida[3], and Oren Weimann[⋆⋆,1]

[1] Department of Computer Science, University of Haifa,
Mount Carmel, Haifa 31905, Israel
landau@cs.haifa.ac.il, oweimann@cs.haifa.ac.il
[2] Department of Computer and Information Science, Polytechnic University,
Six MetroTech Center, Brooklyn, NY 11201-3840, USA
landau@poly.edu
[3] Computational Biology Center, IBM TJ Watson Research Center,
Yorktown Heights, New York 10598, USA
parida@us.ibm.com

Abstract. Permutations on strings representing gene clusters on genomes have been studied earlier in [18, 14, 3, 12, 17] and the idea of a maximal permutation pattern was introduced in [12]. In this paper, we present a new tool for representation and detection of gene clusters in multiple genomes, using PQ trees [6]: this describes the inner structure and the relations between clusters succinctly, aids in filtering meaningful from apparently meaningless clusters and also gives a natural and meaningful way of visualizing complex clusters. We identify a minimal consensus PQ tree and prove that it is equivalent to a maximal πpattern [12] and each subgraph of the PQ tree corresponds to a non-maximal permutation pattern. We present a general scheme to handle multiplicity in permutations and also give a linear time algorithm to construct the minimal consensus PQ tree. Further, we demonstrate the results on whole genome data sets. In our analysis of the whole genomes of human and rat we found about 1.5 million common gene clusters but only about 500 minimal consensus PQ trees, and, with *E Coli K-12* and *B Subtilis* genomes we found only about 450 minimal consensus PQ trees out of about 15,000 gene clusters. Further, we show specific instances of functionally related genes in the two cases.

Key Words: Pattern discovery, data mining, clusters, patterns, motifs, permutation patterns, PQ trees, comparative genomics, whole genome analysis, evolutionary analysis.

1 Introduction

Given two permutations of n distinct characters, Uno and Yagiura [18] defined a *common interval* to be a pair of intervals of these permutations consisting

⋆ Partially supported by NSF grant CCR-0104307, by the Israel Science Foundation grant 282/01, and by IBM Faculty Partnership Award.
⋆⋆ Partially supported by the Israel Science Foundation grant 282/01.

of the same set of characters, and devised an $\mathcal{O}(n + K)$ time algorithm for extracting all common intervals of two permutations, where $K(\leq \binom{n}{2})$ is the number of common intervals. Heber and Stoye [14] extended this result to k sequences and presented an $\mathcal{O}(nk+K)$ time algorithm for extracting all common intervals of k permutations. The characters here represent genes and the string of characters represent the genome. Bergeron, Corteel and Raffinot [3] relaxed the "consecutive" constraint by introducing *gene teams* - allowing genes in a cluster to be separated by gaps that do not exceed a fixed threshold, and presented an $\mathcal{O}(kn \log^2 n)$ time algorithm for finding all gene teams.

A common technique of deciding whether two genes are similar is using the biological concept of orthologs and paralogs. Two genes are matched if they are either orthologous (appear in different organisms, but have the same evolutionary origin and are generated during speciation) or paralogous (appear in the same organism and caused by the duplication of ancestor genes). A slightly modified model of a genome sequence, that allows paralogs, was introduced by Schmidt and Stoye [17]. They extended the previous model of [14] by representing genomes as sequences rather then permutations, and devised a $\Theta(n^2)$ algorithm for extracting all common intervals of two sequences. He and Goldwasser [13] extended the notion of gene teams [3] to COG teams by allowing any number of paralogs and orthologs, and devised an $\mathcal{O}(mn)$ time algorithm to find such COG teams for *pairwise* chromosome comparison (where m and n are the number of orthologous genes in the two chromosomes).

In [12], pattern discovery was formalized as the π*pattern* problem. Let the pattern P=$p_1, p_2, \ldots, p_m$ and the string S=$s_1, s_2, \ldots, s_n$ be both sequences of characters (with possible repeats) over a given alphabet Σ (in our case genes). P *appears* in location i in S iff $(p_1, p_2, \ldots, p_m)$ is a permutation of $(s_i, \ldots, s_{i+m-1})$. P is a πpattern if it appears at least K times in S for a given K. A notation for *maximal* π*patterns* was introduced as a model to filter meaningful from apparently meaningless clusters. A πpattern p_1 is non-maximal with respect to πpattern p_2, if each occurrence of p_1 is covered by an occurrence of p_2 and each occurrence of p_2 covers an occurrence of p_1. The algorithm presented in [12] works in two stages: In stage 1, all πpatterns of sizes $\leq L$ are found in $\mathcal{O}(Ln \log |\Sigma| \log n)$ time, where n is the total length of all the sequences. For every πpattern found the algorithm stores a list of all the locations where the pattern appears, i.e, *location list*. In stage 2, a straightforward comparison of every two location lists is used to extract the maximal πpatterns out of all the πpatterns found in stage 1. Assume stage 1 outputs p πpatterns, and the maximum length of a location list is ℓ, stage 2 runs in $\mathcal{O}(p^2\ell)$ time. Integrating the two stages to produce only the maximal πpatterns was introduced as an open problem.

The common approach in practice is to output all the found patterns as sets of genes. This approach provides no knowledge of the ordering of the genes in each appearance of the pattern, and also outputs meaningless clusters. In this paper we use the PQ tree data structure [6] to devise a new tool for obtaining

the *maximal notation* of the appearances of a pattern in linear time. A formal definition of the maximal notation is given in [12]:

Definition 1. ***Maximal notation****: Given k permutations on an alphabet Σ, representing k occurrences of a pattern. The maximal notation is obtained by using a '-' between two groups of one or more genes to denote that these groups appear as immediate neighbors in all the k permutations, and using a ',' otherwise.*

Example. Consider the pattern $\{a, b, c, d, e, f\}$ appearing once as $abcdef$ and once as $bdacfe$, then the maximal notation of this pattern is $((a, b, c, d)-(e-f))$.

There are two main reasons for obtaining the maximal notation: (1). This notation provides knowledge of the inner structure of a pattern. $((a, b, c, d) - (e-f))$ shows that e appears always adjacent to f, and they both appear always adjacent to the group $\{a, b, c, d\}$. (2) This notation provides knowledge of the non-maximal relations between patterns, and can be used to filter meaningful from apparently meaningless clusters (non-maximal). $((a, b, c, d)-(e-f))$ shows that the patterns $\pi_1 = \{e, f\}$ and $\pi_2 = \{a, b, c, d\}$ are non-maximal w.r.t the pattern $\pi_3 = \{a, b, c, d, e, f\}$. Thus $((a, b, c, d)-(e-f))$ holds all the information of patterns π_1, π_2 and π_3.

Results. Our main theoretical results are: (a) we prove that a minimal consensus PQ tree is equivalent to a maximal πpattern [12] and each subgraph of the PQ tree corresponds to a non-maximal permutation pattern, and (b) give an algorithm that obtains the maximal notation of a πpattern p in $\mathcal{O}(nk)$ time, where k is the number of appearances of p and n is the number of characters in p (we assume all the characters in p are distinct. In section 5 we suggest a solution for patterns containing repeats of characters). We present several uses of this algorithm: (1) In the genome model that allows only orthologous genes (all k sequences are permutations of $\{1, 2, \ldots, n\}$), we can use this algorithm to obtain the maximal notation of the entire πpattern $\{1, 2, \ldots, n\}$. (2) In the most general genome model that allows orthologous and paralogous genes (a gene may appear any number of times in a sequence, and may appear in only some of the sequences) we assume some other gene clustering algorithm found the πpattern p. We use our tool to obtain the maximal notation of p. (3) We modify this algorithm to an $\mathcal{O}(nk^2)$ time algorithm that finds all maximal πpatterns in the genome model that allows orthologous genes as well as genes that do not appear in all the sequences. We present experimental results for the various types of data (Section 6). Our main practical results are the use of the tool on whole genome data sets: (1) human and rat genomes and (2) *E Coli K-12* and *B Subtilis* genomes. In both we show that the PQ trees help reduce the number of clusters to be analyzed as well as help in visualizing the internal structures of the clusters. We also hypothesize the function of an unknown gene in *E Coli* based on the permutation pattern.

Roadmap. We begin with an introduction to the PQ tree data structure in Section 2 and show that the minimal consensus PQ tree is indeed the maximal notation of a permutation pattern in Section 3. We present an $\mathcal{O}(kn)$ time algorithm to obtain the minimal consensus PQ tree in Section 4 and present

variations of this algorithm for the different genome models in Section 5. We conclude with experimental results in Section 6. The proofs of the theorems and lemmas are omitted and will appear in the full version of this paper.

2 Preliminaries - PQ Tree

In this section we present the PQ tree data structure and definitions as introduced by Booth and Leuker [6], as a tool to solve the general consecutive arrangement problem. The general consecutive arrangement problem is the following: *Given a finite set X and a collection $\mathcal{I}$ of subsets of X, does there exist a permutation π of X in which the members of each subset $I \in \mathcal{I}$ appear as a consecutive substring of π?* Booth and Leuker introduced an efficient algorithm (linear in the length of the input, $\mathcal{O}(n^2)$ in our terms) that solves this problem using a PQ tree. A PQ tree is a rooted tree whose internal nodes are of two types: P and Q. The children of a P-node occur in no particular order while those of a Q-node appear in a left to right or right to left order. We designate a P-node by a circle and a Q-node by a rectangle. The leaves of T are labeled bijectively by the elements of X. The *frontier* of a tree T, denoted by $F(T)$, is the permutation of X obtained by reading the labels of the leaves from left to right.

Definition 2. ***Equivalent PQ trees****: Two PQ trees T and T' are equivalent, denoted $T \equiv T'$, if one can be obtained from the other by applying a sequence of the following transformation rules: (1) Arbitrarily permute the children of a P-node, and (2) Reverse the children of a Q-node.*

Any frontier obtainable from a tree equivalent with T is considered *consistent* with T, and $\mathcal{C}(T)$ is defined as follows: $\mathcal{C}(T) = \{F(T')|T' \equiv T\}$. These last definitions are illustrated in Figure 1. We accordingly define the number of frontiers obtainable from a tree equivalent with T to be $|\mathcal{C}(T)|$. Clearly the equiva-

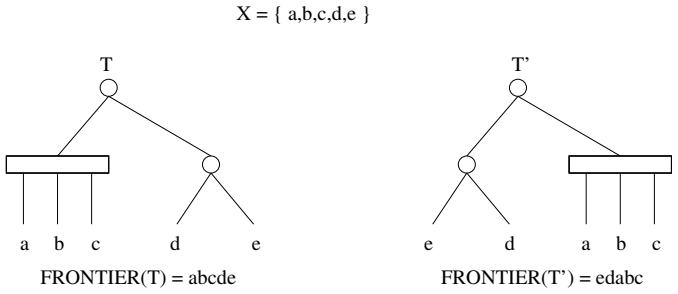


Fig. 1. Two equivalent PQ trees, $T' \equiv T$ and their frontiers. Note that $\mathcal{C}(T)$=$\mathcal{C}(T')$=$\{abcde, abced, cbade, cbaed, deabc, decba, edabc, edcba\}$.

lence relation is reflexive, symmetric and transitive. To make it computationally

straightforward, we use a slightly stricter version of a PQ tree called the *canonical PQ tree*.

Definition 3. ***Canonical PQ tree****: A PQ tree that has no node with only one child and no P node with only two children.*

Note that it is straightforward to convert any PQ tree to its canonical form: a node with a single child is merged with its immediate predecessor. This process is continued until no such node remains. Further, any P node with exactly two children is changed to a Q node. Through the rest of the paper, we assume a PQ tree is a canonical PQ tree. Some PQ trees are given special names: Given a finite set X, the PQ tree consisting of a single P node with $|X|$ children that are all leaves is called the *universal PQ tree*. We denote the universal tree as T_U. Another important PQ tree is the *null tree*, which has no nodes at all. By convention the null tree has no frontier and it's set of consistent permutations is empty. The most important contribution of [6] is an efficient algorithm for the *REDUCE*() function defined below.

Definition 4. ***REDUCE($\mathcal{I}$,T')****: Given a collection $\mathcal{I}$ of subsets of $N = \{1, 2, \ldots, n\}$ and a PQ tree T' whose leaves are $\{1, 2, \ldots, n\}$, the function* REDUCE*($\mathcal{I}$,T') builds a PQ tree T such that $f \in \mathcal{C}(T)$ iff $f \in \mathcal{C}(T')$ and every $i \in \mathcal{I}$ appears as a consecutive substring of f.*

The procedure *REDUCE*($\mathcal{I}$,T') will return the null tree if no frontier $f \in \mathcal{C}(T')$ is such that every $i \in \mathcal{I}$ appears as a consecutive substring of f. Note that if T_U is the universal PQ tree, then *REDUCE*($\mathcal{I}$,T_U) builds a PQ tree T such that $f \in \mathcal{C}(T)$ iff every $i \in \mathcal{I}$ appears as a consecutive substring of f. By [6], if the number of subsets in $\mathcal{I} \leq n$, as in our case (see Section 4), then the time complexity of *REDUCE*($\mathcal{I}$,T_U) is $\mathcal{O}(n^2)$. In Section 4.1 we present an $\mathcal{O}(n)$ time complexity algorithm for the *REDUCE* function when $\mathcal{I}$ is a set of at most n intervals, based on a data structure presented in [14].

The following observation is immediate from the definition of the *REDUCE*() function. Informally, it says that if T is the PQ tree returned by *REDUCE*($\mathcal{I}$,T_U) then if we add more subsets of N to $\mathcal{I}$ then $|\mathcal{C}(T)|$ gets smaller.

Observation 1. *Given two collections $\mathcal{I}_1$, $\mathcal{I}_2$ of subsets of N, if $\mathcal{I}_1 \subseteq \mathcal{I}_2$ and T_1=*REDUCE*($\mathcal{I}_1$,T_U) and T_2=*REDUCE*($\mathcal{I}_2$,T_U) then $\mathcal{C}(T_2) \subseteq \mathcal{C}(T_1)$.*

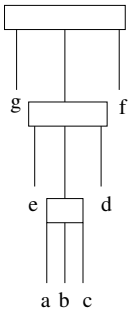
3 The Minimal Consensus PQ Tree

In this section we define a minimal consensus PQ tree as a representation of the maximal notation of the k occurrences of a πpattern. Through the rest of this paper we define Π to be a set of k permutations $\pi_1, \pi_2, \ldots, \pi_k$ representing k occurrences of a πpattern $N = \{1, 2, \ldots, n\}$.

Definition 5. ***Notation of a PQ tree****: The notation of a PQ tree is obtained by writing the PQ tree as a parenthesized string with different symbols encoding P (comma separators) and Q (dash separators) nodes.*

For example in Figure 1, T is denoted as $((a-b-c),(d,e))$ and the PQ tree in Figure 2 is denoted as $(g-(e-(a-b-c)-d)-f)$. Given Π, our main goal is to construct a PQ tree T from Π, such that the notation of T is the maximal notation of Π. We would like to construct a PQ tree T such that $\mathcal{C}(T) = \{\pi_1, \pi_2, \ldots, \pi_k\}$ however, this is not always possible. Consider a πpattern $\{a, b, c, d, e\}$ appearing four times as $abcde, abced, cbade, edabc$, the PQ tree T in Figure 1 is the one that best describes these appearances. However, $edcba \in \mathcal{C}(T)$ although the πpattern never appeared as $edcba$. On the other hand, notice that the universal PQ tree over Σ, T_U ,is such that $\{\pi_1, \pi_2, \ldots, \pi_k\} \subseteq \mathcal{C}(T_U)$. Hence the idea of *minimal* is introduced. In section 3.2 we suggest a way of reducing the redundant frontiers from the minimal consensus PQ tree. We next present a way of relating a set of permutations to a PQ tree.

Definition 6. ***minimal consensus PQ tree****: Given Π, A consensus PQ tree T of Π is such that $\Pi \subseteq \mathcal{C}(T)$ and the consensus PQ tree is minimal when there exists no $T' \not\equiv T$ such that $\Pi \subseteq \mathcal{C}(T')$ and $|\mathcal{C}(T')| < |\mathcal{C}(T)|$.*

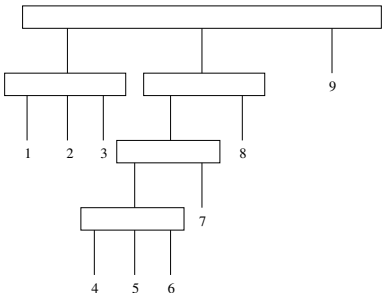


(1) Consensus Trees: If T_U is the universal PQ tree, note that $\pi_1, \pi_2 \in \mathcal{C}(T) \subseteq \mathcal{C}(T_U)$. Thus the consensus of two strings need not give rise to a unique PQ tree.
(2) PQ tree expression power: Consider $\pi_3 = gdcbaef$, clearly $\pi_3 \neq \pi_1$ and $\pi_3 \neq \pi_2$, however $\pi_3 \in \mathcal{C}(T)$.
(3) Height of the PQ tree: A consensus PQ tree of only two strings (permutations) of length L can possibly be of height $\mathcal{O}(L)$.

Fig. 2. Let $\pi_1 = geabcdf$ and $\pi_2 = fecbadg$. The PQ tree T in the figure is a minimal consensus PQ tree of $\{\pi_1, \pi_2\}$. We use this example to illustrate three different ideas as shown on the right.

Figure 2 illustrates the motivation for the definition of minimal consensus. By defining the minimal consensus PQ tree, the problem now is to devise a method to construct the minimal consensus PQ tree given Π. Later we show that the notation of the minimal consensus PQ tree of Π is the maximal notation of Π. We use the following definition from [18]:

Definition 7. ***Common Interval (C_Π)****: Given Π, w.l.o.g we assume that $\pi_1 = id_n := (1, 2, \ldots, n)$. An interval $[i, j]$ $(1 \leq i < j \leq n)$ is called a common interval of Π iff the elements of the set $\{i, i+1, \ldots, j\}$ appear as a consecutive substring in every $\pi_i \in \Pi$ $(i = 1, 2, \ldots, k)$. The set of all common intervals of Π is denoted C_Π.*



The tree on the left is the minimal consensus PQ tree of $\Pi = \{\pi_1, \pi_2, \pi_3\}$ where $\pi_1 = (1, 2, 3, 4, 5, 6, 7, 8, 9)$, π_2=(9,8,4,5,6,7,1,2,3), and π_3=(1,2,3,8,7,4,5,6,9).
C_Π = {[1,2],[1,3],[1,8],[1,9],[2,3],[4,5],[4,6], [4,7],[4,8],[4,9],[5,6]}. The maximal pattern whose three occurrences are given by π_1, π_2, π_3 is ((1-2-3)-(((4-5-6)-7)-8)-9).

Fig. 3. Maximal notation of a πpattern and the corresponding minimal consensus PQ tree.

See Figure 3 for an example of common intervals and a minimal consensus PQ tree. Next we state some theorems leading up to the uniqueness of a minimal consensus tree.

Theorem 2. *Given Π, T_C=REDUCE(C_Π,T_U) is a minimal consensus PQ tree of Π.*

The following corollary is immediate from the proof of Theorem 2.

Corollary 1. *If T_1 and T_2 are two minimal consensus PQ trees of Π, then $\mathcal{C}(T_1) = \mathcal{C}(T_2)$.*

Theorem 3. *For two PQ trees T_1 and T_2', if $\mathcal{C}(T_1) = \mathcal{C}(T_2')$, then $T_1 \equiv T_2'$.*

The following lemma is straightforward to verify.

Lemma 1. *Given Π, the minimal consensus PQ tree T of Π is unique (up to equivalence).*

The minimal consensus PQ tree is not necessarily unique when a character can appear more than once in a πpattern. We handle this problem in Section 5.

3.1 Identifying Maximal πpatterns in the Minimal Consensus PQ Tree

In this section we describe a PQ subtree as a method for identifying non-maximal permutation patterns, and we make the simplifying assumption that there are no multiplicities in the πpatterns, this problem is addressed in Section 5.

Definition 8. ***PQ subtree**: Given a PQ tree T, the variant v' of a node v is defined as follows: (1) If v is a P node then it's only variant v' is the P node itself. (2) If v is a Q node with k children, then a variant v' of v is a Q node with any $k' \leq k$ consecutive children of v. A PQ subtree is rooted at a variant v' of node v and includes all its descendants in the PQ tree T.*

Let $L(v')$ denote the set of the labels of the leafnodes reachable from v'. Further, given the leafnode labels $p = \{\alpha_1, \alpha_2, \ldots, \alpha_n\}$, the least common ancestor

(LCA) of p is that variant v' of a node v satisfying the following: (1) $p \subseteq L(v')$ and (2) there exists no variant v'' of v or any other node such that $p \subseteq L(v'')$ and $|L(v'')| < |L(v')|$.

Recall that a πpattern p_1 is non-maximal with respect to πpattern p_2, if each occurrence of p_1 is covered by an occurrence of p_2 and each occurrence of p_2 covers an occurrence of p_1 (notice that $p_1 \subseteq p_2$). Through the rest of this section we assume $p_1 = \{\alpha_1, \alpha_2, \ldots, \alpha_n\}$ and p_2 are πpatterns such that p_1 is non-maximal with respect to p_2. We denote T_i as the minimal consensus PQ tree of the appearances of πpattern p_i, and C_{Π_i} as the set of all common intervals of the occurrences of p_i. The following definition will aid in describing the connection between PQ subtrees and non-maximal πpatterns (Theorem 4).

Definition 9. $\boldsymbol{T_i^{p_j}}$*: Given a PQ tree T_i, and $p_j = \{\alpha_1, \alpha_2, \ldots, \alpha_n\}$, let v' be the LCA of p_j in T_i. Then $T_i^{p_j}$ is the PQ subtree rooted at v'.*

Theorem 4. *Given πpatterns p_1, p_2 on some S, if p_1 is non-maximal w.r.t p_2 then $T_2^{p_1} \equiv T_1$.*

Notice that the converse of Theorem 4 is true only if every occurrence of p_1 is covered by an occurrence of p_2. The following theorem proves that given Π, the problem of obtaining the maximal notation of Π is equivalent to the problem of constructing the minimal consensus PQ tree of Π.

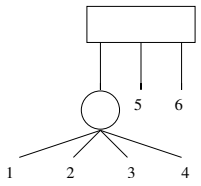
Theorem 5. *The notation of the minimal consensus PQ tree of a πpattern is the maximal notation of the πpattern.*

3.2 Specializing the PQ Tree

Given Π, we would like to construct a PQ tree T such that $\mathcal{C}(T) = \Pi$. However, as shown earlier this is not always possible using a PQ tree. This requires more precise definitions of the P and the Q node. Adding restrictions to the PQ tree will help solve the problem. We suggest the following: (1) Assigning a bi-directional annotation to the Q node as $\Leftrightarrow$ only when the children appear in both directions in the strings and un-annotated otherwise. (2) The exact permutations appearing in the strings for the P node. For example if a P node has 7 children and the annotation is (3162574,5142736), then this implies that the P node has three possible permutations on it's children as 1234567, 3162574 and 5142736. Note that the children are not necessarily leaf nodes. See Figure 4 for an example. The advantage of this is that the PQ tree remains the same and the annotations simply help remove the extra frontiers, $\mathcal{C}(T) \setminus \Pi$, where T is the un-annotated PQ tree.

4 Constructing a Minimal Consensus PQ Tree in $\mathcal{O}(kn)$ Time

In this section we devise new algorithms for computing the minimal consensus PQ tree. The first algorithm runs in $\mathcal{O}(kn + n^2)$ time. We then improve this algorithm to $\mathcal{O}(kn)$ time, which is optimal since the length of the input is kn.



Consider $\Pi = \{123456, 241356\}$. The PQ tree on the left is the minimal consensus PQ tree of Π. Consider the following restrictions: (1) The Q node is un-annotated (2) The exact permutations of the P node is (2413). The restricted PQ tree is such that $\mathcal{C}(T) = \Pi$.

Fig. 4. The restricted PQ tree.

We first find a subset of C_Π of size $\mathcal{O}(n)$ that holds sufficient information about the k permutations. For example, consider the πpattern $\{1, 2, 3\}$ appearing twice as $\Pi = \{123, 321\}$, then $C_\Pi = \{[1, 2], [2, 3], [1, 3]\}$. In Theorem 2 we proved that the minimal consensus PQ tree T is such that in every $f \in \mathcal{C}(T)$ the sets $\{1, 2\}$, $\{2, 3\}$ and $\{1, 2, 3\}$ appear as a consecutive substring. Notice that the common interval $[1, 3]$ is redundant in the sense that if the sets $\{1, 2\}$ and $\{2, 3\}$ appear as a consecutive substring in every $f \in \mathcal{C}(T)$, then $\{1, 2, 3\}$ must also appear as a consecutive substring in every $f \in \mathcal{C}(T)$. The common interval $[1, 3]$, which is the union of $[1, 2]$ and $[2, 3]$, is therefore not necessary for constructing T. We next show that the set of common intervals that are necessary for constructing T is the set of *irreducible intervals* as defined in [14]: Given Π, without loss of generality, we assume that $\pi_1 = id_n := (1, 2, \ldots, n)$. Two common intervals $c_1, c_2 \in C_\Pi$ have a non-trivial overlap if $c_1 \cap c_2 \neq \emptyset$ and they do not include each other. A list $p = (c_1, c_2, \ldots, c_{\ell_{(p)}})$ of common intervals $c_1, c_2, \ldots, c_{\ell_{(p)}} \in C_\Pi$ is a chain (of length $\ell_{(p)}$) if every two successive intervals in p have a non-trivial overlap. A chain of length one is called a trivial chain. A common interval I is called *reducible* if there is a non-trivial chain that generates it (I is the union of all elements in all the intervals of the chain), otherwise it is called *irreducible*. This partitions the set of common intervals C_Π into the set of reducible intervals and the set of irreducible intervals, denoted I_Π. Obviously, $1 \leq |I_\Pi| \leq |C_\Pi| \leq \binom{n}{2}$ [14]. The set of irreducible common intervals of Π from the example in Figure 3 is: $I_\Pi = \{[1, 2], [1, 8], [2, 3], [4, 5], [4, 7], [4, 8], [4, 9], [5, 6]\}$ and their chains are illustrated in Figure 5.

The algorithm. The following algorithm takes advantage of the fact that the irreducible intervals hold as much information as C_Π.

Algorithm PQ-Construct:
Input: Π.
Output: The minimal consensus PQ tree T of Π.

1. Compute I_Π using the algorithm described in [14].
2. Compute T=REDUCE(I_Π,T_U) using the algorithm described in [6].
3. Return T.

In Theorem 2 we proved that T_C=REDUCE(C_Π,T_U) is the minimal consensus PQ tree of Π. In the following lemma we prove that if T_I=REDUCE(I_Π,T_U) then T_C=T_I, thus proving the correctness of the algorithm.

Lemma 2. *Given Π, T_I=REDUCE(I_Π,T_U) is the minimal consensus PQ tree of Π.*

Time complexity of the algorithm. Given k permutations each of length n, by [14], $|I_\Pi| < n$ and further, I_Π can be computed in $\mathcal{O}(kn)$ time. By [6], computing T=REDUCE(I_Π,T_U) takes $\mathcal{O}(n^2)$ time. The minimal consensus PQ tree can therefore be computed in $\mathcal{O}(kn + n^2)$ time.

4.1 Computing REDUCE(I_Π,T_U) in linear time

In this section we modify step 2 of the PQ-Construct algorithm to run in $\mathcal{O}(n)$ time. In [14] a data structure $\mathcal{S}$ was used to obtain the irreducible intervals. For each chain of non-trivially overlapping irreducible intervals, $\mathcal{S}$ contains a doubly-linked list that holds the intervals of that chain in left-to-right order. Moreover, intervals from different lists with the same left or right end are connected by *vertical pointers* yielding for each index $x \in N$ a doubly-linked *vertical list*. The final data structure $\mathcal{S}$ of the example in Figure 3 is shown in Figure 5. We next describe a new algorithm called *REPLACE* that transform $\mathcal{S}$ to the

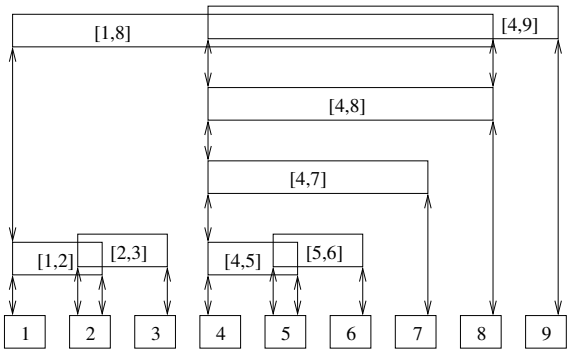


Fig. 5. Sketch of the data structure $\mathcal{S}$ of the example in Figure 3. The chains of non-trivially overlapping irreducible intervals are: $([1,2],[2,3])$, $([4,5],[5,6])$ and $([1,8],[4,9])$

minimal consensus PQ tree. The general idea is to replace every chain by a Q node where the children of the Q node are the roots of subtrees with leaves induced by the intersection between the intervals of the chain. For example, in Figure 5 the chain $([1,8],[4,9])$ is replaced by a Q node of three children where each child is the root of a subtree containing the leaves $\{1,2,3\}$, $\{4,5,6,7,8\}$ and $\{9\}$ respectively. Then, every element that is not a leaf or a Q node and is

pointed by a vertical link is replaced with a P node. For example, in Figure 5 the vertical links from $[4,8]$ to $[4,7]$ and 8 implies that $[4,8]$ is replaced by a P node with two children where each child is the root of a subtree containing the leaves $\{4,5,6,7\}$ and $\{8\}$ respectively. Finally, a P node with 2 children is replaced by a Q node. The PQ tree obtained by REPLACE on $\mathcal{S}$ is illustrated in Figure 3. A similar idea, for the case of *conserved intervals* [2] (as opposed to irreducible intervals) is discussed in [2, 5, 4].

The following theorem proves that we obtain the minimal consensus PQ tree using REPLACE.

Theorem 6. *If T' is the PQ tree obtained from $\mathcal{S}$ by REPLACE and T=REDUCE(I_Π,T_U) then $T \equiv T'$.*

Time complexity of the algorithm. By [14], I_Π and $\mathcal{S}$ can be computed in $\mathcal{O}(kn)$ time. REPLACE can be performed by a simple bottom up traversal of $\mathcal{S}$, therefore in $\mathcal{O}(n)$ time. The minimal consensus PQ tree can therefore be computed in $\mathcal{O}(kn)$ time.

5 Algorithms for Various Genome Models

We next present three different genome models that use the PQ tree tool to detect and represent the maximal patterns as PQ trees. The first model allows only orthologous genes [18] (all the k sequences are permutations of the same n genes, thus, every gene appears exactly once in every sequence). The second model allows both orthologous and paralogous genes (a gene may appear any number of times in a sequence, and may appear in only some of the sequences). The third allows orthologous genes as well as genes that do not appear in all the sequences (a gene can appear at most once in a sequence).

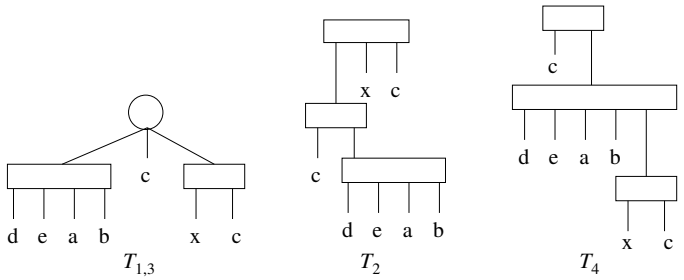
(1) Genomes as Permutations With no Multiplicity. This model is ideal for our tool. Since the k sequences are permutations of Σ with no multiplicity, Σ is a πpattern (common interval) of size n. Furthermore, it is the only maximal πpattern in the sequences. We construct the minimal consensus PQ tree, T, of the set of sequences and obtain the maximal notation of the only maximal πpattern in $\mathcal{O}(kn)$ time. Notice that by traversing T we can output all the πpatterns (common intervals) of the sequences (every subtree of T is a πpattern) in $\mathcal{O}(K)$ time, where $K(\leq \binom{n}{2})$ is the number of πpatterns. Therefore, in $\mathcal{O}(nk+K)$ time we can output all the non-maximal πpatterns, exactly like Heber and Stoye [14] do, but we also present the maximal notation of every πpattern found, and present the non-maximal relations between them. We introduce experimental results of human and rat whole genome comparison for this type of data in Section 6.

(2) Genomes as Strings With Multiplicity. In this case the input is a set of k sequences of n genes, where a gene can appear $K \geq 0$ times.

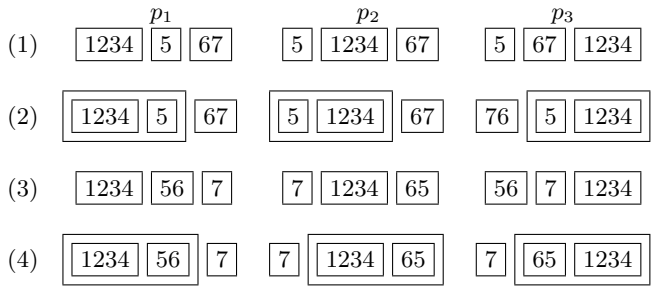
A string that has at least one character that appears more than once is termed as a string with *multiplicity*. For example if $p_1 = abcegd$ and $p_2 = acgcab$, then

p_1 has no multiplicity. However that is not the case with p_2 where a and c each appear more than once.

Consider a pattern p with occurrences as $acbdefc$ and $cdabfec$. Clearly p has a unique minimal consensus PQ tree corresponding to $acbdefc'$ and $cdabfec'$ and treating c' as a distinct character. However, the minimal consensus PQ tree is not necessarily unique when a character can appear more than once in a πpattern. This is illustrated in an example in Figure 6. We handle multiplicity by reporting the multiple minimal consensus PQ trees. This is explained using the example of Figure 6. Each character is labeled with a distinct integer in the reference sequence and the remaining sequences are treated as multi-sets (strings of sets of characters). In the example, $p_1 = deabcxc = 1234567$ and, $p_2 = cdeabxc = [57]12346[57]$ and $p_3 = cxcbaed = [57]6[57]4321$. If Π_1 and Π_2 are two choices such that $C_{\Pi_2} \subset C_{\Pi_1}$, then clearly the choice of Π_1 is made over Π_2. Continuing the example, the two choices for p_2 are (1) $p_2 = 5123467$, hence $p_3 = 5674321$ or $p_3 = 7654321$ so that $[6, 7] \in C_\Pi$ and (2) $p_2 = 7123465$, hence $p_3 = 5674321$ or $p_3 = 7654321$ so that $[5, 6] \in C_\Pi$. See Figure 6 for the corresponding PQ trees. We present an experimental result for this type of data, of a pairwise comparison



Let p_1=*deabcxc*, p_2=*cdeabxc* and p_3=*cxcbaed*. Then $p_1 = deabcxc = 1234567$ and, $p_2 = cdeabxc = [57]12346[57]$ and $p_3 = cxcbaed = [57]6[57]4321$. The two choices for p_2 are (1) $p_2 = 5123467$, hence $p_3 = 5674321$ or $p_3 = 7654321$ and (2) $p_2 = 7123465$, hence $p_3 = 5674321$ or $p_3 = 7654321$. Thus the four cases are

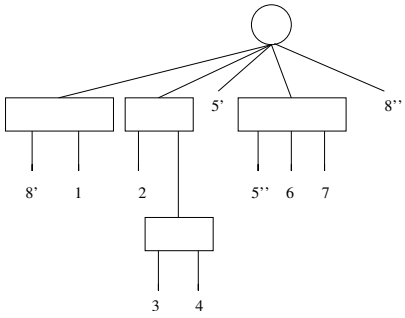


The trees above represent the four cases: $T_{1,3}$ represents the first and third cases, T_2 and T_4 represent the second and fourth cases respectfully. Notice that T_2 and T_4 are both minimal consensus PQ trees of $\{p_1, p_2, p_3\}$ since $|\mathcal{C}(T_2)| = |\mathcal{C}(T_4)| = 8$.

Fig. 6. πpatterns with multiplicity.

between the genomes of *E Coli K-12* and *B Subtilis* in Section 6. We used the algorithm described in [12] to find the πpatterns and our tool to present the maximal patterns as PQ trees (thus automatically filtering out the non-maximal patterns).

(3) Genomes as Strings With no Multiplicity. In this case the input is a set of k sequences of n genes, where a gene can appear $K \leq 1$ times. We present an $\mathcal{O}(nk^2)$ time algorithm that finds all maximal πpatterns in the sequences (notice that there can now be more then one maximal πpattern). The idea is to transform the sequences into permutations of the same set, and then build the minimal consensus PQ tree of these permutations. Consider the following example where there are two sequences, 1234567 and 1824376. First we go over the sequences and tag the genes that do not appear in all the sequences, we get $12345'67$ and $18'24376$. Then, for every tagged gene g' we replace it with $g'g''$ in the sequences where g' appears, and in the sequences where g' doesn't appear we add g' in the beginning of the sequence and g'' in the end of the sequence. After doing that we get $8'12345'5''678''$ and $5'18'8''243765''$. Now the tagged sequences are permutations of the same set, and furthermore, every πpattern that appeared in all the original sequences, appears in the tagged sequences, and every πpattern that appears in the tagged sequences but doesn't appear in the original sequences must contain a tagged element (this is achieved by splitting the tagged and double tagged elements). Next we construct the minimal consensus PQ tree T, of the tagged sequences. Notice that if a subtree T_i of T has no tagged leaves and there is no subtree T_j of T such that T_i is a subtree of T_j and T_j has no tagged leaves, then T_i represents a maximal πpattern. The minimal consensus PQ tree of the set of tagged sequences created from our example is shown in Figure 7.



Consider two sequences 1234567 and 1824376, after tagging and adding characters as explained we get the two permutations $8'12345'5''678''$ and $5'18'8''243765''$, their minimal consensus PQ tree is illustrated above. The subtrees that have no tagged elements (notated $(2-(3-4))$ and $(6-7)$) are the only maximal πpatterns of the original sequences.

Fig. 7. An example illustrating the use of tagged sequences.

Time complexity of the algorithm. There are initially k sequences of length n each. It takes $\mathcal{O}(nk^2)$ time to tag and add the elements as needed and we get k tagged sequences of the same length (which is at most $2nk$ if every gene appears in only one sequence, which rarely happens). The minimal consensus PQ

tree construction of the tagged sequences takes $\mathcal{O}(nk^2)$ time using the $\mathcal{O}(nk)$ algorithm presented in Section 4. Therefore, the algorithm takes $\mathcal{O}(nk^2)$ time.

We present an experimental result for this type of data, of a comparison between eight *chloroplast* genomes in the full version of this paper.

6 Experimental Results

Human and Rat genomes. In order to build a PQ tree for human and rat whole genome comparisons we used the output of a program called SLAM [1, 8, 7], SLAM is a comparative-based annotation and alignment tool for syntenic genomic sequences that performs gene finding and alignment simultaneously and predicts in both sequences symmetrically. When comparing two sequences, SLAM works as follows: Orthologous regions from the two genomes as specified by a homology map are used as input, and for each gene prediction made in the human genome there is a corresponding gene prediction in the rat genome with identical exon structure. We used the results from SLAM of comparing human (NCBI Build 31, November 2002) and rat (RGSC v2, November 2002) genomes, sorted by human chromosomes. The data in every chromosome is presented as a table containing columns: *Gene name, rat coords, human coords, rat coding length, human coding length* and *# Exons*.

There were 25,422 genes predicted by SLAM, each gene appears exactly once in each of the genomes. We mapped every one of the 25,422 genes to an integer, thus, the human genome becomes the identity permutation $(1, 2, 3, \ldots, 25422)$, and the rat genome becomes a permutation of $\{1, 2, 3, \ldots, 25422\}$ obtained from the SLAM output table. The full mapping can be found in our web page: http://crilx2.hevra.haifa.ac.il/∼orenw/MappingTable.ps.

Ignoring the trivial permutation pattern involving all the genes, there were only 504 interesting maximal ones out of 1,574,312 permutation patterns in this data set (we only consider patterns that do not cross chromosomes). In

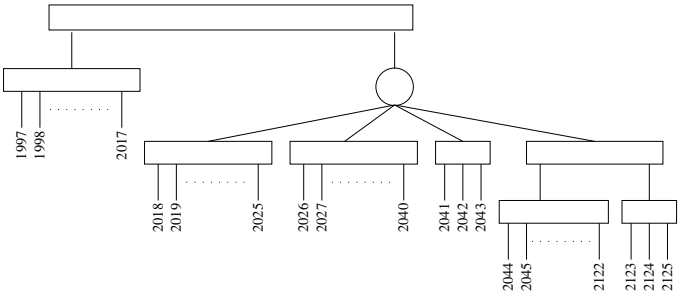


Fig. 8. A PQ subtree of the minimal consensus PQ tree of the human and rat orthologous genes, as predicted by SLAM.

Figure 8 we present a subtree of the Human-Rat whole genome PQ tree. This

tree corresponds to a section of 129 genes in human chromosome 1 and in rat chromosome 13. By our mapping, these genes appear in the human genome as the permutation: (1997-2125) and in the rat genome as the permutation: (2043-2041, 2025-2018, 2123-2125, 2122-2044, 2040-2026, 2017-1997). Figure 8 is the minimal consensus PQ tree of these two permutations.

Another subtree of the Human-Rat whole genome PQ tree, corresponding to a section of 156 genes in human chromosome 17 and in rat chromosome 10 is ((21028-21061)-(21019-21027)- (21018-20906)). The neighboring genes PMP22 and TEKTIN3 (corresponding to 21014 and 12015) are functionally related genes as explained in [10].

E Coli K-12 and B Subtilis genomes. Here we present a simple, yet interesting PQ tree obtained from a pairwise comparison between the genomes of *E Coli K-12* and *B Subtilis*. The input data was obtained from NCBI GenBank, in the form of the order of COGs (Clusters Of Orthologous Groups) and their location in each genome.
The data can be found in http://euler.slu.edu/∼goldwasser/cogteams/data as part of an experiment discussed by He and Goldwasser in [13], whose goal was to find COG teams. They extracted all clusters of genes appearing in both sequences, such that two genes are considered neighboring if the distance between their starting position on the chromosome (in bps) is smaller than a chosen parameter $\delta > 0$. One of their experimental results, for $\delta = 1900$ was the detection of a cluster of only two genes: COG0718, whose product is an uncharacterized protein conserved in bacteria, and COG0353, whose product is a recombinational DNA repair protein. They conjecture that the function of COG0353 might give some clues as to the function of COG0718 (which is undetermined).

In our experiment we built PQ trees of clusters of genes appearing in both sequences, such that two genes are considered neighboring if they are consecutive in the input data irrespective of the distance between them. There were 450 maximal permutation patterns out of 15,000 patterns discovered by our tool. Here we mention a particularly interesting cluster: (COG2812-COG0718-COG0353). The product of COG2812 is DNA polymerase III, which according to [9] is also related to DNA repair. The PQ tree clearly shows that COG0718, whose function is undetermined is located between two genes whose function is related to DNA repair. This observation further contributes to the conjecture that the function of COG0718 might be also related to DNA repair. Note that the reason that COG2812 was not clustered with COG0718 and COG0353 in [13] is because the distance between COG2812 and COG0718 is 1984 ($> \delta = 1900$).

Acknowledgments. We would like to thank Jens Stoye, Mathieu Raffinot and Ross McConnell for fruitful discussions.

References

1. M. Alexandersson, S. Cawley and L. Pachter. SLAM- Cross-species gene finding and alignment with a generalized pair hidden Markov model. In *Genome Research*,

13(3):496–502, 2003.
2. A. Bergeron, M. Blanchette, A. Chateau and C. Chauve. Reconstructing ancestral gene orders using conserved intervals. In *Proceedings of the Fourth Workshop on Algorithms in Bioinformatics (WABI)*, 14–25, 2004.
3. A. Bergeron, S. Corteel and M. Raffinot. The algorithmic of gene teams. In *Proceedings of the Second Workshop on Algorithms in Bioinformatics (WABI)*, 464–476, 2002.
4. A. Bergeron, J. Mixtacki and J. Stoye. Reversal Distance without Hurdles and Fortresses. In *Proceedings of the 15th Annual Symposium on Combinatorial Pattern Matching (CPM)*, 388–399, 2004.
5. A. Bergeron and J. Stoye. On the similarity of sets of permutations and its applications to genome comparison. In *Proceedings of the ninth Annual International Conference on Computing and Combinatorics (COCOON)*, 68–79, 2003.
6. K. Booth and G. Leuker. Testing for the consecutive ones property, interval graphs, and graph planarity using pq-tree algorithms. In *Journal of Computer and System Sciences*, 13:335–379, 1976.
7. N. Bray, O. Couronne, I. Dubchak, T. Ishkhanov, L. Pachter, A. Poliakov, E. Rubin and D. Ryaboy. Strategies and Tools for Whole-Genome Alignments. In *Genome Research*, 13(1):73–80, 2003.
8. N. Bray, I. Dubchak and L. Pachter. AVID: A Global Alignment Program. In *Genome Research*, 13(1):97–102, 2003.
9. S. K. Bryan, M. E. Hagensee and R. E. Moses. DNA Polymerase III Requirement for Repair of DNA Damage Caused by Methyl Methanesulfonate and Hydrogen Peroxide. In *Journal of Bacteriology*, 16(10):4608–4613, 1987.
10. K.H. Burns, M.M. Matzuk, A. Roy and W. Yan. Tektin3 encodes an evolutionarily conserved putative testicular micro tubules-related protein expressed preferentially in male germ cells. In *Molecular Reproduction and Development*, 67:295-302, 2004.
11. G. Didier. Common intervals of two sequences. In *Proceedings of the Third Workshop on Algorithms in Bioinformatics (WABI)*, 17–24, 2003.
12. R. Eres, L. Parida and G.M. Landau. A combinatorial approach to automatic discovery of cluster-patterns. In *Proceedings of the Third Workshop on Algorithms in Bioinformatics (WABI)*, *Lecture Notes in Bioinformatics*, 2812:139–150, 2003.
13. X. He and M.H. Goldwasser. Identifying conserved gene clusters in the presence of orthologous groups. In *Proceedings of the Eighth Annual International Conferences on Research in Computational Molecular Biology (RECOMB)*, 272–280, 2004.
14. S. Heber and J. Stoye. Finding all common intervals of k permutations. In *Proceedings of the 12th Annual Symposium on Combinatorial Pattern Matching (CPM)*, 207–218, 2001.
15. R.M. McConnell. A certifying algorithm for the consecutive-ones property. In *Proceedings of the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 15:761–770, 2004.
16. J. Mulley, P. Holland. Small genome, big insights. In *Nature*, 431:916-917, 2004.
17. T. Schmidt and J. Stoye. Quadratic time algorithms for finding common intervals in two and more sequences. In *Proceedings of the 15th Annual Symposium on Combinatorial Pattern Matching (CPM)*, 347–358, 2004.
18. T. Uno and M. Yagiura. Fast algorithms to enumerate all common intervals of two permutations. In *Algorithmica*, 26(2):290-309, 2000.