ISSN 0026-8933, Molecular Biology, 2015, Vol. 49, No. 3, pp. 327–338. © Pleiades Publishing, Inc., 2015.
Original Russian Text © K.Yu. Gorbunov, R.A. Gershgorin, V.A. Lyubetsky, 2015, published in Molekulyarnaya Biologiya, 2015, Vol. 49, No. 3, pp. 372–383.

MOLECULAR PHYLOGENETICS

UDC 575.852

Rearrangement and Inference of Chromosome Structures

K. Yu. Gorbunov, R. A. Gershgorin, and V. A. Lyubetsky

Institute for Information Transmission Problems of the Russian Academy of Sciences (Kharkevich Institute), Moscow, 127051 Russia;

e-mail: lyubetsk@iitp.ru

Received December 17, 2014; in final form, December 24, 2014

Abstract—The chromosome structure is defined as a set of chromosomes that consist of genes assigned to one of the DNA strands and represented in a circular or linear arrangement. A widely investigated problem is to define the shortest algorithmic path of chromosome rearrangements that transforms one chromosome structure into another. When equal rearrangement costs and constant gene content are considered, the solution to the problem is known. In this work, a principally novel approach was developed that presents an exact algorithm with linear time complexity for both equal and unequal costs, in which chromosome structures defined on the same set of genes were considered. In addition, to solve the problem of the inference of ancestral chromosome structures containing different sets of genes when the original structures are fixed in leaves, exact and heuristic algorithms were developed.

DOI: 10.1134/S0026893315030073

Keywords: chromosome structure, chromosome rearrangement, effective exact algorithm, ancestral structure, species tree, evolution along a species tree, parsimony

INTRODUCTION

In [1–3] and other works, various tasks were investigated related to estimating the number of chromosome rearrangements in different genomic regions of various species, determining the regions that are unlikely or highly likely to be affected by rearrangements, estimating the frequency of chromosome rearrangements at different stages of the evolution, and identifying the periods of sharp increase in the rearrangement frequency. These problems were solved based on the identification and comparison of homologous synteny blocks. The algorithm allows one to obtain an heuristic solution to the problems mentioned above, to define the most significant events related with chromosome rearrangements, and to perform an approximate reconstruction of the structure of ancestral genomes. However, it does not allow one to construct scenarios of evolutionary chromosome rearrangement along a species tree. Thus, it is important to develop effective exact algorithms for calculating the distances between chromosome structures and use this information to construct a complete scenario for chromosome rearrangements in genomes of various species. The first problem requires definition of the shortest sequence of rearrangement operations leading to a transformation of one chromosome structure into another. The second problem allows one to define the most likely chromosome rearrangement scenarios, estimate the evolutionary proximity of different species, and reveal rearrangements related to mobile elements and characteristic proteins.

This problem was solved in the case of equal rearrangement costs and constant gene content (see article [4] and books that were published thereafter [5, 6]). The historical development and biological motivation of the problems are given in detail in [1–6]. An extensive series of works by P. Pevzner and his school, in which algorithms of chromosome rearrangements included inversions and in part transversions and translocations, are noteworthy (see review, chapter 4 in [6]). In our opinion, in this chapter the exact polynomial algorithms were only discussed in the case of a chromosome, a structure that consists of a single linear chromosome with no paralogs and the operation included the inversion of an interval. This is a special case, i.e., a so-called double cut and join operation, which is one of many operations that are discussed below.

In this work, a novel, exact algorithm of a problem with linear computational complexity is proposed that is based on a method that is principally different from those employed in [4–6]. We also present an heuristic algorithm for the broader task of defining the shortest sequence of chromosome rearrangements with unequal operating costs when the total sequence costs were minimized. The task can also be solved with some limitations using an exact linear algorithm; however, this solution was submitted for publication in the journal *Problems of Information Transmission* as its presentation requires the use of specific mathematical means.

An exact algorithm is defined as an algorithm that always results in the global minimum of the corre-

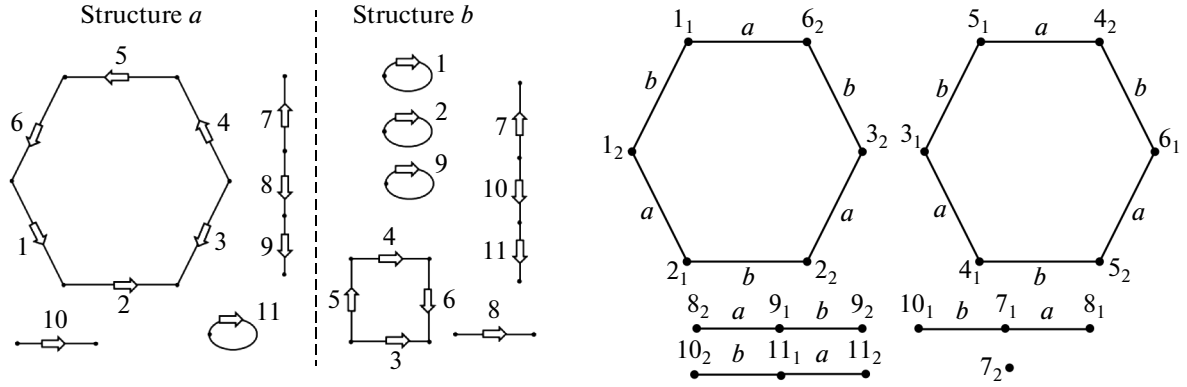

Fig. 1. Two chromosome structures *a* and *b* consisting of $n = 11$ genes each (left panel) and their joint graph $a + b$ (right panel).

sponding cost or some other functional defined in the problem statement.

We also propose cubic exact algorithms, as well as heuristic algorithms, for a new problem, i.e., the reconstruction of chromosome structures (the structures were fixed in leaves) that consists of different sets of genes on internal vertices of a species tree. These two algorithms emphasize different definitions of determining the distance between pairs of structures corresponding to edge ending points in a species tree. The precision of the algorithm was proved for one distance and is not proved for the other, which is likely to be more significant from an evolutionary point of view.

All algorithms were computer tested, and examples of the use of corresponding computer programs (chromo and chrom_reconstruction) can be found at <http://lab6.iitp.ru/ru/pr_chromo/>. Programming, computing, data preparation and analysis were performed by R. Gershgorin. Definitions and the results described in the section "Problem-Solving Algorithm and Its Justification" and subsections "Special Distance" and "Case of Unequal Operating Costs" were prepared by K. Gorbunov and V. Lyubetsky. This work represents an extended version of two plenary lectures given by the authors at conferences [7, 8].

FORMULATION OF A PROBLEM OF THE SHORTEST DISTANCE OF CHROMOSOME REARRANGEMENTS

A chromosome structure was defined as a set of linear and circular chromosomes, where every gene is assumed to have a head and a tail; the length of a gene, its nucleotide composition, and the intergenic regions of a chromosome, were not taken into account. In this model, all genes in the chromosome were assumed to have a linear localization in a linear or circular chromosome and were connected to each other. When two strands were considered, the following connection options were possible for adjacent genes: the tail of one gene coincided with the head of the other or the same gene, or with the tail of another gene, and the head-to-head connection of genes was also possible. A chromosome structure can be looked at as a graph that consists of components, each of which denotes a chromosome. The names (numbers from one to *n* with no overlap) of genes were ascribed to the edges; thus, every gene was represented only by its name (number). An example of two chromosome structures (*a* and *b*) is given in Fig. 1 (left panel). The heads and/or tails of two genes were equated in a vertex connecting the genes (in accordance with arrows); the head or tail of a gene were ascribed to the extreme vertex of a linear chromosome. The head of a gene *i* (at $j = 1$) or its tail (at $j = 2$) were denoted as i_j; a loop in a vertex represented a gene, the head of which coincided with its tail, e.g., gene 11 in the structure *a* as shown in Fig. 1 (left panel).

Let structures *a* and *b* defined on the *same number* of genes *n* be fixed as shown, e.g., in Fig. 1 (left panel), where $n = 11$.

Equating heads and tails (ends) of genes that correspond to their adjacent positions on a chromosome were denoted with a ~ symbol; see, e.g., cycle, where $5_2 \sim 6_1$, $4_2 \sim 5_1$, $3_1 \sim 4_1$, $2_2 \sim 3_2$, $1_2 \sim 2_1$, and $1_1 \sim 6_2$ (Fig. 1, left panel).

The novel approach proposed in this work is based on the use of two definitions that possess nontrivial properties, including a joint graphjoint graph and its quality. A *joint graph* of structures *a* and *b* was defined as a graph in which vertices were represented by the ends i_j of all genes that belong to *a* (or the same genes in *b*), and the edges connect two vertices if they equate in *a* or *b*. Every edge was denoted by the name of a structure, in which equating gene ends occurred, i.e., with the structure names *a* or *b*. The edges of a graph can be parallel to each other, that is, one edge from *a* can be parallel to another one from *b* (cycles of length two). The joint graph (denoted $a + b$) can be described as alternating (*a*, *b*) paths and (*a*, *b*) cycles (Fig. 1, right panel). Thus, the joint graph $a + b$ defined the ends of the gene that equate in structures *a* and *b*. Equating is often referred to as *joining* the corresponding i_j ends.

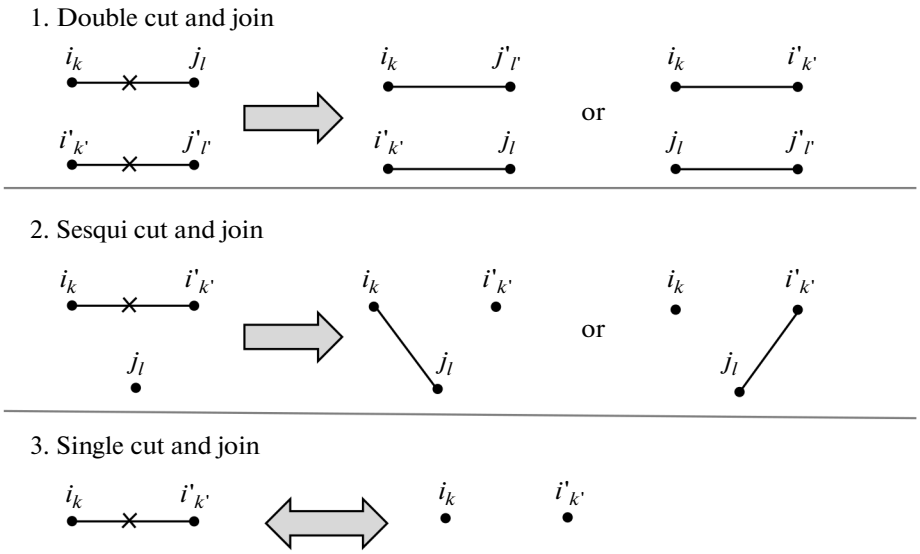

Fig. 2. Operations applied to a chromosome structure. A cross indicates a cut (disjoining of two joint vertices) and a double arrow indicates the result of an operation. Operations were introduced in [4].

The *length* of a path (or cycle) corresponds to its number of edges; isolated vertices were considered to be paths with zero length, taking zero as an even number. The *quality* of the joint graph was the number of cycles summed with half of the numerical value of its even paths (an even path consists of an even number of edges, while odd paths are not taken into account).

The formal problem can be formulated in terms of either a and b structures or its joint graph $a + b$. Let us start with the first statement. For structures a and b, we aimed to define the sequence consisting of a minimal number of operations that transforms one structure into the other, for example, structure a into the structure b (Fig. 1, left panel). We denoted both the sequence path and its length as minimal. In other words, we concentrated on searching for a structure c, into which both structures a and b could be converted via a minimal total number of operations. This follows from a notion that a set of operations is closed with respect to the inverse operation.

The set of operations on a chromosome or a pair of chromosomes (and thus on a chromosome structure) included the following: a cut of two joint vertices and a join of the four loose ends in a different way (double cut and join); the cut of a joint vertex and a join of its one loose end with a disjoint end (sesqui cut and join); the cut of a vertex and the inverse operation of joining two disjoint ends (single cut and join, Fig. 2) [4].

We denoted a joint graph of two similar structures, that is, the graph $c + c$ of the structure c, as *final* (*final representation*). The graph contained only cycles of length two (one edge from structure a and another one from structure b) and isolated vertices. These structures are further referred to as two-cycle.

Let us consider the second formulation of the problem, that is, the construction of a joint graph $a + b$, to which natural analogs of the operations described above can be applied. These operations included, e.g., removing two similarly labeled edges and joining its four ends with two new nonincident edges with a similar label; removing an edge and joining (with a similarly labeled edge) one of its ends with a vertex that is incident to no edges with this label; and removing any edge and inserting an edge, e.g., labeled with a, between two vertices that are nonincident to any other edge with the label a. The task was to find the shortest sequence of operations that brings $a + b$ to its final representation, i.e., to the form $c + c$ for structure c. We denoted this sequence of operations as minimal. The quality of the final graph equals n, and $n - k$ is the length of the shortest sequence of operations, where k represents the quality of the original joint graph $a + b$ and n is the number of genes in the original structure a (or the same number in b). Thus, defining the shortest operation sequence can be described as defining the quality of the graph $a + b$.

It is noteworthy that the operation applied to a pair of structures (when an operation was applied to one structure, the other structure remained unaltered) corresponds to an analogous operation applied to a joint graph.

The two problem formulations that were described above are equivalent to each other and this follows from the equivalency of each of these formulations to the task of defining structure c and two sequences of operations, one of which transforms the structure a into c, while the other transforms b into c so that the total number of operations in both of these sequences is minimized.

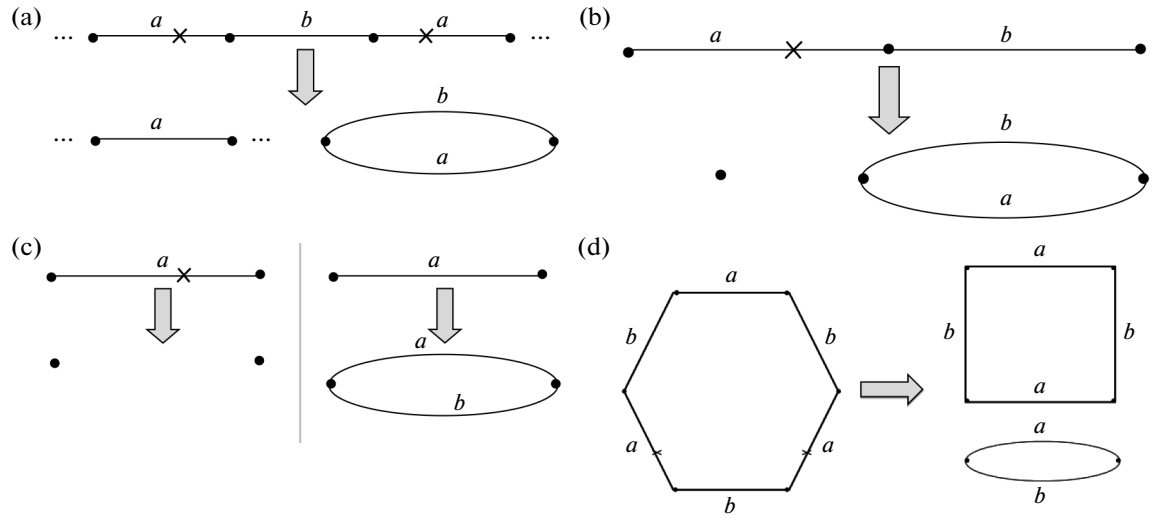

Fig. 3. (a) Isolating a cycle from a path with a length that is greater than two. Head (before the cross) of the left edge of structure *a* is glued to the tail (from the cross) of the right edge *a*. Tail (from the cross) of the left edge *a* is glued to the head (before the cross) of the right edge *a*. Edge *b* does not change. (b) Isolating a cycle from a path of length two. (c) Operations removing a chain with length one, resulting in two isolated dots (left) or a two-cycle (right). (d) Splitting a cycle with a length that is greater than two into two cycles. As a result, heads (before the crosses) of edges of structure *a* connect to form a square; tails (from the crosses) of edges *a* connect with *b* to form a cycle.

The quality of the joint graph did not change or is changed by precisely one upon application of any operation to it; this can be proved by considering all pairs that consist of an operation and a type of joint graph components to which the operation was applied. Our approach was based on the following statements: the quality of the final graph equals n, while the graph is not final and the operation that increases its quality can be applied; in the case of the final graph, an operation that can lead to an increase in its quality does not exist. In other words, the quality of the joint graph of two distinct structures is strictly below n, whereas the highest quality is equal to n and is achieved in the case of coincident structures, i.e., the final graph.

Note. It is important to note that the joint graph $a + b$ can be conveniently stored in the dataset M with indexes ranging from $-n$ to n; negative labels denote the heads of corresponding genes, whereas positive enumeration labels correspond to gene tails in structures a and b. The $M[i]$ value represents a pair of end labels, by which the end i is joined in structures a and b (if it is not joint, the value equals zero). This way of storing data provides linear time and memory of the algorithm for constructing the graph $a + b$ for structures a and b, as well as a quick assessment of its components and a switch from operations with $a + b$ to operations with a and b.

PROBLEM-SOLVING ALGORITHM AND ITS JUSTIFICATION

Let us describe an exact algorithm with linear time complexity that solves the problem in its second formulation, i.e., it brings the joint graph of two structures $a + b$ to its final representation.

The algorithm modified $a + b$ as follows (see points 1–4):

(1) If the path's length is greater than two, we reduce it by removing a cycle of length two using a double cut and join operation, which results in a cycle and a path (Fig. 3a). Let us consider a sequence of type *..aba..* within a path in detail. We discard *ba* from the path, leaving *..a..*, and add a cycle *ab* (Fig. 3a); similar operations were performed with a sequence of type *..bab...*

(2) If the path's length equals two, we use a sesqui cut and join operation to split the path into a cycle and a path of length zero (Fig. 3b).

(3) If the path's length equals one, we split it into two paths of length zero using a single cut operation or use a single join operation to form a cycle (Fig. 3c).

(4) If there is a cycle with length greater than two, we break it into two cycles using a double cut and join operation (Fig. 3d). More precisely, we consider a sequence of type *..aba..*, for which we keep an edge *a* that forms a shorter cycle with the rest of the original cycle and add a two-cycle in the structure (Fig. 3d). We perform similar operations with the sequence *..bab...*

If in any order to apply these operations as long as possible, then we obtain the required minimum sequence of operations.

It is important that the algorithm only uses the information about the *graph type* for constructing the sequence, including the number of cycles in the graph

and their lengths, as well as the number and lengths of paths.

Complexity and Correctness of the Algorithm

The algorithm obviously has a linear time complexity. The quality of the joint graph increases by one every time an operation described in the algorithm is used. If the joint graph is not in its final representation, one of the operations can be used again.

Let us note a new operation that also leads to an increase in the quality of the joint graph by one. The operation represents a cut of an odd path with a formation of two even paths using a single cut of an edge labeled similarly to an extreme edge of the path, i.e., *a* or *b*. More precisely, this edge is discarded whereas the remaining edges form two new paths; alternatively, an isolated vertex is formed instead of an edge. It is possible to add this operation and simultaneously limit the use of the first operation to even paths only. Moreover, any sequence of operations that strictly increases the quality of the joint graph can be used in our algorithm, and one of them can be applied until the graph achieves its final representation.

The following sections are devoted to generalizations of the original problem formulated above.

PROBLEM IN THE CASE OF UNEQUAL OPERATION COSTS

Let us assume a cost represented by a strictly positive number to every operation from the list given above and every inverse operation. Genome structures were defined on the same set of genes. The operations that occur rarely in the evolution process were assigned high costs, whereas frequently occurring operations were assumed lower costs. The task was to define a sequence of operations that transforms the chromosome structure *a* into *b* and has a minimal total cost. We denote the sequence of operations as the *shortest* and its cost as *minimal*. The original task is a special case with all operations assumed to be equal costs. The question of whether a linear or at least a polynomial algorithm for solving this general problem exists remains open. To solve the general problem, an heuristic algorithm was proposed in this work (see notes at the end of this section).

Let us define a directed graph, for which the vertices are the joint graphs and call it a *digraph*. More precisely, the only information assumed to its vertices was the information about the type of joint graph of this particular vertex, in other words, the information used in the algorithm that was discussed above. In a digraph, the edge drawn from vertex v_1 to vertex v_2 if graph v_1 can be transformed by one of the operations into the graph v_2; in this case, the edge was labeled with the cost of the operation if it was applied to the structure *a* or with the cost of the inverse operation if it was applied to the structure *b*. The root of the digraph assumed the joint graph $a + b$ of the given *a* and *b* structures. All paths start in the root and continue toward a leaf, which was assumed in the final graph obtained on this walk (with a cost of *n*). Thus, the task was to find the shortest path in the digraph from the root into a leaf.

For each joint graph v, a minimal sequence defined by the algorithm that was described above, brings v to its final representation. We denote the cost of the most expensive operation multiplied by the minimal length as $t(v)$; similarly, the lowest cost multiplied by the minimal length was denoted as $l(v)$.

The following algorithm was then applied to the digraph, which results in an heuristic solution to the problem defined in this section. Only a small part of the digraph was used in the algorithm; therefore, it did not require to be constructed in full.

Let O be the set that includes the root of the digraph and consists of vertices v; the shortest paths in the digraph from the root to v that are located strictly within O were defined, and every vertex in O was ascribed the shortest path cost $c(v)$. Let us designate α as the minimum $c(v) + t(v)$ value for all vertices v in the set O. Let us label all the vertices v in O for which $c(v) + l(v) > \alpha$. Our aim was to extend O with a single new vertex. To do this, let us consider a set O_1 that consists of vertices that do not belong to O, which are connected by edges with unlabeled vertices from O. Every vertex v in O_1 was assumed a number $c(v)$ that represents the shortest path cost from the root to v among all paths and the vertices of which belong to O, except for v itself. A vertex w from O_1 with the minimal $c(v)$ value, which results in a new neighborhood $O \cup \{w\}$, was chosen.

Neighborhoods were enclosed into each other and expanded for as long as possible. A vertex with the minimal $c(v)$ value was selected from the leaves that were a part of the latter neighborhood. To construct the shortest path from the root to the vertex, the inverse of the algorithm was used.

Note. Let us designate c_1, c_1', $c_{1.5}$, and c_2 the costs of cut, join, sesqui cut and join, and double cut and join operations. If the condition that the search for the shortest path is carried out among minimal paths (conventionally the shortest) is imposed, an effective exact algorithm with linear time complexity can be constructed for solving the problem in this section, at least in the case of cost ratios of $c_2 \le c_1 \le c_1' \le c_{1.5}$ and $c_1 \le c_1' \le c_{1.5} \le c_2$. If the costs are relatively similar to each other, e.g., they change within the δ to ε range and $\delta/\varepsilon > n/(n + 1)$, the shortest path conventionally coincides with the shortest path. In the general case, they might be different.

RECONSTRUCTION OF CHROMOSOME STRUCTURES ALONG A TREE: STATEMENT OF THE PROBLEM, ALGORITHMS, AND ARTIFICIAL EXAMPLES

Let us consider a *new problem* of reconstructing a chromosome structure along a tree. For simplification we assume equal operation costs. Given the evolution species tree (that does not have to be binary), each leaf of which was assigned a structure defined on a certain set of genes, we aimed to reconstruct structures of internal vertices of the tree (*ancestral structures*) so that the *functional*, which equals the sum (over all edges) of distances between structures at edge ends, is minimized. Structures in the internal vertices can only include genes that are represented in leaves, and we assigned S to denote the set of these genes. Thus, the reconstruction of structures defined in leaves was based on the principle of parsimony.

Choosing this distance is a nontrivial task and requires further discussion. We thus considered a number of tasks depending on definition of the distance. In this work we consider two such definitions. According to the first definition, the distance between structures a and b (assumed to ends of edges of the tree) was defined as the number of pairs of different gene ends which are joint in one structure and absent (one or both) or disjoint in the other structure, summed with the number of genes that are present in one structure and absent in the other. We designated this distance as *special*. According to the second definition, which is more relevant biologically, *distance* was a minimal length between a and b as defined by the original problem statement. The argument of these functionals was the *distribution* of the structures in all internal vertices of the tree. Since the structures were assumed in leaves, any distribution assigned a single structure to every vertex of the tree. The value of the functional for this distribution was called the *cost*.

In the case of the first distance definition, the problem was solved with an effective exact algorithm, the complexity of which has an order of the product of $|S|^2$ and a number of leaves, i.e., it is relatively low. The algorithm can be applied to a tree partitioned with time slices [9–11]. In addition, the algorithm extends to a functional, every summand of which has a coefficient that reflects the evolutionary length of an edge. In case of the second definition of the distance, which we called *biological*, an heuristic algorithm with a cubic complexity (in practice) was used to solve the problem.

Special distance. For a given distribution, each tree vertex, and each pair of different ends of genes in S, a variable that equals one when the ends join in the structure corresponding to the vertex and zero in any other situation, was introduced. In addition, for each gene in S, we introduce a variable that equals one when the gene was absent from this structure and zero otherwise. We do not distinguish between the vertex and its structure.

The values of all variables were defined in the leaves of the tree. We aimed to find the values for each variable in internal vertices in order to minimize the functional. The functional can be reformulated as the number of edges where the values of the variable differ at the ends; every edge is counted the number of times that corresponds to the number of these variables. The algorithm that we used to minimize the functional is described below. We start with finding the minimum for each variable separately; it is evident that this step is characterized by a linear time complexity. As a result, we obtain a set of values of all variables in all vertices that we further refer to as the *marking* of the tree. The marking obtained here may indicate joining one end with two different ends in a vertex (type-1 violation) or joining the end of a gene that is absent in a vertex (type-2 violation).

Therefore, the next step of the algorithm includes the removal of all violations. This step also has a linear time complexity of the product of a number of variables and the size of the tree; importantly, the cost of the marking does not change during this step. Thus, the algorithm finds a solution, i.e., a marking at which the cost is minimized.

Step of the algorithm to remove violations. Let us arrange the ends of all genes in S in a linear order, e.g., lexicographically. Next, we sort out the vertices of the tree in arbitrary order; in each vertex, the ends of the gene were sorted according to the mentioned order. For each end, all of its join adjacencies in a given vertex were discarded if there is an even number of them; otherwise, a single joint adjacency was retained with the end that is the largest in this order. These operations allow one to remove type-1 violations. The genes were then placed in ascending order based on their names. Here, for every end of an absent gene that is joined with any other end (it is evident that it can only be joined with one other end), this joint was discarded and the gene was considered to be in the vertex. This step removes type-2 violations.

Justification of violation removal. This step apparently removes all violations; however, it is important to demonstrate that the marking cost does not increase. Type-1 violation in vertex v is a pair of variables that take on the value one, which corresponds to two pairs (a, b) and (a, c) of ends of genes in S; we denoted these pairs as *incident*. An order at the gene ends induces the order at incident pairs of variables. The procedure for removing type-1 violations that was described above is equivalent to the following steps. First, we sort incident pairs of variables in this order. For each pair, we sort vertices of the tree and, in every vertex, we replace both values with zero if the corresponding variables take on a value of one. In this case, new violations for other end pairs do not arise. Thus, it is sufficient to show that the marking cost does not change for a fixed pair of variables (x, y) during the procedure.

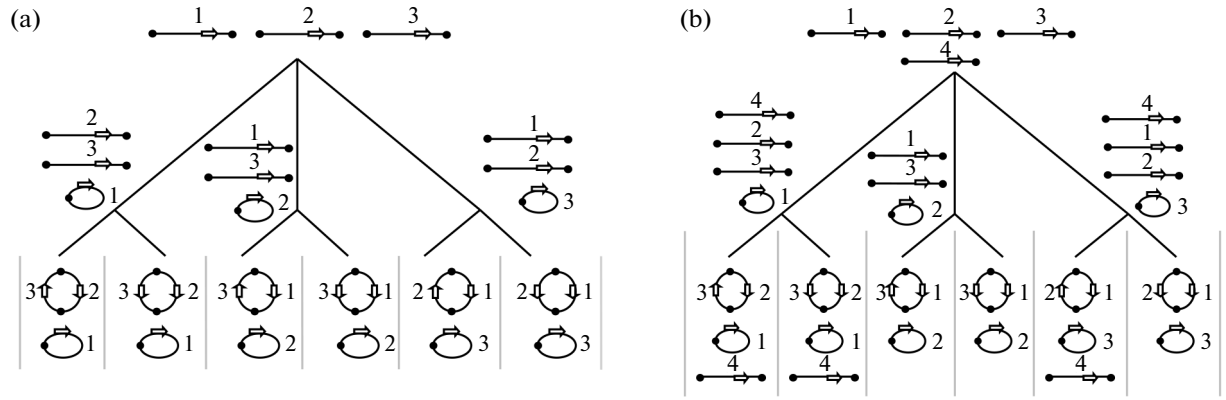

Fig. 4. Mapping of structures in leaves and the mapping of ancestral structures along the tree solved by the algorithm are given: (a) case of equal gene sets in leaves and (b) case of unequal gene sets in leaves.

Let us call the difference between the marking cost and the minimal marking cost a *defect* in marking. An edge of the tree was considered *inconsistent* if its marking at one end was (1, 1), i.e., $x = 1$, $y = 1$, whereas its marking at the other edge was (0, 0). By definition, violations were inpossible in the leaves of the tree. The internal vertices of the tree were then sorted in arbitrary order and if the marking in a vertex was found to be (1, 1), it was replaced with (0, 0). Let defect of the current marking be designated as d. Let us prove that the two following conditions are always true: d is an even number (zero is considered even) and there are at least $d/2$ inconsistent edges in the tree. Indeed, the marking defect is initially equal to zero and the conditions are satisfied. Let us consider a vertex v. If there is a violation in this vertex, then the marking (1, 1) is replaced with (0, 0). Then, let us consider a vertex u that is incident to the vertex v. The following three cases should be considered. If the marking in u is (0, 1) or (1, 0), then the marking cost of the edge (v, u) does not change and this edge was and is consistent. If the marking in u equals (0, 0), the marking cost of the edge (v, u) decreases by two; thus, this edge was inconsistent but became consistent. If the marking in u equals (1, 1), the marking cost of the edge (v, u) increases by two; this edge was consistent but became inconsistent. Thus, the number of consistent edges increases (decreases) by c if and only if the cost of the marking increases (decreases) by $2c$.

At the end of the procedure application, violations and inconsistent edges will be removed and the marking defect would become equal to zero. Since the variable values were always changed from one to zero, new violations (for other variable pairs) would not arise. It is noteworthy that the procedure for removing violations could also be used for a arbitrary graph.

A pair of variables that take on the value of one, the first variable of which is an indicator of the absence of a gene and the second variable of which corresponds to a joint gene end, represent a type-2 violation in vertex v; we designate pairs of the variable as *incident.* An order at gene ends induces the order at incident pairs of variables. The procedure for removing type-2 violations includes the following steps. First, we sort incident pairs of variables in this order. For each pair, we sort vertices of the tree and, at every vertex, we replace both values with zero if the corresponding variables take on a value of one. We then precisely repeat the steps used to remove type-1 violations.

Artificial Examples of the Evolution of Chromosome Structures

Two examples using biological data are given below (see examples 4 and 5).

Example 1. Every genome structure contains three genes with names 1, 2, and 3 in leaves (Fig. 4a). The following conditions are true in leaves: eighteen variables are equal to one, three of which correspond to a joint adjacency of the head of a gene with its tail, and two more variables in each leaf correspond to the ratio of the other ends (Fig. 4a). The minimal marking of the tree assumes the value of one to the first three variables in only one internal vertex of the tree, in the parent of two corresponding leaves. All other variables are assigned zero values in all internal vertices. There are no violationes.

Example 2. This is the case of different gene sets in leaves. A new gene with a name 4 was added in three leaves of the tree in example 1 (cf., Figs. 4a, 4b). A variable that describes the absence of this gene equals zero in the root and two side vertices, and one in the vertex in the middle.

Case of Unequal Operation Costs

The algorithm is valid in the case when the costs of four events, i.e., joining two ends, cutting them, and adding or removing a gene, are unequal. The step of the algorithm constructing the minimal marking does not change. Now, when violations are resolved, the cost of marking may increase. However, at a cost ratio

that seems biologically relevant, i.e., the cost of joining two ends does not exceed the cost of their cut and the cost of a gene loss does not exceed that of its gain, the marking cost also does not increase. In other words, our algorithm remains the exact solving algorithm.

The following statement that is significantly stronger is also true: no violations arise if the cost of $1 \to 0$ transformation is strictly below the cost of $0 \to 1$ transformation. Indeed, let us designate d the difference of costs of these transformations and fix an incident pair of variables. The use of induction on the height of the tree shows that, if there is a violation in the root of the tree, its removal leads to a decrease in the marking cost by at least $2d$; otherwise, the cost does not change. This is valid for any (not just minimal) marking. The initial step of the induction is obvious; therefore, let us define the inductive step. Consider a tree with the root r and let us assume that the statement is true for all children trees of the vertex r. If the marking in a children's vertex r' equals (1, 1), we *call* the edge (r, r') and a tree with the root r' singular. Let us sort the cases of the marking (x, y) of the vertex r. Let $(x, y) = (0, 0)$. It is obvious that the marking cost would not increase after resolving the violations. Let $(x, y) = (0, 1)$ or $(x, y) = (1, 0)$. In this case the marking cost at every singular edge increases by d but it decreases by $2d$ in every singular tree. The total marking cost does not change.

Let $(x, y) = (1, 1)$. The cost decreases by at least d at every nonsingular edge, it does not change in case of a singular edge, and decreases by $2d$ in a singular tree. Since the vertex contains at least two children, the total marking cost decreases by at least $2d$.

The next step is to prove the absence of any violations in the marking by contradiction. Let us consider a maximum (by insertions) tree T with the root v labeled with a pair (1, 1). If there were violations in the marking, resolving them in T (marking cannot be changed outside T) would have decreased the marking cost in T by $2d$, whereas the cost on the *parent* edge of the vertex v could have increased by less than d (we use here the fact that there is no marking (1, 1) above the vertex v). The total marking cost decreases, which is not true, since it is already minimal.

Biological Distance

Examples of the algorithm with the costs of joining operations of three and the costs of cut operations of two, which is different from the costs considered in this section, are described in the following section. The marking cost increases insignificantly as a result of resolving all violations.

The following heuristic algorithm is considered. Firstly, the ancestral structures were mapped using a special distance, for which the values of costs are used. In the case of long edges, the costs can be multiplied by correcting factors that account for the length of evolution. Thus, the obtained mapping serves as the first step for the algorithm of minimizing the mapping cost based on the biological distance. At every step of the algorithm, all internal vertices of the tree and all operations were sorted out. A step of the algorithm included choosing a pair <vertex, operation>, which minimizes the mapping cost, and further the operation was used in the vertex. We denote this algorithm as the *descent*.

To allow multiple descent steps using different initial mappings, parameter p, which reflects the penalty for disjoining the two ends of the gene, was introduced to the functional with a special distance. The algorithm described above is characterized by $p = 0$. The descent approach was carried out at values of p ranging from zero to three with a step of 0.1, resulting in a number of different mappings. Among those, the best mapping was selected that represents the final mapping of ancestral structures.

The given definition of the biological distance can be corrected. In detail, if the structure consists of mainly circular chromosomes, the double cut and join operations are the most frequent ones; therefore, the cost of the double cut and join operation, c_{double}, was equal to 0.8, whereas the cost of any other cut and join operation, c_{other}, was 1.2. Let us designate l the number of edges in a component of the joint graph of two initial structures. Thus, the distance between these two structures is equal to the total of the value $0.5c_{\text{double}}(l-2)$ over all cycles summed with the total of $0.5c_{\text{double}}(l-1) + c_{\text{other}}$ over even paths and $0.5c_{\text{double}}(l-2) + c_{\text{other}}$ over odd paths. It is easy to demonstrate that this distance is equal to the minimal total cost of operations among the sequence of operations that transform one structure into another and have a minimal length.

We obtained a different mapping of ancestral structures based on the biological distance as compared with similar approaches based on the use of special distance (see, e.g., Fig. 4a). Precisely, the (1, 2, 3) cycle is in the root, cycles (1) and (2, 3) are on the left, cycles (2) and (1, 3) are in the center, and cycles (3) and (1, 2) are on the right; all genes are located on the same strand. Indeed, for the data in Fig. 4a, the evolutionary scenario with a special distance contains 15 single joint adjacencies (one at upper edges and two at lower edges), while the same scenario with the biological distance includes six double-cut and join events (one at the upper edges and one at the three lower edges).

Example 3. Sets of genes are given in leaves; the solving algorithms using a special and biological distance are different (Fig. 5).

It is evident that, in the case of biological distance, the scenario in Fig. 5a contains one single cut at two upper edges and one deletion and one single join at four lower edges. In Fig. 5b, one deletion at four lower edges is observed.

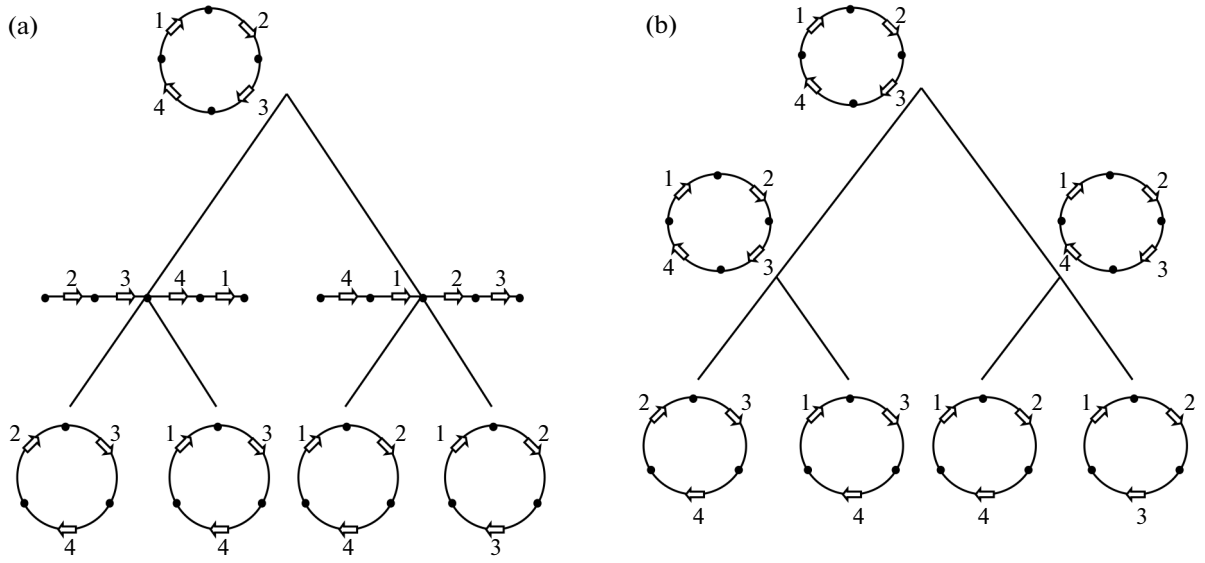

Fig. 5. Mapping of structures in leaves and the mapping of ancestral structures along the tree solved by the algorithm are given: (a) case of a special distance and (b) case of a biological distance.

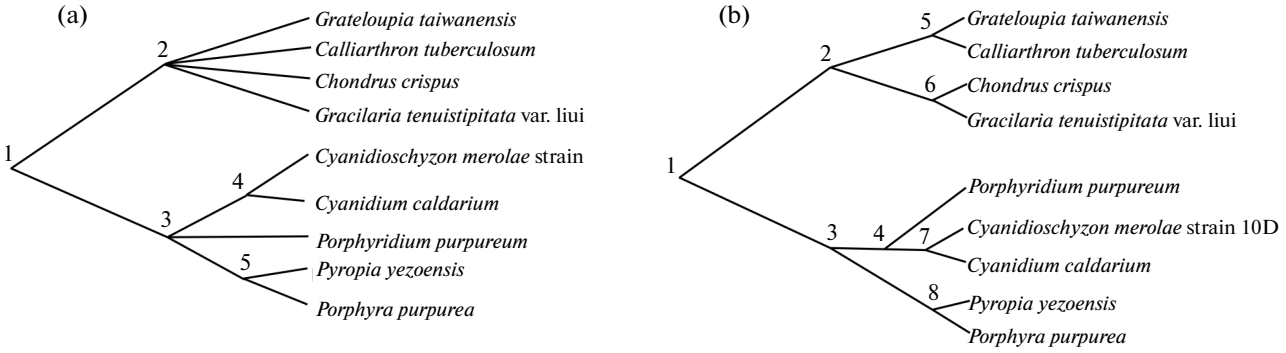

Fig. 6. Tree for given species with enumerated internal vertices: (a) original polytomic tree; (b) its binary resolution.

Examples of the Reconstruction of the Chromosome Structure Using Biological Data

Example 4. Original data are given as structures of plastid genes from the following nine species of red algae: NC_021618 *Grateloupia taiwanensis*, NC_021075 *Calliar throntuberculosum*; NC_020795 *Chondrus crispus*, NC_006137 *Gracilaria tenuistipitata* var. *liui*, NC_023133 *Porphyridium purpureum*, NC_004799 *Cyanidioschyzon merolae* strain 10D, NC_001840 *Cyanidium caldarium*, NC_007932 *Pyropia yezoensis*, and NC_000925 *Porphyra purpurea*. A genome of every red algae consists of a single circular chromosome. In each genome, genes related with photosystems I and II were selected. According to NCBI taxonomy, the species tree is polytomic, i.e., nonbinary (Fig. 6a). Thus, the following chromosome structures, each containing 25 genes, were given in leaves. Let us list genes of plastids in the order given for the species above. The order of genes reflects their location on the chromosome, * indicates the complementary strand, and symbol |C indicates a circular chromosome as follows:

psaK *psaC psaI *psbJ *psbL *psbF *psbE *psaM psbA *psbV *psaJ *psaF psbD psbC psaE *psbH psbN *psbT *psbB psbK *psaA psbI psaL |C;

psbA *psbV *psaJ *psaF psbD psbC psaE *psbH psbN *psbT *psbB psbK *psaB *psaA *psaD psbI psaL psaK *psaC psaI *psbJ *psbL *psbF *psbE *psaM |C;

psaF psaJ psbV *psbA psbD psbC psaE *psbH psbN *psbT *psbB psbK *psaB *psaA *psaD psbI psaL psaK *psaC psaI *psbJ *psbL *psbF *psbE *psaM |C;

psaF psaJ psbV *psbA psbD psbC psaE *psbH psbN *psbT *psbB psbK *psaB *psaA *psaD psbI psaL psaK *psaC psaI *psbJ *psbL *psbF *psbE *psaM |C;

psbD psbC *psbV *psaB *psaA *psbK *psaE psbA *psaJ *psaF psaD *psbJ *psbL *psbF *psbE *psaI *psaL psaM *psbI *psbH psbN *psbT *psbB *psaC *psaK |C;

psaM psbA *psaK *psaC psbD psbC psbB psbT *psbN psbH *psaE psaA psaB *psbK *psaD psaF psaJ psbV psaI *psbJ *psbL *psbF *psbE psaL *psbI |C;

psbD psbC psaI *psbJ *psbL *psbF *psbE psaL *psbI psbA psaK *psaC psaE *psbH psbN *psbT *psbB *psaM psaF psaJ psbV psaD psbK *psaB *psaA |C;

psaF psaJ psbV *psbA psaL *psbI psaD psaA psaB psbK psbB psbT *psbN psbH *psaE *psbC *psbD psaK *psaC psaI *psbJ *psbL *psbF *psbE *psaM |C;

psaF psaJ psbV psbA psaL *psbI psaD psaA psaB psbK psbB psbT *psbN psbH *psaE *psbC *psbD psaK *psaC psaI *psbJ *psbL *psbF *psbE *psaM |C.

The algorithm was tested in two different scenarios, i.e., it was applied directly to the polytomic tree and to each of its binary solutions. The minimal solving cost in the case of the polytomic tree is 27.9. The solution is given in the order of vertex numbering.

psaL *psbIpsaDpsaApsaB *psbKpsbBpsbT *psbBpsbH *psaE *psbC *psbDpsaK psaCpsaIpsbJ psbLpsbF *psbE *psaMpsaFpsaJpsbVpsbA |C;

psaLpsaK *psaCpsaI *psbJ *psbL *psbF *psbE *psaMpsaFpsaJpsbV *psbApsbDpsbCpsaE *psbHpsbN *psbT *psbBpsbK *psaB *psaA *psaDpsbI |C;

psaL *psbIpsaDpsaApsaB *psbKpsbBpsbT *psbNpsbH *psaE *psbC *psbDpsaK psaCpsaIpsbJ *psbL *psbF *psbE *psaMpsaFpsaJpsbVpsbA |C;

psaL *psbIpsaDpsbK *psaB *psaApsbDpsbCpsaI *psbJ *psbL *psbF *psbE |C and psbBpsbT *psbNpsbH *psaEpsaC *psaK *psbA *psbV *psaJ *psaFpsaM |C;

psaL *psbIpsaDpsaApsaBpsbKpsbBpsbT *psbNpsbH *psaE *psbC *psbDpsaK *psaCpsaI *psbJ *psbL *psbF *psbE *psaMpsaFpsaJpsbVpsbA |C.

The minimal mapping indicates that, during evolution, a structure that consists of a single circular chromosome, except for the vertex 4, in which the second circular chromosome arose, was favored. The latter is likely to be related to the necessity of faster replication in unfavorable environmental conditions.

The algorithm gives the minimal cost of 25.2 for a binary resolution shown in Fig. 6b of all polytomies in the original tree. The solution is ordered according to the vertex enumeration:

psaL *psbIpsaDpsaApsaB *psbKpsbBpsbT *psbNpsbH *psaE *psbC *psbDpsaK *psaCpsaI*psbJ *psbL *psbF *psbE *psaMpsaFpsaJpsbV *psbA |C;

psaLpsaK *psaCpsaI *psbJ *psbL *psbF *psbE *psaMpsaFpsaJpsbV *psbApsbDpsbCpsaE *psbFIpsbN *psbT *psbBpsbK *psaB *psaA *psaDpsbI |C;

psaL *psbIpsaDpsaApsaB *psbKpsbBpsbT *psbNpsbH *psaE *psbC *psbDpsaK *psaCpsaI*psbJ *psbL *psbF*psbE *psaMpsaFpsaJpsbV *psbA |C;

psaL *psbIpsaMpsaI *psbJ *psbL *psbF *psbEpsbK *psaB*psaA *psbA *psbV *psaJ *psaFpsaD |C and psbBpsbT *psbNpsbH *psaE *psbC *psbDpsaK *psaC |C;

psaLpsaK *psaCpsaI *psbJ *psbL *psbF *psbE *psaMpsbA *psbV *psaJ *psaFpsbDpsbCpsaE *psbHpsbN *psbT *psbBpsbK *psaB *psaA *psaDpsbI |C;

psaLpsaK *psaCpsaI *psbJ *psbL *psbF *psbE *psaMpsaFpsaJpsbV *psbApsbDpsbCpsaE *psbHpsbN *psbT *psbBpsbK *psaB *psaA *psaDpsbI |C; psaL *psbIpsaMpsaI *psbJ *psbL *psbF *psbE |C and psaDpsbK *psaB *psaApsbDpsbCpsbBpsbT *psbNpsbH *psaEpsaC *psaK *psbA *psbV *psaJ *psaF |C;

psaL *psbIpsaDpsaApsaBpsbKpsbBpsbT *psbNpsbH *psaE *psbC *psbDpsaK *psaCpsaI *psbJ *psbL *psbF *psbE *psaMpsaFpsaJpsbV *psbA |C.

The minimal mapping of structures to the internal vertices of the tree shown in Fig. 6b indicates that evolution favored a structure consisting of a single circular chromosome, except for vertices no. 4 and 7, in which two chromosomes are present.

Example 5. Genes of ribosomal proteins and RNA-polymerase subunits were selected from the plastids. Chromosome structures, each consisting of 45 genes (the order of the genes is similar to that given in Example 1), were assigned to leaves of the same nonbinary tree as follows:

rps4 *rps6 *rpl27 *rpl21 *rpl32 rpl34 rpsl6 *rpl12 *rpl1 *rpl11 rpl19 *rps10 *rps7 *rps12 *rpl31 *rps9 *rpl13 *rpoA*rps11 *rps13 *rp136 *ips5 *rpl18 *rpl6 *rps8 *rpl5 *rpl24 *rpl14 *rps17 *rpl29 *rpl16 *rps3 *rpl22 *rpl2 *rpl23 *rpl4 *rpl3 *rpsl4 rpl35 rpl20 *rps18 *rpl33 rpoBrps2 *rpl28 |C;

*rps6 *rpl27 *rpl21 *rpl32 rpl34 rps16 *rpl12 *rpl1 *rpl11 rpl19 *rps10 *rps7 *rps12 *rpl31 *rps9 *rpl13 *rpoA*rps11 *rps13 *rpl36 *rps5 *rpl18 *rpl6 *rps8 *rpl5 *rpl24 *rpl14 *rps17 *rpl29 *rpl16 *rps3 *rpl22 *rpl2 *rpl23 *rpl4 *rpl3 *rps14rpl35 rpl20 *rps18 *rpl33 rpoBrps2 *rpl28 rps4 |C;

*rpl34 rpl32 rpl21 rpl27 rps6 rps16 *rpl12 *rpl1 *rpl11 rpl19 *rps10 *rps7 *rps12 *rpl31 *rps9 *rpl13 *rpoA*rps11 *rps13 *rpl36 *rps5 *rpl18 *rpl6 *rps8 *rpl5 *rpl24 *rpl14 *rps17 *rpl29 *rpl16 *rps3 *rpl22 *rpl2 *rpl23 *rpl4 *rpl3 *rps14 rpl35 rpl20 *rps18 *rpl33 rpoBrps2 *rpl28 rps4 |C;

*rpl34 rpl32 rpl21 rpl27 rps6 rps16 rpl1 rpl1 rpl12 rpl19 *rps10 *rps7 *rps12 *rpl31 *rps9 *rpl13 *rpoA*rps11 *rps13 *rpl36 *rps5 *rpl18 *rpl6 *rps8 *rpl5 *rpl24 *rpl14 *rps17 *rpl29 *rpl16 *rps3 *rpl22 *rpl2 *rpl23 *rpl4 *rpl3 *rps14 rpl35 rpl20 *rps18 *rpl33 rpoBrps2 *rpl28 rps4 |C;

*rps6 rpl34 *rpl1 *rpl11 rpl12 rpl35 rpl20 *rps4 rpl21 rpl27 *rps18 *rpl33 rps16 rpl19 *rps5 *rpl18 *rpl6 *rps8 *rpl5 *rpl24 *rpl14 *rpsl7 *rpl29 *rpl16 *rps3 *rpl22 *rpl2 *rpl23 *rpl4 *rpl3 rps10 rps14 *rps7 *rps12 *rpl31 *rps9 *rpl13 *rpoA*rps11 *rps13 *rpl36 *rpl28 rpl32 *rpoBrps2 |C;

rps4 *rpl28 rpl21 rpl27 rpl32 rpl34 rps6 rps16 rpl11 rpl1 rpl12 rpl19 rps14 rpl35 rpl20 rpl3 rpl4 rpl23 rpl2 rpl22 rps3 rpl16 rpl29 rps17 rpl14 rpl24 rpl5 rps8 rpl6 rpl18 rps5 rpl36 rps13 rps11 rpoArpl13 rps9 rpl31 rps12 rps7 rps10 *rps18 *rpl33 rpoBrps2 |C;

*rps18 *rpl33 rpoBrps2 rpl21 rpl27 rpl32 rpl34 rps6 *rpl19 rpl11 rpl1 rpl12 *rps16 rps4 *rpl28 *rps10 *rps7 *rps12 *rpl31 *rps9 *rpl13 *rpoA*rps11 *rps13 *rpl36 *rps5 *rpl18 *rpl6 *rps8 *rpl5 *rpl24 *rpl14 *rpsl7 *rpl29 *rpl16 *rps3 *rpl22 *rpl2 *rpl23 *rpl4 *rpl3 *rpl20 *rpl35 *rps14 |C;

*rpl34 rpl32 rpl21 rpl27 *rps4 *rpl28 *rps2 *rpoBrpl33 rps18 *rpl20 *rpl35 *rps10 *rps7 *rps12 rpl31 *rps9 *rpl13 *rpoArps11 rps13 *rpl36 *rps5 *rpl18 *rpl6 *rps8 *rpl5 *rpl24 *rpl14 *rps17 *rpl29 *rpl16 *rps3 *rpl22 *rpl2 *rpl23 *rpl4 *rpl3 *rps14 *rpl19 rpl11 rpl1 rpl12 *rps16 *rps6 |C;

*rpl34 rpl32 rpl21 rpl27 *rps4 *rpl28 *rps2 *rpoBrpl33 rps18 *rpl20 *rpl35 *rps10 *rps7 *rps12 *rpl31 *rps9 *rpl13 rpoA*rps11 *rps13 *rpl36 *rps5 *rpl18 *rpl6 *rps8 *rpl5 *rpl24 *rpl14 *rpsl7 *rpl29 *rpl16 *rps3 *rpl22 *rpl2 *rpl23 *rpl4 *rpl3 *rps14 *rpl19 rpl11 rpl1 rpl12 *rps16 *rps6 |C.

The algorithm solves the problem with a minimal cost of 31.5. All vertices, except for vertex no. 4, were assumed a single circular chromosome; vertex no. 4 was assumed to be two circular chromosomes. The solution is given below and arranged in the order of vertex enumeration as follows:

rpl28 *rps2 *rpoBrpl33 rps18 *rpl20 *rpl35 rps14 rpl3 rpl4 rpl23 rpl2 rpl22 rps3 rpl16 rpl29 rps17 rpl14 rpl24 rpl5 rps8 rpl6 rpl18 rps5 rpl36 rps13 rps11 rpoArpl13 rps9 rpl31 rps12 rps7 rps10 *rpl19 rpl11 rpl1 rpl12 *rps16 *rps6 *rpl27 *rpl21 *rpl32 rpl34 *rps4 |C;

rpl28 *rps2 *rpoBrpl33 rps18 *rpl20 *rpl35 rps14 rpl3 rpl4 rpl23 rpl2 rpl22 rps3 rpl16 rpl29 rps17 rpl14 rpl24 rpl5 rps8 rpl6 rpl18 rps5 rpl36 rps13 rps11 rpoArpl13 rps9 rpl31 rps12 rps7 rps10 *rpl19 rpl11 rpl1 rpl12 *rps16 *rps6 *rpl27 *rpl21 *rpl32 rpl34 *rps4 |C;

rpl28 *rps2 *rpoBrpl33 rps18 *rpl20 *rpl35 *rps10 *rps7 *rps12 *rpl31 *rps9 *rpl13 *rpoA *rps11 *rps13 *rpl36 *rps5 *rpl18 *rpl6 *rps8 *rpl5 *rpl24 *rpl14 *rps17 *rpl29 *rpl16 *rps3 *rpl22 *rpl2 *rpl23 *rpl4 *rpl3 *rps14 *rpl19 rpl11 rpl1 rpl12 *rps16 *rps6 *rpl34 rpl32 rpl21 rpl27 *rps4 |C;

rpl28 *rps4 |C and rps2 rpl21 rpl27 rpl32 rpl34 rps6 rps16 *rpl12 *rpl1 *rpl11 rpl19 rps14 rpl35 rpl20 rpl3 rpl4 rpl23 rpl2 rpl22 rps3 rpl16 rpl29 rps17 rpl14 rpl24 rpl5 rps8 rpl6 rpl18 rps5 rpl36 rps13 rps11 rpoArpl13 rps9 rpl31 rps12 rps7 rps10 *rps18 *rpl33 rpoB |C;

rpl28 rps4 *rpl27 *rpl21 *rpl32 rpl34 rps6 rps16 *rpl12 *rpl1 *rpl11 rpl19 rps14 rpl3 rpl4 rpl23 rpl2 rpl22 rps3 rpl16 rpl29 rps17 rpl14 rpl24 rpl5 rps8 rpl6 rpl18 rps5 rpl36 rps13 rps11 rpoArpl13 rps9 rpl31 rps12 rps7 rps10 rpl35 rpl20 *rps18 *rpl33 rpoBrps2 |C.

Testing the algorithm with different binary resolutions of the polytomic tree showed that the best result was achieved at the same binary tree (Fig. 6b). A minimal cost of 29.6 was obtained. All vertices, except for vertex no. 7, were assumed to be one circular chromosome, whereas the vertex no. 7 was assumed as two circular chromosomes. The mapping of structures to internal vertices of the tree was ordered according to their enumeration as follows:

rpl28 *rps2 *rpoBrpl33 rps18 *rpl20 *rpl35 *rps10 *rps7 *rps12 *rpl31 *rps9 *rpl13 *rpoA*rps11 *rps13 *rpl36 *rps5 *rpl18 *rpl6 *rps8 *rpl5 *rpl24 *rpl14 *rps17 *rpl29 *rpl16 *rps3 *rpl22 *rpl2 *rpl23 *rpl4 *rpl3 *rps14 *rpl19 rpl11 rpl1 rpl12 *rps16 *rps6 *rpl34 rpl32 rpl21 rpl27 *rps4 |C;

rpl28 *rps2 *rpoBrpl33 rps18 *rpl20 *rpl35 rps14 rpl3 rpl4 rpl23 rpl2 rpl22 rps3 rpl16 rpl29 rps17 rpl14 rpl24 rpl5 rps8 rpl6 rpl18 rps5 rpl36 rps13 rps11 rpoArpl13 rps9 rpl31 rps12 rps7 rps10 *rpl19 rpl11 rpl1 rpl12 *rps16 *rpl27 *rpl21 *rpl32 rpl34 *rps4 |C;

rpl28 *rps2 *rpoBrpl33 rps18 *rpl20 *rpl35 *rps10 *rps7 *rps12 *rpl31 *rps9 *rpl13 *rpoA *rps11 *rps13 *rpl36 *rps5 *rpl18 *rpl6 *rps8 *rpl5 *rpl24 *rpl14 *rps17 *rpl29 *rpl16 *rps3 *rpl22 *rpl2 *rpl23 *rpl4 *rpl3 *rps14 *rpl19 rpl11 rpl1 rpl12 *rps16 *rps6 *rpl34 rpl32 rpl21 rpl27 *rps4 |C;

rpl28 *rps2 *rpoBrpl33 rps18 *rpl20 *rpl35 *rps10 *rps7 *rps12 *rpl31 *rps9 *rpl13 *rpoA *rps11 *rps13 *rpl36 *rps5 *rpl18 *rpl6 *rps8 *rpl5 *rpl24 *rpl14 *rps17 *rpl29 *rpl16 *rps3 *rpl22 *rpl2 *rpl23 *rpl4 *rpl3 *rps14 *rpl19 rpl11 rpl1 rpl12 *rps16 *rps6 *rpl34 *rpl32 rpl21 rpl27 *rps4 |C;

rpl28 *rps2 *rpoBrpl33 rps18 *rpl20 *rpl35 rps14 rpl3 rpl4 rpl23 rpl2 rpl22 rps3 rpl16 rpl29 rps17 rpl14 rpl24 rpl5 rps8 rpl6 rpl18 rps5 rpl36 rps13 rps11 rpoArpl13 rps9 rpl31 rps12 rps7 rps10 *rpl19 rpl11 rpl1 rpl12 *rps16 *rpl34 rpl32 rpl21 rpl27 rps6 *rps4 |C;

rpl28 *rps2 *rpoBrpl33 rps18 *rpl20 *rpl35 rps14 rpl3 rpl4 rpl23 rpl2 rpl22 rps3 rpl16 rpl29 rps17 rpl14 rpl24 rpl5 rps8 rpl6 rpl18 rps5 rpl36 rps13 rps11 rpoArpl13 rps9 rpl31 rps12 rps7 rps10*rpl19 rpl11 rpl1 rpl12 *rps16 *rps6 *rpl27 *rpl21 *rpl32 rpl34 *rps4 |C;

rpl28 *rps4 |C and rps2 rpl21 rpl27 rpl32 rpl34 rps6 rps16 *rpl12 *rp11 *rpl11 rpl19 rps14 rpl35 rpl20 rpl3 rpl4 rpl23 rpl2 rpl22 rps3 rpl16 rpl29 rps17 rpl14 rpl24 rpl5 rps8 rpl6 rpl18 rps5 rpl36 rpsl3 rps11 rpoArpl13 rps9 rpl31 rps12 rps7 rps10 *rps18 *rpl33 rpoB |C;

rpl28 rps4 *rpl27 *rpl21 *rpl32 rpl34 rps6 rps16 *rpl12 *rpl1 *rpl11 rpl19 rps14 rpl3 rpl4 rpl23 rpl2 rpl22 rps3 rpl16 rpl29 rps17 rpl14 rpl24 rpl5 rps8 rpl6 rpl18 rps5 rpl36 rps13 rps11 rpoArpl13 rps9 rpl31 rps12 rps7 rps10 rpl35 rpl20 *rps18 *rpl33 rpoBrps2 |C.

ACKNOWLEDGMENTS

We are grateful to A. V. Troitsky and S. A. Spirin for their useful comments that significantly improved our work. The study was supported by the Russian Scientific Fund (project no. 14-50-00150).

REFERENCES

1. Donthu R., Lewin H.A., Larkin D.M. 2009. SyntenyTracker: A tool for defining homologous synteny blocks using radiation hybrid maps and whole-genome

sequence. *BMC Res. Notes*. **23** (2), 148. doi 10.1186/1756-0500-2-148

2. Romanov M.N., Farré-Belmonte M., Lithgow P.E., O'Connor B., Fowler K.E., Larkin D.M., Griffin D.K. 2014. In silico reconstruction of chromosomal rearrangements and an avian ancestral karyotype. In: *XXII International Plant and Animal Genome Conference, January 11–16, 2014, San Diego, CA, USA*.
3. Romanov M.N, Farré M., Lithgow P.E., Fowler K.E., Skinner B.M., O'Connor R., Fonseka G., Backström N., Matsuda Y., Nishida C., Houde P., Jarvis E.D., Ellegren H., Burt D.W., Larkin D.M., Griffin D.K. 2014. Reconstruction of gross avian genome structure, organization and evolution suggests that the chicken lineage most closely resembles the dinosaur avian ancestor. *BMC Genomics*. **15**, 1060. doi 10.1186/1471-2164-15-1060
4. Bergeron A., Mixtacki J., Stoye J. 2006. A unifying view of genome rearrangements. *Algorithms Bioinform. LNCS*. **4175**, 163–173.
5. Fertin G., Labarre A., Rusu I., Tannier E., Vialette S. 2009. *Combinatorics of Genome Rearrangements*. Cambridge, MA: MIT Press.
6. *Models and Algorithms for Genome Evolution*. 2013. Eds. Chauve C., El-Mabrouk N., Tannier E. Comput. Biol. Series. London: Springer.
7. Lyubetsky V.A., Gorbunov K.Yu. 2013. Problems and algorithms related to chromosomal rearrangements. In: *Sbornik izbrannykh trudov VIII Mezhdunarodnoi nauchno-prakticheskoi konferentsii. MGU im. M.V. Lomonosova, 8–10 noyabrya 2013 g.* (Proc. 8th Int. Sci.-Pract. Conf., Moscow State Univ., November 8–10, 2013), Moscow: INTUIT.RU, pp. 764–768.
8. Lyubetsky V.A., Gorbunov K.Yu. 2014. Chromosome structures reconstruction. In: *Sbornik materialov 4-i Moskovskoi mezhdunarodnoi konferentsii "Molekulyarnaya filogenetika MolPhy-4". MGU im. M.V. Lomonosova, 23–26 sentyabrya 2014 g.)* (Proc. 4th Moscow Int. Conf. on Molecular Phylogenetics, MolPhy-4, Moscow State Univ., September 23–36, 2014). Moscow: Torus Press, p. 42.
9. Gorbunov K.Yu., Lyubetsky V.A. 2009. Reconstructing the evolution of genes along the species tree. *Mol. Biol*. (Moscow). **43** (5), 881–893.
10. Lyubetsky V.A., Rubanov L.I., Rusin L.Y., Gorbunov K.Yu. 2012. Cubic time algorithms of amalgamating gene trees and building evolutionary scenarios. *Biol. Direct*. **7** (1), 1–20.
11. Rusin L.Y., Lyubetskaya E.V., Gorbunov K.Yu., Lyubetsky V.A. 2014. Reconciliation of gene and species trees. *Biomed. Res. Int*. 642089. doi 10.1155/2014/642089

Translated by S. Khoronenkova