



Three-Dimensional Cost-Matrix Optimization and Maximum Cospeciation

Fredrik Ronquist

Department of Zoology, Uppsala University, Villavägen 9, SE-752 36 Uppsala, Sweden

Accepted 12 May 1997

In recent years, event-based approaches have been gaining ground in coevolutionary and biogeographical inference. Unlike pattern-based methods, event-based protocols deal directly with evolutionary events, such as dispersals and host switches. Three protocols have been proposed to date: (1) a coevolutionary method based on optimization of a standard two-dimensional cost matrix; (2) dispersal–vicariance analysis, based on optimization of a three-dimensional cost matrix; and (3) the maximum cospeciation method, thus far not considered a cost matrix method. I describe here general three-dimensional cost matrix optimization algorithms and how they can be applied to the maximum cospeciation problem. The new algorithms demonstrate that all existing event-based protocols, as well as possible future methods based on more complicated process models, can be incorporated into the three-dimensional cost matrix optimization framework. © 1998 The Willi Hennig Society

INTRODUCTION

In recent years, event-based approaches have been gaining ground in coevolutionary and biogeographical inference. Unlike pattern-based methods, event-based protocols are derived from explicit process models.

Different types of events are identified and assigned costs or benefit values. The history of species–area or species–species associations is then inferred by searching for the minimum-cost or maximum-benefit reconstruction.

Event-based protocols proposed to date include: (1) the coevolutionary two-dimensional cost matrix method developed by Ronquist and Nylin (Ronquist and Nylin, 1990; Ronquist, 1996b); (2) dispersal–vicariance analysis (DIVA), a biogeographical method based on optimization of a three-dimensional step matrix derived from a simple biogeographical model (Ronquist, 1996a, 1997); and (3) the “maximum cospeciation” method (MC) proposed by Page (1995a,b).

MC recognizes four different types of events in its model of coevolving associations: cospeciations, sorting events, duplications and host switches. At any single point in time, a parasite is assumed to be restricted to a single host. Furthermore, hosts shifts are assumed to be tied to speciation (these speciations cannot then be cospeciations), and only one of the daughter species resulting from such a speciation is allowed to shift to a new host. Under these constraints, the task is to reconstruct the history of associations by maximizing the number of cospeciations between the parasites and their hosts.

Page’s (1995a) original MC optimization algorithms work with alternative mappings of host nodes onto parasite nodes, trying to find the best fit between the

trees. Here, I present cospeciation-maximizing algorithms that are based on preliminary-pass and final-pass optimization in the parasite tree of a three-dimensional cost matrix derived from the host tree. The cost matrix algorithms have several attractive features. First, they show that MC can be considered a cost-matrix optimization method like other event-based methods of coevolutionary and biogeographic inference. Second, the cost matrix algorithms allow complete control over contradictory combinations of host switches, whereas Page's original algorithms do not (Ronquist, 1996b). Third, the cost matrix algorithms are fast, allowing exact solutions for large and difficult problems, and precise evaluation of statistical tests based on optimization of numerous randomized data sets.

COST MATRIX OPTIMIZATION

Cost matrix optimization algorithms consider one triplet of nodes at a time (Fig. 1B). When calculating the cost of a particular ancestral state assignment, the lengths of the left and right descendant branches are usually evaluated separately and then summed. Thus it is sufficient if the cost matrix contains all possible combinations of one ancestral state with one descendant state (a two-dimensional matrix). In MC and DIVA, however, cospeciations and vicariance events are of special importance. These events cannot be recognized unless both descendants are considered simultaneously. Thus, MC and DIVA require three-dimensional cost matrices, in which each cell specifies the cost of a particular combination of ancestral, left descendant, and right descendant states. Since the left and right descendants are equivalent, such a matrix is necessarily symmetric in one plane, but it is asymmetric in that the ancestor cannot replace any of the descendants.

For mapping purposes, all internodes in the host tree are labelled (Fig. 1A), with each node being considered to belong to the internode below it. This naming convention is preferable to just recognizing host nodes, since some events concern host nodes while others map to internodes in the host tree.

Cost matrices usually contain cost values which are minimized on a particular tree. Since the matrix

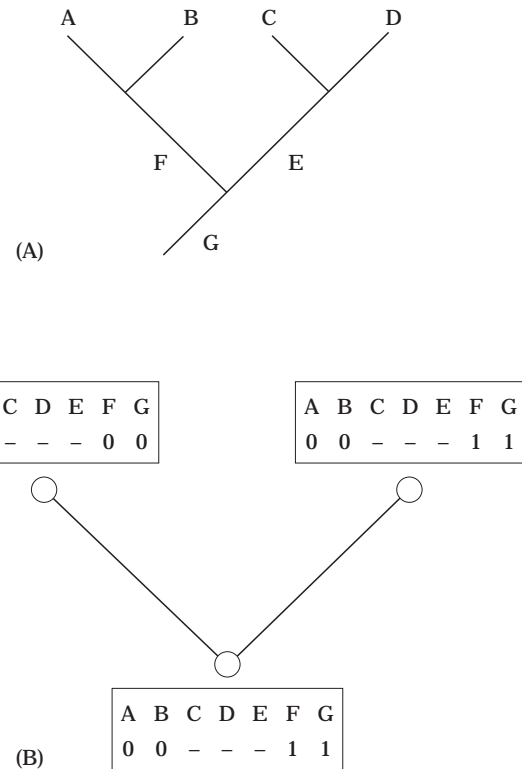


FIG. 1. Preliminary pass of a three-dimensional cost matrix optimization. (A) Host tree. (B) Three adjacent nodes in the parasite tree. Each node is associated with an array specifying the maximum benefit (in cospeciations) of particular host assignments. The preliminary-pass algorithm determines the array of the ancestral node given the arrays of the descendant nodes. Hosts marked with “-” are not possible states at the node (associated with an infinite negative benefit).

needed for MC records an entity (the number of cospeciations) that will be maximized, I will be referring to it as a “benefit matrix” containing “benefit values”. However, benefit values are just negative cost values, and maximizing benefit is equivalent to minimizing negative cost.

DERIVING THE BENEFIT MATRIX

The values in the benefit matrix are determined using four simple rules.

1. If the host of the ancestral parasite is the most recent common ancestor of the hosts of the

descendant parasites, and the hosts of the descendant parasites are not identical or in ancestor–descendant relationship to each other, the combination allows cospeciation and the benefit is 1.

2. If the host of the ancestral parasite is ancestral or identical to both of the hosts of the descendant parasites, and cospeciation is not possible, the combination implies duplication and the benefit is 0.
3. If the host of the ancestral parasite is ancestral or identical to only one of the hosts of the descendant parasites, and not a descendant of the host of the other parasite, the combination implies a host shift and the benefit is 0.
4. In all other cases, the combination is not allowed (it is assigned an infinite negative benefit).

These rules are so simple that the benefit values can be calculated as they are needed during the optimization. Very little time, if any, is lost in the optimization, and the computer memory that would have been required to store the matrix will be available for other purposes.

The preliminary and final pass algorithms are similar to those used in standard two-dimensional cost matrix optimization (Sankoff, 1975; Sankoff and Rousseau, 1975; Sankoff and Cedergren, 1983). First, each node in the parasite tree is assigned an array consisting of all possible hosts and an associated benefit value for each (Fig. 1B). These benefit arrays start out as empty, and will then in turn be filled with preliminary-pass and final-pass values.

THE PRELIMINARY PASS

The preliminary-pass algorithm is recursive, so it is sufficient to describe it for a set of three nodes, one being the immediate ancestor of the other two (Fig. 1). First, the cost arrays of the terminals are filled with an infinite cost for all states except the observed host, which is assigned a cost of 0. The tree is then traversed from the top towards the root, so we can safely assume that the benefit arrays of the descendant nodes have already been obtained. The benefit of a particular

ancestral state is calculated by taking each possible combination of descendant states in turn. The benefits already obtained for these descendant state assignments are added to the benefit of the particular combination of ancestral and descendant states being considered. When all combinations of descendant states have been tried for a particular ancestral state, the maximum benefit of that ancestral state assignment is known and can be stored in the benefit array of the ancestral node.

Consider a simple example (Fig. 1). To obtain the maximum benefit of F being the ancestral host, first assume that the host of both descendants is A. The combination is possible but does not allow cospeciation, so the benefit is obtained by simply summing the benefit of the two As, i.e. $0+0=0$. Next, combine A on the left with B on the right. The combination allows cospeciation, the benefit of F being $0+0+1=1$, which is better than the preceding combination. When all possible combinations of descendant hosts have been tried, the maximum benefit of the ancestral host being F is known, and we turn to the next ancestral state (G).

The preliminary pass is complete when the benefit array of the root node in the parasite tree has been calculated. The maximum number of cospeciations for any host in the root array is also the maximum number of cospeciations in the entire tree. If this is the only value needed, no further calculations are necessary. The final pass algorithm is only required to obtain all equally optimal node assignments throughout the tree.

THE FINAL PASS

Before the final pass, the cost array of the root node is converted from preliminary-pass to final-pass values by assigning a benefit of zero to all hosts having the maximum number of cospeciations and an infinite negative benefit to all other states. The final pass is a recursive preorder traversal through the tree (from root to terminals), so it is sufficient to define it for one triplet of nodes. The final-pass array of the ancestral node has already been obtained; the task is to update the arrays of the descendent nodes from preliminary-pass to final-pass values.

Allocate a temporary array for each descendant node, recording the states that are optimal for that

node. Take the ancestral states in turn. We only have to consider the optimal states, since the others have an infinite negative benefit in the final-pass array determined already. Each possible combination of descendant states is then tried. The preliminary-pass values of the descendant states, the final-pass value of the ancestral state, and the benefit of the particular combination of ancestral and descendant states being considered (according to the benefit matrix) are added. If the sum is equivalent to the maximum number of cospeciations, the descendant states are recorded to be optimal. When all combinations of ancestral and descendant states have been considered, we know the descendant states that are optimal. The final-pass values for these states are obtained by subtracting their preliminary-pass values from the maximum number of cospeciations. The non-optimal states are assigned an infinite negative benefit.

Consider a simple example (Fig. 2). F is the only optimal ancestral state, and has a final-pass benefit of 4. Assume that the maximum number of cospeciations over the entire tree is 5. We first combine F with A in both descendants. No cospeciation is possible, so the total benefit is $0+0+4=4$, which is smaller than the maximum. Next, combine A on the left with B on the right. This combination allows cospeciation, and the benefit is $0+0+4+1=5$. Since this is equivalent to the maximum number of cospeciations, A is an optimal state for the left descendant and B is an optimal state for the right descendant. When all possible combinations of descendant and ancestral states have been considered, the arrays of the descendants are updated to contain final-pass values.

Once the final-pass arrays and optimal states of all ancestral parasite nodes have been determined, the specific cospeciation, dispersal, sorting, and duplication events that each optimal reconstruction implies are easily obtained by simple algorithms that will not be discussed here.

IMPROVING SPEED

The preliminary pass of a three-dimensional cost matrix optimization, as described above, completes in time proportional to mn^3 , where m is the number of ancestral nodes in the parasite tree and n the number of

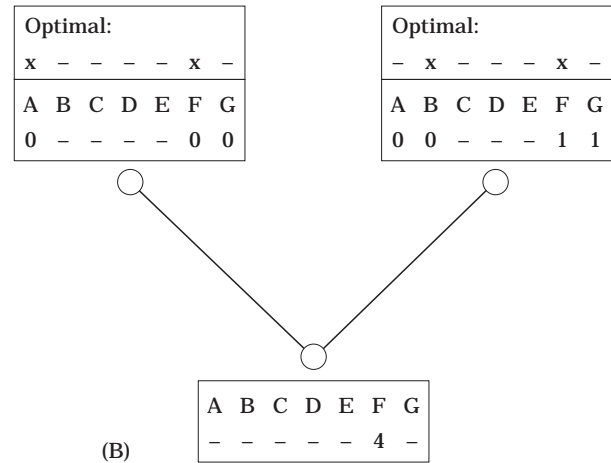
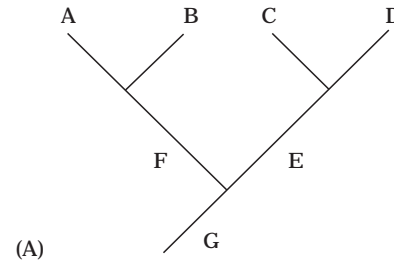


FIG. 2. Final pass of a three-dimensional cost matrix optimization. (A) Host tree. (B) Three adjacent nodes in the parasite tree. The final-pass array of the ancestral node and the preliminary-pass arrays of the descendant nodes are given. The final-pass algorithm updates the arrays of the descendant nodes to contain final-pass values. In the process, a temporary array is needed for each of the descendant nodes; it records the host assignments which are optimal for the node. Hosts marked with “-” are impossible states or states that cannot be optimal at the node.

nodes in the host tree. This is considerably more time-consuming than two-dimensional cost matrix optimization, which requires time proportional to $2mn^2$ (Williams and Fitch, 1990). However, by using special properties of the MC problem, it is possible to bring down the time consumption of preliminary-pass MC optimization considerably.

Instead of inferring the states of the vertices in the tree, consider the problem of inferring the states of points slightly below the real vertices. The arrays of the subterminal points will have a zero benefit for the observed host *and* all hosts ancestral to it. The first benefit-assignment rule of MC is modified by replacing “the most recent common ancestor” with “any common ancestor”. Now, the conditions are such that a

host cannot have a lower benefit than any of its descendants. By taking this into account, we only need to consider the following combinations of descendant states for each ancestral state.

1. Combinations allowing cospeciation. In the host tree, locate the immediate descendants of the host being considered for the ancestral parasite node. Combine these hosts states in the descendant parasite nodes: there are two possibilities depending on what host maps onto what parasite. Several other combinations of descendant states may allow cospeciation, but they cannot have a larger benefit.
2. Combinations implying duplication. Combine the ancestral parasite state with the same state in both descendants (one possibility). Several other combinations may imply duplication, but they cannot have a larger benefit.
3. Combinations implying host switches. Fix the state of the left descendant to the same host as the ancestor. For the right descendant, consider each host being an immediate descendant of a node ancestral to the fixed host in the host tree, but not identical or ancestral to the fixed host. Then repeat the procedure but shift left and right descendants. Several other combinations may require host shifts, but they cannot have a larger benefit.

Consider the example in Fig. 1. Rather than assessing all combinations of descendant states to determine the maximum benefit of the ancestor having host F, it is sufficient to examine the following possibilities. First, combine A on the left with B on the right, and vice versa, to obtain the maximum benefit assuming cospeciation. Then combine F on the right with F on the left to obtain the maximum benefit assuming duplication. Finally, combine F on the left with E on the right, and vice versa, to obtain the maximum benefit assuming a host switch.

The refined preliminary-pass algorithm only requires $2m(n^2/4+n+1/4)$ comparisons in the worst case, which (assuming n is large) is roughly one-fourth of the comparisons needed for standard optimization of a two-dimensional cost matrix with the same number of states. If the maximum number of cospeciations is the only parameter of interest, it is also possible to skip the switches between sister hosts, bringing down the worst-case number of comparisons to $2m(n^2/4+5/$

4). The final-pass algorithm needs to be modified to fit the state assignments of the quick preliminary pass, but this has little effect on the overall time consumption.

CONTRADICTORY HOST SHIFTS

The problem of contradictory host shifts in MC optimization occurs because some sequences or combinations of host shifts imply time reversals. For instance, it is impossible to shift from a host onto one of its ancestors. Yet, Page's original MC algorithms and the cost matrix algorithms described above allow such shifts if they occur via an intermediate host spanning a suitable time interval (Fig. 3A). In the context of cost matrix optimization, the problem can be solved by dividing long branches in the host tree into time segments (Fig. 3B) (Ronquist, 1996b). Time segmentation incurs a cost in that each segment has to be treated as a separate state in the cost matrix, increasing the computational effort needed to solve the optimization problem. However, it guarantees that contradictory combinations of host shifts are not postulated in the resulting reconstructions.

METRICITY

An important point concerns three-dimensional cost matrices and metricity. Metricity requires that $d(A,A)=0$ and $d(A,B)\geq 0$, where A and B are arbitrary states. This means that metricity is violated if duplications are not assigned zero cost, or if some events are assigned benefit values. Thus, the three-dimensional cost matrices required by both DIVA and MC are non-metric. Violation of metricity is a necessary consequence of cospeciations and vicariance events being considered more likely than duplications. Since duplications imply independent parasite speciation in coevolution and sympatric speciation in historical biogeography, metricity imposes constraints on coevolutionary or biogeographical inference that most biologists would be unwilling to accept.

As I have shown here, all event-based methods in coevolution and historical biogeography proposed to

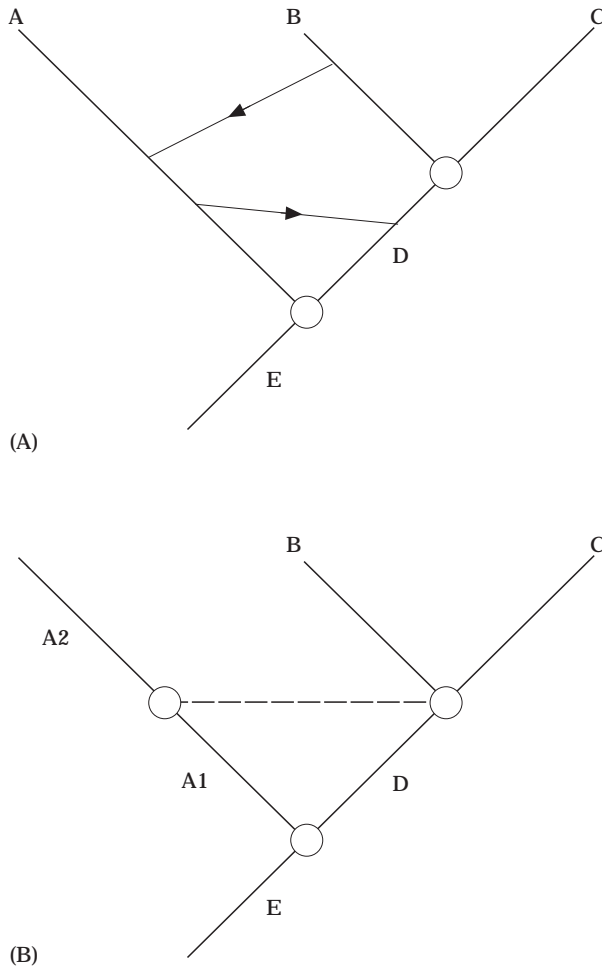


FIG. 3. (A) Ordinary host trees allow some impossible sequences or combinations of host switches. For instance, the illustrated tree prohibits shifts from B to its ancestor D. However, shifts from B to A and from A to D are allowed, making it possible to shift from B to D via A. (B) By dividing long internodes into time segments, it is possible to eliminate the inconsistency. Shifts between A1 and D, between A2 and B, and from A1 to A2 are allowed, but not shifts between A1 and B, between A2 and D, or from A2 to A1.

date can be incorporated in the cost matrix optimization framework. The models proposed so far are oversimplified, but the three-dimensional cost matrix framework is powerful enough to allow optimization based on virtually any conceivable coevolutionary or

biogeographical model, regardless of its complexity. For instance, in biogeographical inference it is possible to assign costs to dispersal events relative to the distance traversed (Ronquist, 1997) or impose constraints based on time-segmented reticulate biogeographical scenarios (cf. Ronquist, 1996a, 1997). In the future, three-dimensional cost matrix optimization will undoubtedly be used more frequently in dealing quantitatively and objectively with such complex biogeographical and coevolutionary models.

ACKNOWLEDGEMENTS

This research was supported by the Swedish Natural Science Research Council.

REFERENCES

- Page, R. D. M. (1995a). Parallel phylogenies: Reconstructing the history of host-parasite assemblages. *Cladistics* **10**, 155–173.
- Page, R. D. M. (1995b). "TreeMap, version 1.0". Computer program and manual available from <http://taxonomy.zoology.gla.ac.uk/rod/rod.html>. University of Glasgow, Glasgow.
- Ronquist, F. (1996a). "DIVA, version 1.1". Computer program and manual available from <ftp.systbot.uu.se> and www.systbot.uu.se/staff/f_ronquist.html. Uppsala University, Uppsala.
- Ronquist, F. (1996b). Reconstructing the history of host-parasite associations using generalised parsimony. *Cladistics* **11**, 73–89.
- Ronquist, F. (1997). Dispersal-vicariance analysis: A new approach to the quantification of historical biogeography. *Syst. Biol.* **46**, 193–201.
- Ronquist, F., and Nylin, S. (1990). Process and pattern in the evolution of species associations. *Syst. Zool.* **39**, 323–344.
- Sankoff, D. (1975). Minimal mutation trees of sequences. *SIAM J. Appl. Math.* **28**, 35–42.
- Sankoff, D., and Cedergren, R. J. (1983). Simultaneous comparison of three or more sequences related by a tree. In "Time Warps, String Edits, and Macromolecules" (D. Sankoff, and J. B. Kruskal, Eds), pp. 253–263. Addison-Wesley, Reading.
- Sankoff, D., and Rousseau, P. (1975). Locating the vertices of a Steiner tree in an arbitrary metric space. *Math. Program.* **9**, 240–246.
- Williams, P. L., and Fitch, W. M. (1990). Finding the minimum change in a given tree. In "The Hierarchy of Life" (B. Fernholm, K. Bremer, and H. Jörnvall, Eds), pp. 453–470. Elsevier, Amsterdam.