JOURNAL OF COMPUTATIONAL BIOLOGY
Volume 12, Number 10, 2005
© Mary Ann Liebert, Inc.
Pp. 1289–1306

Gene Proximity Analysis across Whole Genomes via PQ Trees[1]

GAD M. LANDAU,[2] LAXMI PARIDA,[3] and OREN WEIMANN[4]

Downloaded by Ben Gurion from online.liebertpub.com at 11/07/17. For personal use only.

ABSTRACT

Permutations on strings representing gene clusters on genomes have been studied earlier by Uno and Yagiura (2000), Heber and Stoye (2001), Bergeron *et al.* (2002), Eres *et al.* (2003), and Schmidt and Stoye (2004) and the idea of a maximal permutation pattern was introduced by Eres *et al.* (2003). In this paper, we present a new tool for representation and detection of gene clusters in multiple genomes, using PQ trees (Booth and Leuker, 1976): 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 π pattern (Eres *et al.*, 2003) and each subgraph of the PQ tree corresponds to a nonmaximal 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 datasets. 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, 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, and when comparing eight different *Chloroplast* genomes, we found only 77 minimal consensus PQ trees out of about 6,700 gene clusters. Further, we show specific instances of functionally related genes in two of the cases.

Key words: pattern discovery, 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 (2000) defined a *common interval* to be a pair of intervals of these permutations consisting of the same set of characters and devised an $\mathcal{O}(n + K)$ time algorithm for extracting all common intervals of two permutations, where

[1]A preliminary version of this paper appeared in Landau *et al.* (2005).

[2]Department of Computer Science, University of Haifa, Mount Carmel, Haifa 31905, Israel; and Department of Computer and Information Science, Polytechnic University, Six MetroTech Center, Brooklyn, NY 11201-3840.

[3]Computational Biology Center, IBM TJ Watson Research Center, Yorktown Heights, New York 10598.

[4]Department of Computer Science, University of Haifa, Mount Carmel, Haifa 31905, Israel.

Downloaded by Ben Gurion from online.liebertpub.com at 11/07/17. For personal use only.

$K \left(\leq \binom{n}{2}\right)$ is the number of common intervals. Heber and Stoye (2001) 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 *et al.* (2002) 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 are caused by the duplication of ancestor genes). A slightly modified model of a genome sequence that allows paralogs, was introduced by Schmidt and Stoye (2004). They extended the previous model of Heber and Stoye (2001) by representing genomes as sequences rather than permutations and devised a $\Theta(n^2)$ algorithm for extracting all common intervals of two sequences. He and Goldwasser (2004) extended the notion of gene teams (Bergeron *et al.*, 2002) 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 Eres *et al.* (2003), pattern discovery was formalized as the πpattern problem. Let the pattern $P = p_1, \ldots, p_m$ 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 $(s_i, s_{i+1}, \ldots, s_{i+m-1})$. P is a πpattern if it appears at least K times in S for a given K. A notation for maximal πpatterns was introduced as a model to filter meaningful from apparently meaningless clusters. A πpattern p_1 is nonmaximal 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 by Eres *et al.* (2003) 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, a *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 (Booth and Leuker, 1976) 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 Eres *et al.* (2003).

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 1. *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 nonmaximal relations between patterns and can be used to filter meaningful from apparently meaningless clusters (nonmaximal); $((a, b, c, d) - (e - f))$ shows that the patterns $\pi_1 = \{e, f\}$ and $\pi_2 = \{a, b, c, d\}$ are nonmaximal 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) proof that a minimal consensus PQ tree is equivalent to a maximal πpattern (Eres *et al.*, 2003) and each subgraph of the PQ tree corresponds to a nonmaximal

Downloaded by Ben Gurion from online.liebertpub.com at 11/07/17. For personal use only.

permutation pattern and (b) 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, (2) *E. Coli K-12* and *B. Subtilis* genomes, and (3) *Chloroplast* genomes in the flowering plant family Campanulaceae. In all, 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.

2. PRELIMINARIES—PQ TREE

In this section, we present the PQ tree data structure and definitions as introduced by Booth and Leuker (1976), 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 Fig. 1. We accordingly define the number of frontiers obtainable from a tree equivalent with T to be $|\mathcal{C}(T)|$.

Clearly the equivalence 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

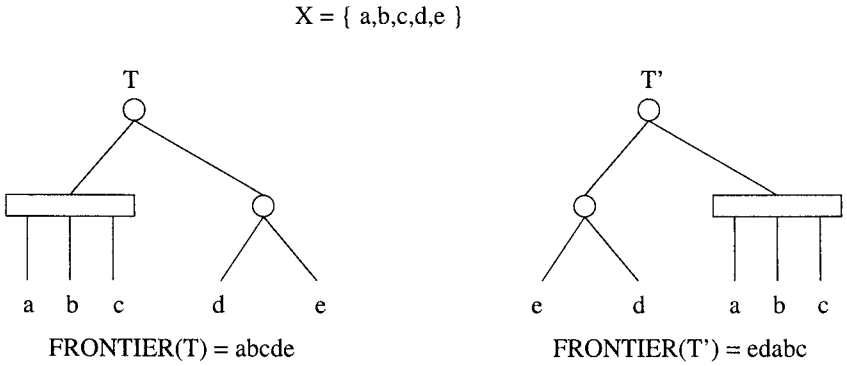


FIG. 1. Two equivalent PQ trees, $T' \equiv T$ and their frontiers. Note that $\mathcal{C}(T) = \mathcal{C}(T') = \{abcde, abced, cbade, cbaed, deabc, decba, edabc, edcba\}$.

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 Booth and Leuker (1976) 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 Booth and Leuker (1976), 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 by Heber and Stoye (2001).

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 $\mathcal{C}(T_2) \subseteq \mathcal{C}(T_1)$.*

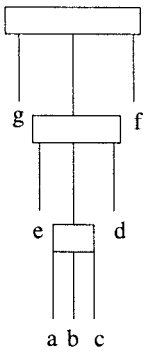
3. THE MINIMAL CONSENSUS PQ TREE

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 Fig. 1, T is denoted as $((a - b - c), (d, e))$ and the PQ tree in Fig. 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

Downloaded by Ben Gurion from online.liebertpub.com at 11/07/17. For personal use only.



(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.

Downloaded by Ben Gurion from online.liebertpub.com at 11/07/17. For personal use only.

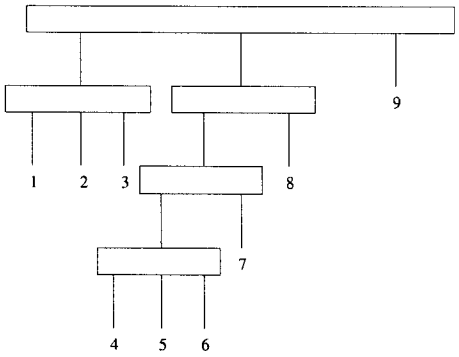
that $\mathcal{C}(T) = \{\pi_1, \pi_2, \ldots, \pi_k\}$; however, this is not always possible. Consider a πpattern $\{a, b, c, d, e\}$ appearing four times as $abcde, abced, cbade, edabc$; the PQ tree T in Fig. 1 is the one that best describes these appearances. However, $edcba \in \mathcal{C}(T)$ although 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 $|\mathcal{C}(T')| < |\mathcal{C}(T)|$.*

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 Uno and Yagiura (2000):

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.*

See Fig. 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.



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 π-pattern and the corresponding minimal consensus PQ tree.

Theorem 1. *Given* Π, $T_C = REDUCE(C_\Pi, T_U)$ *is a minimal consensus PQ tree of* Π.

Proof. By the definition of the procedure *REDUCE*() (see Section 2), $\Pi \subseteq \mathcal{C}(T_C)$. Hence, T_C is a consensus PQ tree of Π. Next we show that this tree is also minimal. Assume $T \not\equiv T_C$ is a minimal consensus PQ tree of Π. By Booth and Leuker (1976), there exists a collection $\mathcal{I}$ of subsets of N such that $T = REDUCE(\mathcal{I}, T_U)$. Each $I \in \mathcal{I}$ appears as a consecutive substring of each frontier $f \in \mathcal{C}(T)$. Since T is a minimal consensus PQ tree of Π, each $I \in \mathcal{I}$ appears as a consecutive substring of each $\pi \in \Pi$. Therefore, each $I \in \mathcal{I}$ is a common interval of Π and $\mathcal{I} \subseteq C_\Pi$. Using Observation 1, $\mathcal{C}(T_C) \subseteq \mathcal{C}(T)$; hence the result. ■

The following corollary is immediate from the proof of Theorem 1.

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

Proof. Let $T_2' \equiv T_2$ such that $F(T_1) = F(T_2)$. We define T_{m_j} to be the subtree rooted by the jth child of the root of T_m ($m = 1, 2$ and $j = 1, 2, 3, \ldots, n$). First we prove that for each $j \leq n$, $F(T_{1_j}) = F(T_{2_j})$. Assume the contrary: let $F(T_{1_i}) \neq F(T_{2_i})$ for the smallest such index i. Without loss of generality, there are more leaves in T_{2_i} than in T_{1_i}. Let

(1) $F(T_1) = F(T_2) = \alpha_1\alpha_2 \ldots \alpha_j\beta_1\beta_2 \ldots \beta_k\gamma_1\gamma_2 \ldots \gamma_\ell$, $2 \leq \ell$, $0 \leq j$,
(2) $F(T_{1_i}) = \beta_1\beta_2 \ldots \beta_k$, $2 \leq k$, and
(3) $F(T_{2_i}) = \beta_1\beta_2 \ldots \beta_k\gamma_1\gamma_2 \ldots \gamma_t$, $1 \leq t \leq \ell$.

Since the root of T_2 must have at least two children, $t < \ell$ or $0 < j$. There are two cases to consider:

1. T_1's root is a P node.
   - ($j > 0$) Move T_{1_i} to be the left most child of T_1's root to get T_1' with $F(T_1') = \beta_1 \ldots \beta_k\alpha_1 \ldots \alpha_j\gamma_1 \ldots \gamma_\ell \in \mathcal{C}(T_1)$. From the form of $F(T_{2_i})$ (see (3) above), it is impossible to have an appearance of α_1 between β_1 and γ_t; hence, $F(T_1') \notin \mathcal{C}(T_2)$.
   - ($j = 0$) Here $t < \ell$. Move T_{1_i} to be the rightmost child of T_1's root to get T_1' with $F(T_1') = \gamma_1 \ldots \gamma_\ell\beta_1 \ldots \beta_k \in \mathcal{C}(T_1)$. Again, it is impossible to have an appearance of γ_ℓ between β_1 and γ_t; hence, $F(T_1') \notin \mathcal{C}(T_2)$.
2. T_1's root is a Q node. In T_2, reverse the leaves of T_{2_i} (reversing leaves is possible both in a P node and in a Q node) to get T_2' with $F(T_2') = \alpha_1\alpha_2 \ldots \alpha_j\gamma_t\gamma_{t-1} \ldots \gamma_1\beta_k\beta_{k-1} \ldots \beta_1\gamma_{t+1}\gamma_{t+2} \ldots \gamma_\ell \in \mathcal{C}(T_2)$.
   - ($j > 0$) By the form of $F(T_1)$ (see (1) above), since T_1's root is a Q node it is impossible to have an appearance of γ_1 between α_1 and β_k. Hence, $F(T_2') \notin \mathcal{C}(T_1)$.
   - ($j = 0$) Here $t < \ell$. By the form of $F(T_{1_i})$ (see (2) above), it is impossible to have an appearance of β_1 between γ_t and γ_ℓ. Hence, $F(T_2') \notin \mathcal{C}(T_1)$.

Thus, for each $j \leq n$, $F(T_{1_j}) = F(T_{2_j})$. Further, the roots of T_1 and T_2 have exactly n children each.

Next we show that the roots of T_1 and T_2 are of the same type. Without loss of generality, let T_1's root be a P node (with at least three children) and T_2's root be a Q node. If we swap T_{1_2} and T_{1_1} to get T_1', it is easy to see that $F(T_1') \notin \mathcal{C}(T_2)$. Hence, our assumption must be wrong and the roots of T_1 and T_2 are of the same type.

Finally, since the roots of T_1 and T_2 are isomorphic, we can show the same for each pair T_{1_j} and T_{2_j} ($1 \leq j \leq n$). Hence, $T_1 \equiv T_2$, and consequently $T_1 \equiv T_2'$. ■

The following corollary is straightforward to verify.

Corollary 2. *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 π pattern. We handle this problem in Section 5.

Downloaded by Ben Gurion from online.liebertpub.com at 11/07/17. For personal use only.

Downloaded by Ben Gurion from online.liebertpub.com at 11/07/17. For personal use only.

3.1. Identifying maximal π patterns in the minimal consensus PQ tree

In this section, we describe a PQ subtree as a method for identifying nonmaximal permutation patterns, and we make the simplifying assumption that there are no multiplicities in the π patterns. 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 its 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 π pattern p_1 is nonmaximal with respect to π pattern 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 π patterns such that p_1 is nonmaximal with respect to p_2. We denote T_i as the minimal consensus PQ tree of the appearances of π pattern 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 nonmaximal π patterns (Theorem 3).

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

Theorem 3. *Given π patterns p_1, p_2 on some S, if p_1 is nonmaximal w.r.t. p_2, then $T_2^{p_1} \equiv T_1$.*

Proof. We first prove two lemmas.

Lemma 1. $C_{\Pi_1} \subseteq C_{\Pi_2}$.

If $i \in C_{\Pi_1}$, then the elements of i appear consecutively in every occurrence of p_1. However, since p_1 is nonmaximal with respect to p_2 and there are no multiplicities, every occurrence of p_2 covers exactly one occurrence of p_1. Thus, the elements of i appear consecutively in every occurrence of p_2 and $i \in C_{\Pi_2}$.

Lemma 2. *The leaves of $T_2^{p_1}$ are exactly $p_1 = \{\alpha_1, \alpha_2, \ldots, \alpha_n\}$.*

Since p_1 is nonmaximal with respect to p_2, every occurrence of p_2 covers exactly one occurrence of p_1. Thus, p_1 is a common interval of the occurrences of p_2. In Theorem 1, we showed that every common interval of the occurrences of p_2 appears as a consecutive substring of every $f \in \mathcal{C}(T_2)$. Therefore, p_1 appears consecutively in each $f \in \mathcal{C}(T_2)$. We next show that if the leaves of $T_2^{p_1}$ are not exactly p_1, then there is a frontier $f' \in \mathcal{C}(T_2)$ such that p_1 does not appear as a consecutive substring of f'. Assume $T_2^{p_1}$'s root is v and the leaves of $T_2^{p_1}$ are $\{\alpha_1, \alpha_2, \ldots, \alpha_n, \beta_1, \beta_2, \ldots, \beta_m\}$, $(m > 0)$. Consider the two possible cases depending on the types of node v:

Case 1. (Q): $T_2^{p_1}$ is the PQ tree rooted by the LCA of p_1. Since there is no single child of v that has p_1 as descendants, then there are two children of v, v_1, and v_2 such that α_i is a descendant of v_1 and α_j, β_k are descendants of v_2 for some $0 < k, i, j$ and $i \neq j$ (there has to be a child v_2 that has both α_j and β_k as descendants since v is a variant with minimal number of children that still covers p_1).

- If β_k appears between α_i and α_j in $F(T_2^{p_1})$, then $f' = F(T_2^{p_1})$ is such that p_1 does not appear as a consecutive substring of f'.
- Otherwise, by reversing the leaves of the PQ tree rooted by v_2 (this is possible whether v_2 is a Q node or a P node), we get the frontier f' such that p_1 does not appear as a consecutive substring of f'.

Case 2. (P): Since no single child of v has p_1 as descendants, there are two subcases:

- There exist two children of v, v_1, and v_2 such that α_i is a descendant of v_1 and α_j, β_k are descendants of v_2 for some $0 < k, i, j$ and $i \neq j$. Again, as before, frontier $f' \in \mathcal{C}(T_2)$ is such that p_1 does not appear as a consecutive substring of f'.
- There exist three children of v, v_1, v_2, and v_3 such that α_i is a descendant of v_1, α_j is a descendant of v_2, and β_k is a descendant of v_3 for some $0 < k, i, j$ and $i \neq j$. Here, by permuting the children of the P node such that v_3 is between v_1 and v_2, we get frontier f' such that p_1 does not appear as a consecutive substring of f'.

Thus, both cases lead to a contradiction, therefore the leaves of $T_2^{p_1}$ are exactly p_1.

Back to the theorem. Next, we prove that $\mathcal{C}(T_2^{p_1}) = \mathcal{C}(T_1)$:

- First we show that $\mathcal{C}(T_2^{p_1}) \subseteq \mathcal{C}(T_1)$. Assume the contrary; then there exists a frontier f such that $f \in \mathcal{C}(T_2^{p_1})$ and $f \notin \mathcal{C}(T_1)$. Hence, there is an interval $i \in C_{\Pi_1}$ such that the elements of i do not appear as a consecutive substring of f. However, from Lemma 1, if $i \in C_{\Pi_1}$ then $i \in C_{\Pi_2}$, and since $T_2 = \mathit{REDUCE}(C_{\Pi_2}, T_U)$, the elements of i appear as a consecutive substring of every frontier $f \in \mathcal{C}(T_2)$. Using Lemma 2, elements of i appear as a consecutive substring of f leading to a contradiction.
- Next we show that $\mathcal{C}(T_2^{p_1}) = \mathcal{C}(T_1)$. Assume the contrary, then $\mathcal{C}(T_2^{p_1}) \subsetneq \mathcal{C}(T_1)$. Any occurrence of p_1 is covered by an occurrence of p_2 and therefore occurs as a substring of a frontier $f \in \mathcal{C}(T_2)$. However, using Lemma 2, this occurrence is $\mathcal{C}(T_2^{p_1})$, and thus $T_2^{p_1}$ is a consensus PQ tree of p_1. However, if $\mathcal{C}(T_2^{p_1}) \subsetneq \mathcal{C}(T_1)$, then T_1 is not a minimal consensus PQ tree of p_1.

Thus, $\mathcal{C}(T_2^{p_1}) = \mathcal{C}(T_1)$, and using Theorem 2, $T_2^{p_1} \equiv T_1$. ■

Notice that the converse of Theorem 3 is true only if every occurrence of p_1 is covered by an occurrence of p_2. In our case, this property holds since we deal with πpatterns that appear exactly once in each of the k sequences.

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

Proof. Given a πpattern p and its minimal consensus PQ tree T, let N_T be the notation of T and M_p be the maximal notation of p. We prove that a "-" separates two subsets of p in N_T if and only if a "-" separates these subsets in M_p.

If p_1 and p_2 are two subsets of p separated by a "-" in M_p, then p_1 and p_2 are both nonmaximal πpatterns w.r.t. p and $p_1 \cup p_2$ is also a πpattern. By Theorem 3, this implies that there are two PQ subtrees of T such that their leaves are p_1 and p_2, respectively. Moreover, these PQ subtrees are two consecutive children of a Q node in T. Thus, a "-" separates p_1 and p_2 in N_T.

Conversely, if there is a "-" between p_1 and p_2 in N_T, then p_1 and p_2 are two consecutive children of a Q node, and therefore are immediate neighbors in every occurrence of p. Hence, by definition, there is a "-" between p_1 and p_2 in M_p. ■

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, and (2) using the exact permutations appearing in the strings for the P node. For example, if a P node has seven children and the annotation is (3162574,5142736), then this implies

Downloaded by Ben Gurion from online.liebertpub.com at 11/07/17. For personal use only.

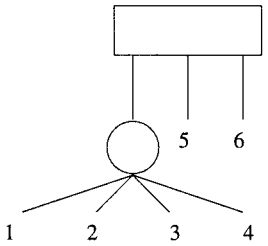


FIG. 4. Consider $\Pi = \{123456, 241356\}$. The PQ tree in the figure is the minimal consensus PQ tree of Π. Consider the following restrictions: (1) The Q node is un-annotated, and (2) the exact permutations of the P node is (2413). The restricted PQ tree is such that $\mathcal{C}(T) = \Pi$.

Downloaded by Ben Gurion from online.liebertpub.com at 11/07/17. For personal use only.

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 Fig. 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.

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 1, 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 by Heber and Stoye (2001): 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 nontrivial 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 nontrivial overlap. A chain of length one is called a trivial chain. A common interval I is called *reducible* if there is a nontrivial 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, $1 \leq |I_\Pi| \leq |C_\Pi| \leq \binom{n}{2}$ (Heber and Stoye, 2001). The set of irreducible common intervals of Π from the example in Fig. 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 Fig. 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 by Heber and Stoye (2001).
2. Compute $T = REDUCE(I_\Pi, T_U)$ using the algorithm described by Booth and Leuker (1976).
3. Return T.

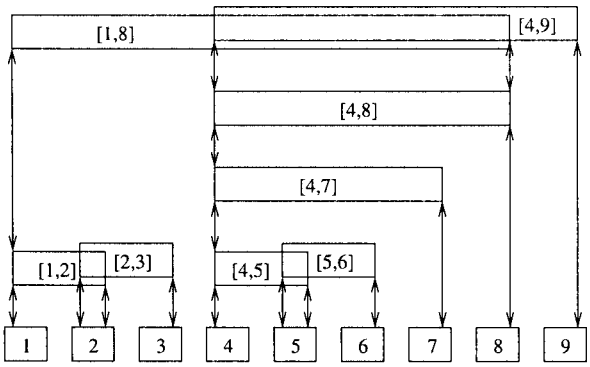


FIG. 5. Sketch of the data structure $\mathcal{S}$ of the example in Fig. 3. The chains of nontrivially overlapping irreducible intervals are ([1, 2], [2, 3]), ([4, 5], [5, 6]), and ([1, 8], [4, 9]).

In Theorem 1, we proved that $T_C = REDUCE(C_\Pi, T_U)$ is the minimal consensus PQ tree of Π. In the following theorem, we prove that if $T_I = REDUCE(I_\Pi, T_U)$ then $T_C = T_I$, thus proving the correctness of the algorithm.

Theorem 5. *Given* Π, $T_I = REDUCE(I_\Pi, T_U)$ *is the minimal consensus PQ tree of* Π.

Proof. In Theorem 1, we proved that if $T_C = REDUCE(C_\Pi, T_U)$ then T_C is the minimal consensus PQ tree of Π. We next prove that $T_I \equiv T_C$ and thus prove the theorem.

$I_\Pi \subseteq C_\Pi$ and from Observation 1 this implies that $\mathcal{C}(T_C) \subseteq \mathcal{C}(T_I)$. On the other hand, every $c \in C_\Pi$ is generated by a chain of one or more overlapping intervals from I_Π and, therefore, appears as a consecutive substring of every $f \in \mathcal{C}(T_I)$. Hence, $\mathcal{C}(T_I) \subseteq \mathcal{C}(T_C)$. Thus, $\mathcal{C}(T_C) = \mathcal{C}(T_I)$, and by Theorem 2, $T_C \equiv T_I$. ■

Time complexity of the algorithm. Given k permutations each of length n, by Heber and Stoye (2001), $|I_\Pi| < n$, and further, I_Π can be computed in $\mathcal{O}(kn)$ time. By Booth and Leuker (1976), computing $T = REDUCE(I_\Pi, T_U)$ 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 $REDUCE(I_\Pi, T_U)$ *in linear time*

In this section, we modify step 2 of the PQ-Construct algorithm to run in $\mathcal{O}(n)$ time. In Heber and Stoye (2001), a data structure $\mathcal{S}$ was used to obtain the irreducible intervals. For each chain of nontrivially 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 Fig. 3 is shown in Fig. 5.

We next describe a new procedure called $REPLACE(\mathcal{S})$ that transform $\mathcal{S}$ to the 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 Fig. 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 Fig. 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 two children is replaced by a Q node. The PQ tree obtained by $REPLACE(\mathcal{S})$ is illustrated in Fig. 3. A similar idea, for the case of *conserved intervals* (Bergeron *et al.*, 2004a) (as opposed to irreducible intervals) is discussed by Bergeron *et al.* (2004a), Bergeron and Stoye (2003), and Bergeron *et al.* (2004b).

The following theorem proves that we obtain the minimal consensus PQ tree using $REPLACE(\mathcal{S})$.

Downloaded by Ben Gurion from online.liebertpub.com at 11/07/17. For personal use only.

Theorem 6. *REPLACE*$(\mathcal{S}) =$ *REDUCE*(I_Π, T_U).

Proof. If T' is the PQ tree obtained by *REPLACE*$(\mathcal{S})$ and $T =$ *REDUCE*(I_Π, T_U), we will prove that $T \equiv T'$. First we prove that $\mathcal{C}(T') \subseteq \mathcal{C}(T)$. If $[i, j]$ is an irreducible interval, then the *REPLACE* operation guarantees that T' will have a PQ subtree (see Definition 8) with $i, i+1, i+2, \ldots, j$ as leaves. Therefore $\{i, i+1, i+2, \ldots, j\}$ will appear consecutively in every frontier $f \in \mathcal{C}(T')$. Thus, $\mathcal{C}(T') \subseteq \mathcal{C}(T)$. Next we prove that $\mathcal{C}(T) \subseteq \mathcal{C}(T')$. The general idea is to show that every Q node in T' is a Q node (or a consecutive part of a Q node) in T and all the leaves of a P node in T' are either leaves of a P node in T or leaves of some consecutive children of a Q node in T. Therefore, every frontier obtainable from T is obtainable from T'.

If v is a Q node in T', then this means that there is a chain of nontrivially overlapping irreducible intervals $c = ([i_1, j_1], [i_2, j_2], \ldots, [i_p, j_p])$ such that *REPLACE* transformed this chain into v. The intersection of every two intervals in the chain must appear consecutively in every frontier $f \in \mathcal{C}(T)$. The only way this is possible is if there is a Q node in T such that some consecutive children of this Q node are induced by the intersection intervals of the chain. Thus, all Q nodes in T' are Q nodes (or a consecutive part of a Q node) in T.

Furthermore, by Theorem 3, every irreducible interval defines the set of leaves of a PQ subtree. Thus, all leaves of a P node in T' are either leaves of a P node in T or leaves of some consecutive children of a Q node in T (by Definition 8). This last observation guarantees that $|\mathcal{C}(T)| \leq |\mathcal{C}(T')|$ and therefore $\mathcal{C}(T) \subseteq \mathcal{C}(T')$.

Therefore, $\mathcal{C}(T) = \mathcal{C}(T')$, and by Theorem 2, $T \equiv T'$. ■

Time complexity of the algorithm. By Heber and Stoye (2001), 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 (Uno and Yagiura, 2000) (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).

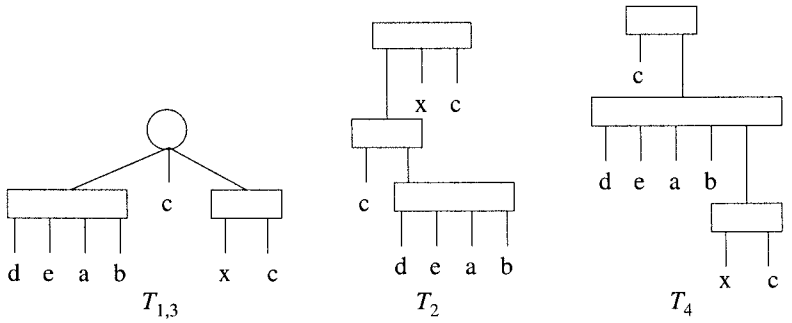
(1) Genomes as permutations with no multiplicity. This model is ideal for our tool. Since the k sequences are permutations of Σ with no multiplicity, Σ is a πpattern (common interval) of size n. Furthermore, it is the only maximal πpattern in the sequences. We construct the minimal consensus PQ tree, 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 $K\left(\leq \binom{n}{2}\right)$ is the number of πpatterns. Therefore, in $\mathcal{O}(nk + K)$ time we can output all the nonmaximal πpatterns, exactly like Heber and Stoye (2001) do, but we also present the maximal notation of every πpattern found and present the nonmaximal relations between them. We introduce experimental results of human and rat whole genome comparison for this type of data in Section 6.1.

(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.

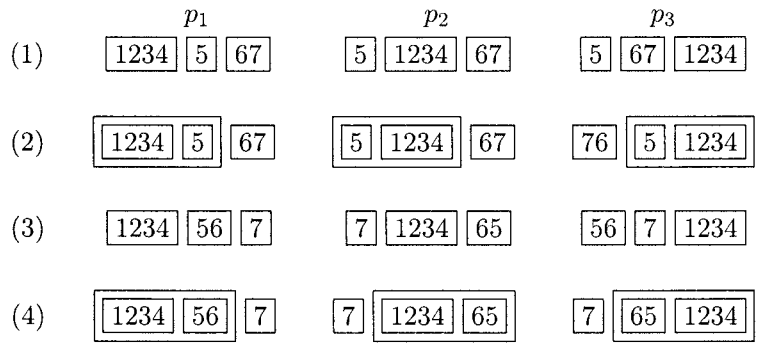
A string that has at least one character that appears more than once is termed as a string with *multiplicity*. For example, if $p_1 = abcegd$ and $p_2 = acgcab$, then p_1 has no multiplicity. However, that is not the case with p_2 where a and c each appear more than once.

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

Downloaded by Ben Gurion from online.liebertpub.com at 11/07/17. For personal use only.



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 $|\mathcal{C}(T_2)| = |\mathcal{C}(T_4)| = 8$.

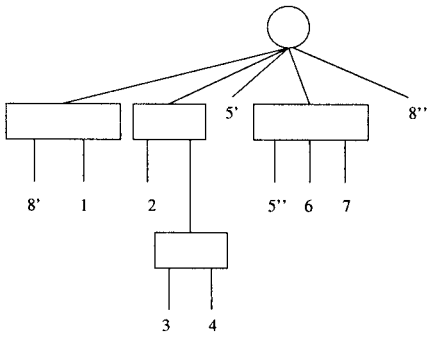
FIG. 6. The πpatterns with multiplicity.

a πpattern. This is illustrated in an example in Fig. 6. We handle multiplicity by reporting the multiple minimal consensus PQ trees. This is explained using the example of Fig. 6. Each character is labeled with a distinct integer in the reference sequence, and the remaining sequences are treated as multisets (strings of sets of characters). In the example, p_1 = *deabcxc* = 1234567, 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 Fig. 6 for the corresponding PQ trees.

In Section 6.2, we present experimental results for this type of data, of a pairwise comparison between the genomes of *E. Coli K-12* and *B. Subtilis*, and of a comparison between 13 chloroplast genomes in the flowering plant family Campanulaceae. We used the algorithm described by Eres *et al.* (2003) to find the πpatterns and our tool to present the maximal patterns as PQ trees (thus automatically filtering out the nonmaximal 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

Downloaded by Ben Gurion from online.liebertpub.com at 11/07/17. For personal use only.



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.

Downloaded by Ben Gurion from online.liebertpub.com at 11/07/17. For personal use only.

$12345'67$ 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 Fig. 7.

Time complexity of the algorithm. There are initially k sequences of length n each. In order to tag and add the elements as needed, for each element we save a list of the locations over all the sequences where this element appears. In the worst case, we have every element appearing in only one sequence, and thus it takes $\mathcal{O}(nk^2)$ time to tag and add the elements as needed. After we tag and add the elements, 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 introduce experimental results for this type of data in Section 6.3 of a comparison between eight chloroplast genomes in the flowering plant family Campanulaceae.

6. EXPERIMENTAL RESULTS FOR VARIOUS GENOME MODELS

In this section, we present experimental results for the three different genome models we discussed in Section 5.

6.1. Genomes as permutations with no multiplicity

Human and rat genomes. In this section, we build and analyze the PQ tree of human and rat whole genomes. In order to build a PQ tree for whole genome comparisons, we used the output of a program called SLAM (Alexandersson *et al.*, 2003; Bray *et al.*, 2003b, 2003a), 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, taken from *http://bio.math.berkeley.edu/slam/rat/geneTables.html*, sorted by human chromosomes. The data in every

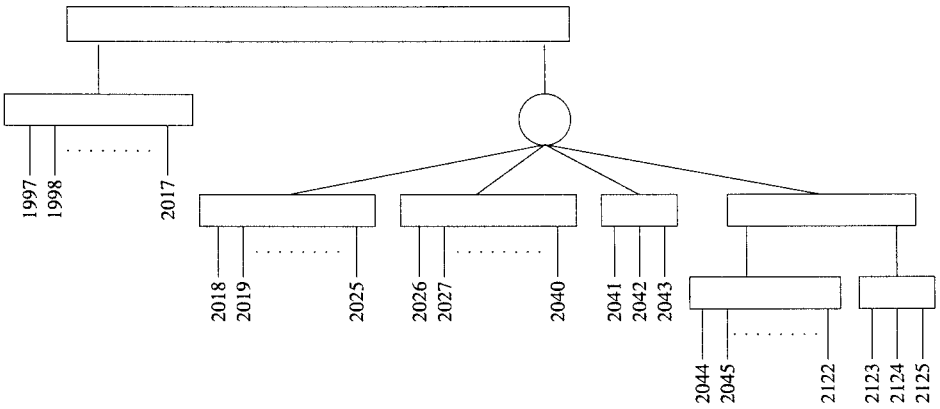


FIG. 8. A PQ subtree of the minimal consensus PQ tree of the human genome and the rat genome orthologous genes, as predicted by SLAM. The PQ tree is (((1997-1998-. . . -2017)-((2018-2019-. . . -2025),(2026-2027-. . . -2040), (2041-2042-2043),((2044-2045-. . . -2122)-(2123-2124-2125))))).

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 by mapping the first gene predicted by SLAM in human chromosome 1 (M4H1U1D4r106.002) to 1, the second (M4H1U1D4r106.001) to 2, and so on (after mapping chromosome 1, we move to chromosome 2, and so on). 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 *http://cs.haifa.ac.il/∼oweimann/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 dataset (we consider only patterns that do not cross chromosomes).

In Fig. 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 (coordinates 203975896-217701720) and in rat chromosome 13 (coordinates 92630521-103808253). By our mapping, these 129 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.

In Fig. 9, we present another subtree of the human-rat whole genome PQ tree. This tree corresponds to a section of 156 genes in human chromosome 17 (coordinates 7670777-21846001) and in rat chromosome 10 (coordinates 43816300-54626178). By our mapping, these 156 genes appear in the human genome as the permutation (20906-21061) and in the rat genome as the permutation (21028-21061, 21019-21027, 21018-20906). Figure 9 is the PQ tree of these two permutations. The two genes in bold represent genes PMP22

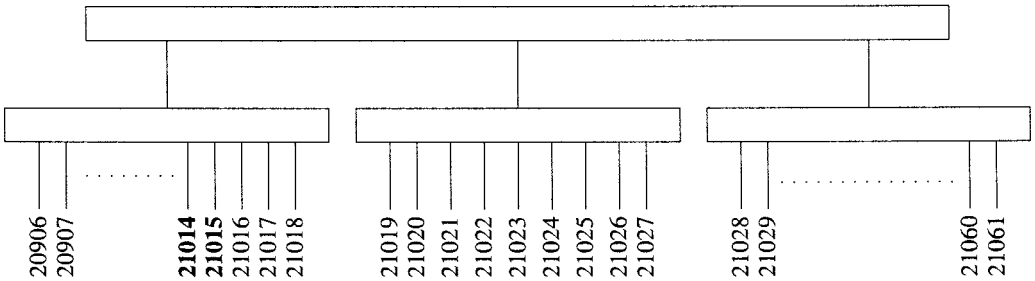


FIG. 9. The PQ tree in the figure is a PQ subtree of the minimal consensus PQ tree of the human genome and the rat genome orthologous genes, as predicted by SLAM. The genes in bold are the functionally related genes PMP22 and TEKTIN3.

Downloaded by Ben Gurion from online.liebertpub.com at 11/07/17. For personal use only.

and TEKTIN3 (denoted M11H17U1D5r2.002 and M11H17U1D5r2.001, respectfully, in SLAM data). The PQ tree clearly shows that these genes appear one next to the other both in human and in rat genomes; the reason for this is that they are indeed functionally related genes as explained by Burns *et al.* (2004). Note that the PQ tree in Fig. 9 shows the notation of the 156-π pattern {20906, 20907, 20908, . . . , 21061} which is (((20906 − 20907 − . . . − 21017 − 21018) − (21019 − 21020 − . . . − 21026 − 21027) − (21028 − 21029 − . . . − 21060 − 21061))).

6.2. Genomes as strings with multiplicity

***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 (2004), 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 $\delta = 1{,}900$, 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 Bryan *et al.* (1987) 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 He and Goldwasser (2004) is because the distance between the starting locations of COG2812 and COG0718 is 1984 ($> \delta = 1{,}900$).

Chloroplast genomes. Chloroplast DNA gene order changes have been useful in phylogenetic reconstruction in many plant groups. These changes have considerable potential to resolve phylogenetic relationships, and they provide valuable insights into the mechanisms of cpDNA evolution (Cosner *et al.*, 2000). We present a minimal consensus PQ tree obtained from a comparison between 13 chloroplast genomes in the flowering plant family Campanulaceae: *Trachelium*, *Campanula*, *Adenophora*, *Symphyandra*, *Legousia*, *Asyneuma*, *Triodanus*, *Wahlenbergia*, *Merciera*, *Codonopsis*, *Cyananthus*, *Platycodon*, and *Tobacco*. The input data was obtained from Cosner *et al.* (2000), where these genomes were analyzed for the purpose of reconstructing phylogenetic trees.

There were 167 maximal permutation patterns out of 10,769 patterns discovered by our tool. We present the minimal consensus PQ tree of the π pattern {1, 2, . . . , 54, 54, 55, 55, 56, . . . , 105} that appears in 2 out of the 13 chloroplast genomes as the following permutations:

- In *Campanula*: (1-15,76-49,39-37,40,35-26,44-41,45-48,36,25-16,90-84,77-83,91-96,55-54,105-97)
- In *Wahlenbergia*: (1-11,60-49,37-40,35-28,12-15,76-61,27-26,44-41,45-48,36,54,25-16,90-84,77-83,91-96,55,105-97)

Notice that this is a π pattern with multiplicity since genes 54 and 55 appear twice in the π pattern. We used the approach presented in Section 5 to address the problem of multiplicity. The four different labeling possibilities of genes 54 and 55 in the two appearances of the π pattern are:

(1) π_1 = (1-15,76-56,**55**,**54**,53-49,39-37,40,35-26,44-41,45-48,36,25-16,90-84,77-83,91-96,$\hat{\mathbf{55}}$,$\hat{\mathbf{54}}$,105-97) and π_2 = (1-11,60-56,**55**,**54**,53-49,37-40,35-28,12-15,76-61,27-26,44-41,45-48,36,$\hat{\mathbf{54}}$,25-16,90-84,77-83,91-96,$\hat{\mathbf{55}}$,105-97)

Downloaded by Ben Gurion from online.liebertpub.com at 11/07/17. For personal use only.

(2) π_1 = (1-15,76-56,**55**,**54**,53-49,39-37,40,35-26,44-41,45-48,36,25-16,90-84,77-83,91-96,$\hat{\mathbf{55}}$,$\hat{\mathbf{54}}$,105-97) and π_2 = (1-11,60-56,**55**,$\hat{\mathbf{54}}$,53-49,37-40,35-28,12-15,76-61,27-26,44-41,45-48,36,**54**,25-16,90-84,77-83,91-96,$\hat{\mathbf{55}}$,105-97)

(3) π_1 = (1-15,76-56,**55**,**54**,53-49,39-37,40,35-26,44-41,45-48,36,25-16,90-84,77-83,91-96,$\hat{\mathbf{55}}$,$\hat{\mathbf{54}}$,105-97) and π_2 = (1-11,60-56,$\hat{\mathbf{55}}$,$\hat{\mathbf{54}}$,53-49,37-40,35-28,12-15,76-61,27-26,44-41,45-48,36,**54**,25-16,90-84,77-83,91-96,**55**,105-97)

(4) π_1 = (1-15,76-56,**55**,**54**,53-49,39-37,40,35-26,44-41,45-48,36,25-16,90-84,77-83,91-96,$\hat{\mathbf{55}}$,$\hat{\mathbf{54}}$,105-97) and π_2 = (1-11,60-56,$\hat{\mathbf{55}}$,**54**,53-49,37-40,35-28,12-15,76-61,27-26,44-41,45-48,36,$\hat{\mathbf{54}}$,25-16,90-84,77-83,91-96,**55**,105-97)

In Fig. 10, we present the only minimal consensus PQ tree of the π pattern. Although we built four possible minimal consensus PQ trees for the four different labeling possibilities of genes 54 and 55, the one in the figure (which corresponds to the first labeling possibility) is the only one that had the smallest $|C(T)|$ and thus is the only minimal consensus PQ tree.

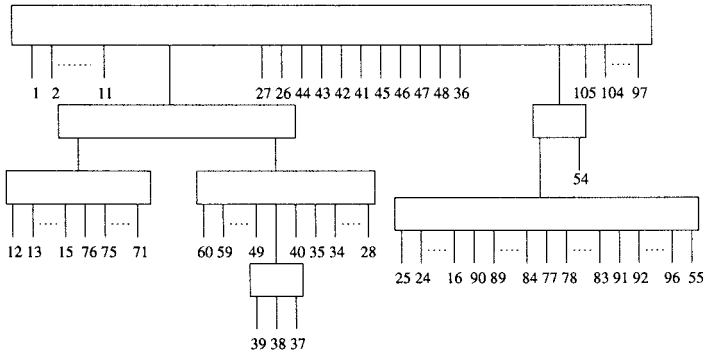


FIG. 10. The PQ tree in the figure is the minimal consensus PQ tree of the π pattern $\{1, 2, \ldots, 54, 54, 55, 55, 56, \ldots, 105\}$ that appears in two chloroplast genomes as (1-15,76-49,39-37,40,35-26,44-41,45-48,36,25-16,90-84,77-83,91-96,55-54,105-97) and (1-11,60-49,37-40,35-28,12-15,76-61,27-26,44-41,45-48,36,54,25-16,90-84,77-83,91-96,55,105-97).

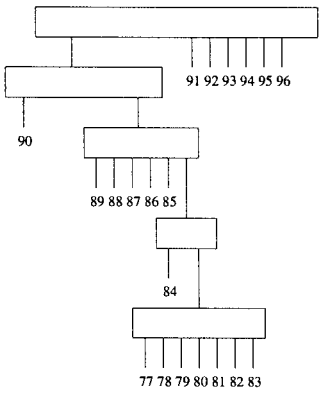
6.3. Genomes as strings with no multiplicity

Chloroplast genomes. We present two PQ trees obtained from a comparison between eight chloroplast genomes in the flowering plant family Campanulaceae: *Trachelium*, *Symphyandra*, *Asyneuma*, *Triodanus*, *Merciera*, *Codonopsis*, *Platycodon*, and *Tobacco*. The input data was obtained from Cosner *et al.* (2000). We used only 8 out of the 13 genome sequences that had no multiplicity (genome model 3). There were 77 maximal permutation patterns out of 6,729 patterns discovered by our tool.

We present the maximal notation of the π pattern $\{1, 2, \ldots, 105\}$ that appears in seven out of the eight chloroplast genomes as the following permutations:

- In *Trachelium*: (1-15,76-56,53-49,37-40,35-26,44-41,45-48,36,25-16,90-84,77-83,91-96,55-54,105-97).
- In *Symphyandra*: (1-15,76-56,39-37,49-53,40,35-26,44-41,45-48,36,25-16,90-84,77-83,91-96,55-54,105-97).
- In *Triodanus*: (1-15,76-56,27-26,44-41,45-48,36-35,25-16,89-84,77-83,90-96,55-53,105-98,28-34,40-37,49-52,97).
- In *Merciera*: (1-10,49-53,28-35,40-37,60-56,11-15,76-61,27-26,44-41,45-48,36,54,25-16,90-85,77-84,91-96,55,105-97).
- In *Codonopsis*: (1-8,36-18,15-9,40,56-60,37-39,44-41,45-53,16-17,54-55,61-76,96-77,105-97).
- In *Platycodon*: (1,8,2-5,29-36,56-50,28-26,9,49-45,41-44,37-40,16-25,10-15,57-59,6-7,60-96,105-97).
- In *Tobacco*: (1-105).

Downloaded by Ben Gurion from online.liebertpub.com at 11/07/17. For personal use only.



The PQ tree on the left is the minimal consensus PQ tree of the πpattern $\{77, 78, \ldots, 96\}$ that appears in all eight chloroplast genomes we explored in the flowering plant family Campanulaceae. as the following permutations:
In *Trachelium* and in *Symphyandra*: (90-84, 77-83, 91-96)
In *Asyneuma* and in *Triodanus*: (89-84, 77-83, 90-96)
In *Merciera*: (90-85, 77-84, 91-96)
In *Codonopsis*: (96-77)
In *Platycodon* and in *Tobacco*: (77-96)

FIG. 11. The minimal consensus PQ tree of the πpattern $\{77, 78, \ldots, 96\}$. The input data was obtained from Cosner *et al.* (2000).

The maximal notation of this pattern is (1-((2-3-4-5),(6-7),8,9,10,(11-12-13-14-15),(76-75-74-73-72-71-70-69-68-67-66-65-64-63-62-61),60,(59-58-57),56,53,(52-51-50),49,(37-38-39),40,35,(34-33-32-31-30-29),28,(27-26),((44-43-42-41)-(45-46-47-48)),36,(25-24-23-22-21-20-19-18),(17-16),((90-((89-88-87-86-85)-(84-(77-78-79-80-81-82-83))))-(91-92-93-94-95-96)),55,54,(105-104-103-102-101-100-99-98),97)).

In Fig. 11, we present another minimal consensus PQ tree of a πpattern that appears in all eight chloroplast genomes.

ACKNOWLEDGMENTS

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

REFERENCES

Alexandersson, M., Cawley, S., and Pachter, L. 2003. SLAM—Cross-species gene finding and alignment with a generalized pair hidden Markov model. *Genome Res.* 13(3), 496–502.

Bergeron, A., Blanchette, M., Chateau, A., and Chauve, C. 2004a. Reconstructing ancestral gene orders using conserved intervals. *Proc. 4th Workshop on Algorithms in Bioinformatics (WABI)*, 14–25.

Bergeron, A., Corteel, S., and Raffinot, M. 2002. The algorithmic of gene teams. *Proc. 2nd Workshop on Algorithms in Bioinformatics (WABI)*, 464–476.

Bergeron, A., Mixtacki, J., and Stoye, J. 2004b. Reversal distance without hurdles and fortresses. *Proc. 15th Ann. Symp. on Combinatorial Pattern Matching (CPM)*, 388–399.

Bergeron, A., and Stoye, J. 2003. On the similarity of sets of permutations and its applications to genome comparison. *Proc. 9th Ann. Int. Conf. on Computing and Combinatorics (COCOON)*, 68–79.

Booth, K., and Leuker, G. 1976. Testing for the consecutive ones property, interval graphs, and graph planarity using pq-tree algorithms. *J. Comput. System Sci.* 13, 335–379.

Bray, N., Couronne, O., Dubchak, I., Ishkhanov, T., Pachter, L., Poliakov, A., Rubin, E., and Ryaboy, D. 2003a. Strategies and tools for whole-genome alignments. *Genome Res.* 13(1), 73–80.

Bray, N., Dubchak, I., and Pachter, L. 2003b. AVID: A global alignment program. *Genome Res.* 13(1), 97–102.

Bryan, S.K., Hagensee, M.E., and Moses, R.E. 1987. DNA polymerase III requirement for repair of DNA damage caused by methyl methanesulfonate and hydrogen peroxide. *J. Bacteriol.* 16(10), 4608–4613.

Burns, K.H., Matzuk, M.M., Roy, A., and Yan, W. 2004. Tektin3 encodes an evolutionarily conserved putative testicular micro tubules-related protein expressed preferentially in male germ cells. *Molecular Reproduction and Development* 67, 295–302.

Cosner, M.E., Jansen, R.K., Moret, B.M.E., Raubeson, L.A., Wang, L., Warnow, T., and Wyman, S. 2000. An empirical comparison of phylogenetic methods on chloroplast gene order data in Campanulaceae. *Proc. of Gene Order*

Downloaded by Ben Gurion from online.liebertpub.com at 11/07/17. For personal use only.

Dynamics, Comparative Maps, and Multigene Families (DCAF), and *in* Sankoff, D., and Nadeau, J., eds., *Comparative Genomics: Empirical and Analytical Approaches to Gene Order Dynamics, Map Alignment, and the Evolution of Gene Families*, 99–121, Kluwer Academic Pub., Dordrecht.

Didier, G. 2003. Common intervals of two sequences. *Proc. 3rd Workshop on Algorithms in Bioinformatics (WABI)*, 17–24.

Eres, R., Landau, G.M., and Parida, L. 2003. A combinatorial approach to automatic discovery of cluster-patterns. *Proc. 3rd Workshop on Algorithms in Bioinformatics (WABI), Lecture Notes in Bioinformatics* 2812, 139–150.

He, X., and Goldwasser, M.H. 2004. Identifying conserved gene clusters in the presence of orthologous groups. *Proc. 8th Ann. Int. Conf. on Research in Computational Molecular Biology (RECOMB)*, 272–280.

Heber, S., and Stoye, J. 2001. Finding all common intervals of k permutations. *Proc. 12th Ann. Symp. on Combinatorial Pattern Matching (CPM)*, 207–218.

Landau, G.M., Parida, L., and Weimann, O. 2005. Using PQ trees for comparative genomics. *Proc. 16th Ann. Symp. on Combinatorial Pattern Matching (CPM)*.

McConnell, R.M. 2004. A certifying algorithm for the consecutive-ones property. *Proc. 15th Ann. ACM-SIAM Symp. on Discrete Algorithms (SODA)*, 15, 761–770.

Mulley, J., and Holland, P. 2004. Small genome, big insights. *Nature* 431, 916–917.

Schmidt, T., and Stoye, J. 2004. Quadratic time algorithms for finding common intervals in two and more sequences. *Proc. 15th Ann. Symp. on Combinatorial Pattern Matching (CPM)*, 347–358.

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

Address correspondence to:
Gad M. Landau
Department of Computer Science
University of Haifa
Mount Carmel
Haifa 31905, Israel

E-mail: landau@cs.haifa.ac.il

Downloaded by Ben Gurion from online.liebertpub.com at 11/07/17. For personal use only.