Polynomial algorithms for the Maximal Pairing Problem: efficient phylogenetic targeting on arbitrary trees

Christian Arnold[1,2], Peter F. Stadler[1,3,4,5,6,*]

[1]Bioinformatics Group, Department of Computer Science, and Interdisciplinary Center for Bioinformatics, University of Leipzig, Härtelstraße 16-18, D-04107 Leipzig, Germany
[2]Harvard University, Department of Human Evolutionary Biology, Peabody Museum, 11 Divinity Avenue, Cambridge MA 02138
[3]Max Planck Institute for Mathematics in the Sciences, Inselstraße 22, D-04103 Leipzig, Germany
[4]Fraunhofer Institute for Cell Therapy and Immunology, Perlickstraße 1, D-04103 Leipzig, Germany
[5]Santa Fe Institute, 1399 Hyde Park Rd., Santa Fe, NM 87501, USA
[6]Institute for Theoretical Chemistry, University of Vienna, Währingerstraße 17, A-1090 Wien, Austria

Email: Peter F. Stadler - studla@bioinf.uni-leipzig.de;

[*]Corresponding author

Abstract

Background: The *Maximal Pairing Problem* (MPP) is the prototype of a class of combinatorial optimization problems that are of considerable interest in bioinformatics: Given an arbitrary phylogenetic tree T and weights ω_{xy} for the paths between any two pairs of leaves (x, y), what is the collection of edge-disjoint paths between pairs of leaves that maximizes the total weight? Special cases of the MPP for binary trees and equal weights have been described previously. Algorithms to solve the general MPP are still missing, however.

Results: We describe a relatively simple dynamic programming algorithm for the special case of binary trees. We then show that the general case of multifurcating trees can be treated by interleaving solutions to certain auxiliary Maximum Weighted Matching problems with an extension of this dynamic programming approach, resulting in an overall polynomial-time solution of complexity $\mathcal{O}(n^4 \log n)$ w.r.t. the number n of leaves. For binary trees, we furthermore discuss several constrained variants of the MPP as well as a partition function approach to the probabilistic version of the MPP.

Conclusions: The algorithms introduced here make it possible to solve the MPP also for large trees with high-degree vertices. This has practical relevance in the field of comparative phylogenetics and, for example, in the context of phylogenetic targeting, i.e., data collection with resource limitations.

Availability: The source code of a C implementation of the polynomial algorithm can be obtained under the GNU Public License from http://www.bioinf.uni-leipzig.de/Software/Targeting.

Background

Comparisons among species are fundamental to elucidate evolutionary history. In evolutionary biology, for example, they can be used to detect character associations [1–3]. In this context, it is important to use statistically independent comparisons, i.e., any two comparisons must have disjoint evolutionary histories (*phylogenetic independence*). The *Maximal Pairing Problem* (MPP) is the prototype of a class of combinatorial optimization problems that models this situation: Given an arbitrary phylogenetic tree T and weights ω_{xy} for the paths between any two pairs of leaves (x, y) (representing a particular comparison), what is the collection of pairs of leaves with maximum total weight so that the connecting paths do not intersect in edges?

Algorithms for special cases of the MPP that are restricted to binary trees and equal weights (which thus simply maximizes the number of pairs) have been described, but not implemented [2]. Since different pairs of taxa may contribute different amounts of information depending on various factors (e.g., their phylogenetic distance or the difference of their character states), the weighted version is of considerable practical interest. A particular question of this type is addressed by *phylogenetic targeting*, where one seeks to optimize the choice of species for which (usually expensive and time-consuming) data should be collected [4]. Phylogenetic targeting boils down to two separate tasks: (1) estimation of the weight ω_{xy} that measures the benefit or our amount of information contributed by including the comparison of species x with species y and (2) the identification of an optimal collection of pairs of species such that they represent independent measurements, i.e., the solution of the corresponding MPP. To date, the only publicly available software package for phylogenetic targeting [5] can handle multifurcating trees; however, it uses a brute force enumeration of subsets of children and hence scales exponentially in the maximal degree.

As a consequence of the ever-increasing amount of available sequence data, phylogenetic trees of interest continue to increase in size, and large trees with hundreds or even thousands of vertices are not an exception any more [6–9]. Most large phylogenies contain a substantial number of multifurcations representing uncertainties in the actual phylogenetic relationships. It appears worthwhile, therefore, to extend previous approaches to efficiently solve the MPP for multifurcating trees and arbitrary weights.

Algorithms

Definitions and Preliminaries

Let $T(V, E)$ be a rooted (unordered) tree with vertex set $V = L \cup J$, where L are the leaves of T, J its interior vertices, $|J|$ the number of interior vertices, $|L|$ the number of leaves, and a single vertex $r \in J$ with degree 2, which we call the the root of T. (Note that, given an unrooted tree without vertices of degree 2, we can obtain a rooted tree by subdividing an arbitrary edge with the degree-2 vertex r.)

Every vertex x, with the exception of the root r, has a unique *father*, $\text{fa}(x)$, which is the neighbor of x closest to the root. We set $\text{fa}(r) = \emptyset$. Furthermore, for each $u \in J$, let $\text{chd}(u)$ be the set of children of u (i.e., its descendants). Obviously, $y \in \text{chd}(u)$ if and only if $\text{fa}(y) = u$ and $\text{chd}(u) = \emptyset$ if and only if $u \in L$. We write $T[u]$ for the subtree rooted at u. Furthermore, we assume that $|\text{chd}(u)| \neq 1$ throughout this contribution. A tree is binary if $|\text{chd}(u)| = 2$ for all $u \in J$, and multifurcating if $|\text{chd}(u)| > 2$ holds for some interior vertices. Finally, let $T[u, C]$ be the subtree of T rooted at an interior vertex $u \in J$, but with only a subset C of its children. All subtrees $T[v]$ with $v \in chd(u) \setminus C$ are thus excluded from $T[u, C]$.

For the purpose of this contribution, we interpret a *path* π in T as a sequence $\{e_1, \ldots, e_l\}$ of edges $e_i \in E$ such that $e_i = e_j$ implies $i = j$ and $e_i \cap e_{i+1} = \{x_i\}$ are single vertices for all $1 \leq i < l$. The vertices $x_0 \in e_1$ and $x_l \in e_l$ are the endpoints of π. For two vertices $x, y \in V$, we denote the unique path with endpoints x and y by π_{xy}. In the following, we will frequently be concerned with paths connecting an interior vertex $u \in J$ with a leaf $x \in L$. This path contains exactly one child of u, which we denote by $u_x = n(u, x)$. The array $n(u, x)$ allows efficient navigation in T.

A *path-system* Υ on T is a set of paths π such that

1. If $\pi = \pi_{xy} \in \Upsilon$, then $x, y \in L$ and $x \neq y$, i.e., every path connects two distinct leaves.
2. If $\pi' \neq \pi''$, then $\pi' \cap \pi'' = \emptyset$, i.e., any two paths in Υ are edge-disjoint.

Note that two paths in Υ have at most one vertex in common (otherwise they would also share the sub-path, and therefore edges, between two common vertices). In binary trees, two edge-disjoint paths are also vertex-disjoint (since two edge-disjoint paths can only run through a vertex of degree at least 4),

Fig. 1. Two edge-disjoint paths can share a vertex u in two distinct situations: (1) if both paths have u as the last common ancestor of their respective leaves, u must have at least four children, (2) if u is the last common ancestor for one path, while the other path also includes an ancestor of u, three children of u are sufficient. These two situations will also lead to distinct cases in the algorithms that are presented next.

Furthermore, let $\omega_{xy} : L \times L \to \mathbb{R}$ be an arbitrary weight function on the leaves of T. We define the weight of a path-system Υ as

$$\omega(\Upsilon) = \sum_{\pi_{xy} \in \Upsilon} \omega_{xy} \tag{1}$$

A path-system Υ that maximizes $\omega(\Upsilon)$, i.e., a solution of the `MPP`, will in the following be called *optimal path-system*. It conceptually corresponds to Maddison's "maximal pairing" [2], although we describe here a more general problem (see Background and Variants). In the following sections our main objective is to compute optimal path-systems.

The Maximal Pairing Problem for binary trees

Forward recursion

In this section we reconsider the approach of [4] for the special case of binary trees. This subsumes also Maddison's [2] discussion of the special unweighted case (see section Variants). We develop the dynamic programming solution for this class of MPP using a presentation that readily leads itself to the desired generalization to arbitrary trees.

For a given interior vertex $u \in J$ we use the abbreviation $C_x = C_x(u) = \mathrm{chd}(u) \setminus u_x$ for the set of children of u that are not contained in the path that connects u with the leaf x. Since T is binary by assumption in this subsection, C_x contains a unique vertex $C_x = \{u_{\bar{x}}\}$.

We will need two arrays (S, R) to store optimal solutions of partial problems. For each $u \in V$, let S_u be the score of an optimal path-system on the subtree $T[u]$. For each $u \in V$ and leaf $x \in T[u]$, we furthermore define R_{ux} as the score of the optimal path-system on $T[u]$ that is edge-disjoint with the path π_{ux}. R_{ux} can be decomposed as follows:

$$R_{ux} = R_{u_x x} + S_{u_{\bar{x}}} \tag{2}$$

For completeness, we set $S_x = R_{xx} = 0$ for all leaves $x \in L$.

An optimal path-system on $T[u]$ either consists of optimal path-systems on each of the two trees $T[v]$ and $T[w]$ rooted at the two children $v, w \in \mathrm{chd}(u)$, or it contains a path π_{xy} with endpoints $x \in T[v]$ and $y \in T[w]$. Thus, S_u can be calculated as follows:

$$S_u = \max \begin{cases} S_v + S_w \\ \max\limits_{x \in T[v]} \max\limits_{y \in T[w]} \{\omega_{xy} + R_{vx} + R_{wy}\} \end{cases} \tag{3}$$

Recursion (3) can then be evaluated from the leaves towards the root.

In order to facilitate the backtracing part of the algorithm, it is convenient to introduce an auxiliary variable F_u. If the optimal score in eq.(3) is obtained by the second alternative, the pair (x, y) that led to the highest score is recorded in F_u; otherwise, we set $F_u = \emptyset$.

Backtracing

An optimal path-system $\Upsilon_{\max}$ on $T = T[r]$ can be obtained by backtracing. For binary trees, this is straightforward. We start at the root r. In the general set, at an interior vertex u with $v, w \in \mathrm{chd}(u)$, we first check whether $F_u = \emptyset$. If this is the case, all paths $\pi_{xy} \in \Upsilon_{\max}$ are contained within the subtrees $T[v]$ and $T[w]$, and we continue to backtrace in both v and w. If $F_u = (x, y)$, then π_{xy} is added to $\Upsilon_{\max}$, and we need to backtrace the optimal path-system for each of the subtrees "hanging off" π_{xy}. In other words, we need optimal path systems for the subtrees rooted at the vertices $u_{\bar{x}}$ and $u_{\bar{y}}$ for $u \in \pi_{xy}$. These can be obtained recursively by following the decompositions of R_{vx} and R_{wy}, respectively, given in eq.(2).

Time and Space complexity

All entries S_u for interior vertices u can be computed in $\mathcal{O}(n^3)$ time, because a total of $n(n-1) \in \mathcal{O}(n^2)$ pairs of leaves have to be considered in eq.(3) and computation of each S_u entry takes at most $\mathcal{O}(n)$ time. Since we need to store the quadratic arrays R_{ux} and $n(u, x)$ as well as the linear arrays S_u and F_u, we need $\mathcal{O}(n^2)$ memory.

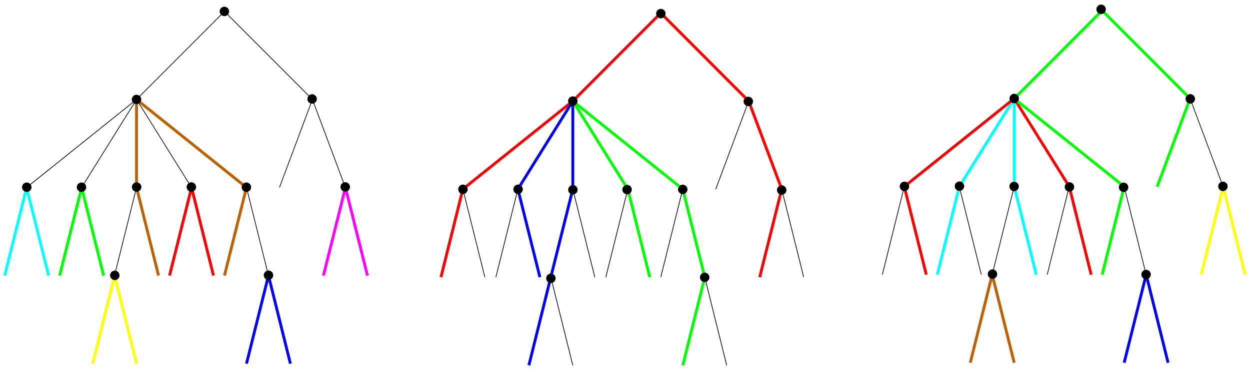

Figure 1: Three different path-systems on a tree with 15 leaves. Each path is shown in a distinctive color, unused edges of the tree are shown as thin black lines. Clearly, no two paths share an edge, i.e., the corresponding collection of pairs of leaves is phylogenetically independent. Note that the paths are not necessarily vertex-disjoint.

The Maximal Pairing Problem for multifurcating trees

Forward recursion

In trees with multifurcations, for a path-system Υ, more than one path can run through each vertex $m \in J$ with $|\mathrm{chd}(m)| > 2$. In addition to the optimal score S_u, we also define the optimal score Q_{ux} of all path-systems Υ'_u on $T[u] \setminus T[u_x]$, i.e., of all Υ'_u that avoid not only the path π_{ux} but the entire subtree $T[u_x]$, where u_x is as usual the child of u along π_{ux}. We therefore have

$$R_{ux} = R_{u_x x} + Q_{ux} \tag{4}$$

The computation of S_u and Q_{ux} are analogous problems. In general, consider an (interior) vertex $u \in J$ and a subset $C \subseteq \mathrm{chd}(u)$ of children of u. Our task is to compute the optimal path-system on the subtree $T[u, C]$ of T. We first observe that any path-system on $T[u, C]$ contains $0 \leq k \leq \lfloor |C|/2 \rfloor$ paths π_k through u. Each of these paths runs through exactly two distinct children v'_k and v''_k of u. For fixed v'_k and v''_k, the path ends in leaves $x'_k \in T[v'_k]$ and $x''_k \in T[v''_k]$ (Fig. 1). The best possible score contribution for the path $\pi_{x'x''}$ is

$$\tilde{Q}_{x',x''} = R_{v'x'} + R_{v''x''} + \omega_{x'x''} \tag{5}$$

and the best possible score for a particular pair of children $v', v'' \in C$ is therefore

$$\tilde{Q}_{v',v''} = \max_{x' \in T[v']} \max_{x'' \in T[v'']} \{R_{v'x'} + R_{v''x''} + \omega_{x'x''}\} \tag{6}$$

For the purpose of backtracing, it will be convenient to record the path π_{xy}, or rather its pair of end points (x, y), that maximized $\tilde{Q}_{v',v''}$ in eq.(6) in an auxiliary variable $F_{v',v''}$.

Since there are k paths through u covering $2k$ of the $|C|$ subtrees, there are $|C| - 2k$ children v_ℓ, $1 \leq \ell \leq |C| - 2k$, of u, each of which contributes to the optimal path-system a sub-path-system that is contained entirely within the subtree $T[v_\ell]$. Since these contributions are independent of each other, they are obtained by solving the MPP on $T[v_\ell]$, i.e., their contribution to the total score of the optimum path system is S_{v_ℓ}.

For each subtree $T[u, C]$ we therefore face the problem of determining the optimal combination of pairs and isolated children. This task can be reformulated as a weighted matching problem on an auxiliary graph $\Gamma(C)$ whose vertex set consists of two copies of the elements of C, denoted v and v^*. Within one copy of C, there is an edge between any two elements. The remaining $|C|$ edges of $\Gamma(C)$ connect each v with its copy v^*. The associated edge weights are $\omega_{v',v''} = \tilde{Q}_{v',v''}$ and $\omega_{v,v^*} = S_v$, respectively. An example is shown in Fig. 2.

Clearly, an optimal path of the form $x', \ldots, v', u, v'', \ldots, x''$ is represented by the edge (v', v'') of $\Gamma(C)$, while a self-contained subtrees $T[v]$ is represented by an edge of the form (v, v^*). It remains to be shown that every maximum matching of the auxiliary graph $\Gamma(C)$ corresponds to a legal conformation of paths, i.e., we have to show that in a maximum matching $\mathcal{M}$, each vertex $v \in C$ is

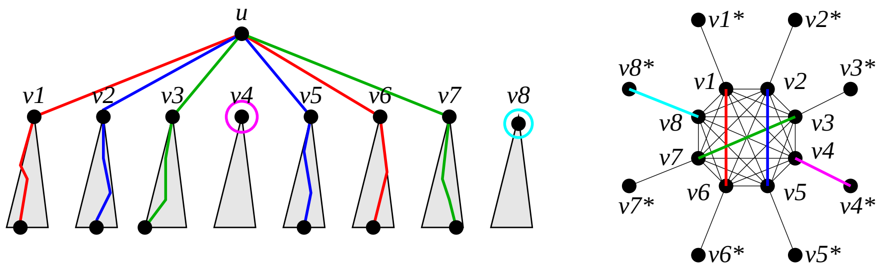

Figure 2: Translation of a path-system on $T[u]$ into a matching on the auxiliary graph $\Gamma(\mathrm{chd}(u))$. See text for details.

contained in an edge. First, note that v^* covered by an edge of $\mathcal{M}$ if and only if $(v, v^*) \in \mathcal{M}$. Suppose v is not covered in $\mathcal{M}$. Since ω_{v,v^*} is non-negative, we can exclude matchings that do not cover all edges of C from the solution set.

We can thus compute the entries of S_u and Q_{ux}, respectively, in polynomial time by solving maximum weighted matching problems with non-negative weights. Introducing the symbol $\mathtt{MWM}(\Gamma)$ for the maximum weight of a matching on the auxiliary graph Γ, we can write this as

$$\begin{aligned} S_u &= \mathtt{MWM}(\Gamma(\mathrm{chd}(u))) \\ Q_{ux} &= \mathtt{MWM}(\Gamma(\mathrm{chd}(u) \setminus \{u_x\})) \end{aligned} \tag{7}$$

Here we make use of the fact that the weight of a matching equals the sum of the weights of the path systems that correspond to the edges of the auxiliary graphs. In order to facilitate backtracing, we keep tabulated not only the weights but also the corresponding maximum matchings for each $\Gamma(\mathrm{chd}(u))$ and $\Gamma(\mathrm{chd}(u) \setminus \{u_x\}))$.

Backtracing

Backtracing for multifurcating trees proceeds in analogy to the binary case. Again we start from the root towards the leaves, treating each interior vertex u. If $|\mathrm{chd}(u)| = 2$, see the backtracing for the binary case. If $|\mathrm{chd}(u)| > 2$, we first need the solution $\mathcal{M}$ of the MWM for $\mathrm{chd}(u)$. For each edge $(v, v^*) \in \mathcal{M}$, v is called recursively to determine its optimal path-system. Each edge $(v', v'') \in \mathcal{M}$, however, represents a path π_{xy} that belongs to the optimal path-system. Each of these paths π_{xy} maximizes $\tilde{Q}_{v',v''}$ for a particular pair of children $v', v'' \in chd(u)$ and therefore has been stored in $F_{v'v''}$ during the forward recursion. Thus, each of these paths π_{xy} can be added to the optimal path-system.

As in the binary case, it remains to add the solutions from the optimal path-systems from the subtrees that are not on the path from x to v' and y to v'', respectively, for each particular edge $(v', v'') \in \mathcal{M}$. This can be done as follows. According to eqns.(2) and (4), $R_{v'x}$ can be decomposed into $R_{v'_x}$ and either $Q_{v'x}$ or $S_{\bar{v}'_x}$. If $|\mathrm{chd}(v')| = 2$, the child node $\bar{v}'_x = k$ that is not on the path from v' to x is called recursively to obtain the optimal path-system in $T[k]$. If $|\mathrm{chd}(v')| > 2$, however, the solution of the MWM for $Q_{v'x}$ is needed to determine an optimal path-system on the subtree $T[v'] \setminus T[v'_x]$, because multiple paths may go through v'. $R_{v'_x}$ can then be further decomposed until R_{xx} is reached. The same procedure is employed for $R_{v''y}$.

Time and Space complexity

A maximum weighted matching on arbitrary graphs with $|V|$ vertices and $|E|$ edges can be computed in $\mathcal{O}(|V||E| \log E)$ time and $\mathcal{O}(E)$ space by Gabow's classical algorithm [10] or one of several more recent alternatives [11, 12]. In our setting, $|E| \in \mathcal{O}(|\mathrm{chd}(u)|^2)$, hence the total memory complexity of our dynamic programming algorithm is $\mathcal{O}(n^2)$.

All entries for $\tilde{Q}_{v',v''}$ (the edge weights for the matching problems) can be computed in $\mathcal{O}(n^3)$ time, because a total of $n(n-1) \in \mathcal{O}(n^2)$ pairs of leaves have to be considered in eq.(6) and computation of each $\tilde{Q}_{v',v''}$ entry takes at most $\mathcal{O}(n)$ time. The effort for one of the $\mathcal{O}(|chd(u)|)$ maximum weighted matching problems $\mathtt{MWM}(\,.\,)$ for a given interior vertex u with more than two children is bounded by $\mathcal{O}(|chd(u)|^3 \log(|\mathrm{chd}(u)|)^2)$. The total effort for all MWMs is therefore bounded by

$$\sum_u |\mathrm{chd}(u)|^4 \log(|\mathrm{chd}(u)|^2) \in \mathcal{O}(n^4 \log n)\,,$$

which dominates the overall time-complexity of the algorithm (see Appendix for a derivation).

As in the binary case, $\mathcal{O}(n^2)$ space is necessary and sufficient to store the arrays R and S. Furthermore, $\mathcal{O}(n^2)$ space is needed to save the array Q and the endpoints (x, y) of the path π_{xy} that maximized each Q entry. The latter is needed for the backtracing. In addition, we keep the quadratic array $n(u, x)$ to allow efficient navigation in T. For each interior vertex u with $|\mathrm{chd}(u)| > 2$, $|\mathrm{chd}(u)| + 1$ different maximal matchings have to be stored: one that corresponds to S_u and $|\mathrm{chd}(u)|$ that correspond to Q_{ux}. Each of these solutions requires $\mathcal{O}(|\mathrm{chd}(u)|)$ space. The total space complexity of all `MWM` solutions is therefore $\sum_u |\mathrm{chd}(u)|^2 \in \mathcal{O}(n^2)$ (see Appendix).

Algorithmic variants

Several variants and special cases of the general `MPP` algorithm are readily derived for related problems. In the following, we briefly touch upon some of them.

Special weight functions

It is worth noting that finding a path-system that simply maximizes the number of pairs, as presented in [2] and applied in [13], for example, constitutes a special case of the MPP with unit weights. (Of course the same result is obtained by setting ω_{xy} to any fixed positive weight.) This case may be of practical use under certain circumstances, as it maximizes the number of independent measurements, thus improving power of subsequent statistical tests. In order to maximize the number of edges that are covered by the optimal path-system, we simply set $\omega_{xy} = d(x, y)$, where $d(x, y)$ is the graph-theoretic distance, i.e., we interpret the edge lengths in the tree as unity.

Fixed number of paths

A variant of practical interest is to limit the optimal path-system to κ leaf-pairs. This may be relevant in a phylogenetic targeting setting, for example, in cases where resources are limiting data acquisition efforts to a small number taxa so that it pays to make every effort to choose them optimally (see also [4]). Typically, κ will be small in this setting.

For binary trees, this variant can be implemented by conditioning the matrices R and S to a given number of paths. Eq.(2) thus becomes

$$R_{ux,k} = \max_{l \in \{0,k\}} \{R_{u_x x,l} + S_{u_{\bar{x}},k-l}\} \tag{8}$$

for a given number $k \le \kappa$ in the partial solutions. If the optimal path-system on $T[u]$ is composed of optimal path-systems on the two trees rooted at its children v and w, respectively, then the k paths are arbitrarily contained within $T[v]$ and $T[w]$. Thus, $k + 1$ different cases have to be considered, and the case with the highest score has to be identified. This yields to the following extension of eq.(3) for $S_{u,k}$:

$$S_{u,k} = \max \begin{cases} \max\limits_{l \in \{0,k\}} \{S_{v,l} + S_{w,k-l}\} \\ \max\limits_{\substack{l \in \\ \{0,k-1\}}} \max\limits_{\substack{x \in T[v] \\ y \in T[w]}} \{\omega_{xy} + R_{vx,l} + R_{wy,k-l}\} \end{cases} \tag{9}$$

We set $S_x = R_{xx,l} = R_{ux,0} = 0$ for all $x \in L$, $u \in J$, and $l \in \{0, k\}$. The latter condition ensures that if no path can be selected anymore in a particular subtree, its score must be 0.

As mentioned above, however, eq.(9) only holds for binary trees. For multifurcating trees, the auxiliary maximum weighted matching problems are replaced by the task of finding matchings that maximize the weight for a fixed number k of edges. We are, however, not aware that this variant of matching problems has been studied in detail so far. For small κ, it could of course be solved by brute force enumeration.

Selecting paths or taxa in addition to already selected paths or taxa

In some applications it may be the case that a subset of taxa or paths is already given, e.g. because the corresponding data have already been acquired in the past. The question then becomes how additional resources should be allocated. Let $Z \subset L$ denote the taxa that are required to appear in the output. Similarly, we may prescribe a partial path-system.

In the simpler case, we are given such a partial path-system Π. It then suffices to remove the corresponding leaves from T (to ensure that they are not selected again) and to set the weight of all paths that have edges in common with Π to $-\infty$ to enforce independence from the prescribed pairs.

The situation is less simple if only the taxa are given and the pairs are not prescribed. Here, the goal is to find an optimal path-system that includes all $z \in Z$. First we note that such a solution not necessarily exists, e.g. if $|Z| = n$ and n is odd. As a simple example, consider a binary tree with three leaves. In that case, only one path and thus two leaves can be selected. This constraint also holds for the subtree rooted at any interior vertex u, i.e., partial solutions of the MPP (see below).

For binary trees, this variant can be implemented by conditioning the matrices R and S to a subset of all possible paths and leaves. This is achieved by setting the score to $-\infty$ for a particular interior vertex if one of the preconditions cannot be met in eqns.(2) and (3). For example, if two leaves $x, y \in Z$ have the same father u, the optimal path-system of both $T[u]$ and T must contain the path π_{xy}, because otherwise, either x or y would not belong to the optimal path-system due to the requirement of independence. Similarly, if a particular path π_{xy} in the second alternative achieves the highest score in eq.(3), π_{xy} must not be selected if this conflicts with the possibility to select other prescribed leaves $z \in Z$ (Fig. 3).

To derive the recursions for this variant, let Z_u denote the leaves $z \in Z$ with $z \in T[u]$ and let L be the leaves of $T[u]$. It is convenient to first check whether a solution exists for $T[u]$. If $L = Z_u$ and $|L|$ is odd, $S_u = -\infty$ (i.e., no path-systems exists that selects all $z \in Z_u$ in $T[u]$). Otherwise, the optimal path-system for $T[u]$ with $v, w \in \mathrm{chd}(u)$ can be calculated as follows:

$$S_u = \max \begin{cases} \begin{cases} S_v + S_w & \text{if } v \notin Z \text{ and } w \notin Z \\ -\infty & \text{otherwise} \end{cases} \\ \\ \max\limits_{\substack{x \in T[v] \\ y \in T[w]}} \begin{cases} -\infty & \text{if } R_{u_x x} = -\infty \\ & \text{or } S_{u_{\bar{x}}} = -\infty \\ \omega_{xy} + R_{u_x x} + S_{u_{\bar{x}}} & \text{otherwise} \end{cases} \end{cases} \tag{10}$$

Furthermore,

$$R_{ux} = \begin{cases} -\infty \text{ if } R_{u_x x} = -\infty \text{ or } S_{u_{\bar{x}}} = -\infty \\ R_{u_x x} + S_{u_{\bar{x}}} \text{ otherwise} \end{cases} \tag{11}$$

and

$$S_x = \begin{cases} 0 & \text{if } x \notin Z \\ -\infty & \text{otherwise} \end{cases} \tag{12}$$

for any $x \in L$. In analogy to the algorithm for the unconstrained MPP we initialize the recursions by $R_{xx} = 0$ for $x \in L$. This variant does not change the overall time and space complexity. Backtracing is also identical to the unconstrained version of the MPP.

For multifurcating trees, the maximum weighted matching problems are replaced by finding matchings that maximize the weight with the constraint that particular vertices must be included in the matching. Similarly to the variant introduced above, however, we are not aware that this particular problem has been studied in detail.

Probabilistic version

Sometimes, not only the optimal solution is of interest. As in the case of sequence alignments [14] or biopolymer structures [15], one may analyze the entire ensemble of solutions. Both for physical systems such as RNA, and for alignments with a log-odds based scoring system, one can show that individual configurations Υ with score $S(\Upsilon)$, in our case path-systems, contribute to the ensemble proportional its Boltzmann weight $\exp(-\beta S(\Upsilon))$, where the "inverse temperature" β defines a natural scale that is implicitly given by the scoring or energy model. In the case of physical systems $\beta = 1/kT$ is linked to the ambient temperature T; for log-odds scores, $\beta = 1$; if the scoring scheme is rescaled, as e.g. in the case of the Dayhoff matrix in protein alignment, then β is the inverse of this scaling factor. In cases where schemes without a probabilistic interpretation are used, suitable values of β have to be determined empirically. The smaller β, the more the optimal path systems is emphasized in the ensemble. The *partition function* of the system is

$$Z = \sum_{\Upsilon} \exp(-\beta S(\Upsilon)) \,. \tag{13}$$

The probability p_Υ to pick Υ from the ensemble is $p_\Upsilon = \exp(-\beta S(\Upsilon))/Z$.

The recursion in eq.(3) can be converted into a corresponding recursion for the partition functions Z_u of path-systems on subtrees $T = T[u]$ because the decomposition of the score-maximization in unambiguous in the sense that every conformation falls into exactly of the case of recursion. This is a generic feature of dynamic programming algorithms that is explored in some depth in the theory of Algebraic Dynamic Programming [16]. We find

$$Z_u = Z_v \cdot Z_w + \sum_{x \in T[v]} \sum_{y \in T[w]} \exp(-\beta \omega_{xy}) \cdot R_{vx} \cdot R_{wy} \tag{14}$$

with $Z_u = 1$ if $u \in L$ and

$$R_{kx} = R_{k_x x} + Z_{k_{\bar{x}}} \tag{15}$$

for $k \in J$. Note that these recursions are completely analogous to the score optimization in eqns.(3) and

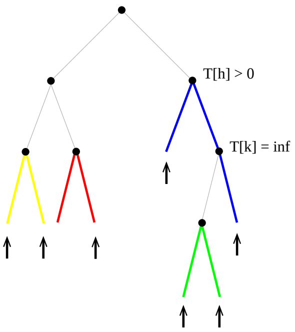

Figure 3: A binary tree for which only one possible path-system exists that fulfills all constraints. Leaves that must appear in the output are highlighted with an arrow, and the (only) valid path-system is displayed in color (left). Note that the score of the subtree $T[k] = \infty$, because no path-system in $T[k]$ exists that includes all three leaves $x \in T[k]$. The score of $T[k]$, however, is greater than 0.

(2): the max operator is replaced by a sum, and addition of scores is replaced by multiplication of partition functions and Boltzmann factors.

In order to compute the probability P_{xy} of a particular path π_{xy} in the ensemble we have to add up the contributions p_Υ of all path-systems that contain π_{xy}

$$Z(\pi_{xy}) := \sum_{\Upsilon \ni \pi_{xy}} \exp(-\beta\omega(\Upsilon)) \tag{16}$$

and compute the ratio $P_{xy} = Z(\pi_{xy})/Z$. The recursions for the restricted partition function $Z(\pi_{xy})$ can be computed in analogy to eq.(14), but with two additional constraints. First, since $\pi_{xy} \in \Upsilon$ by definition, the leaves $i \in T[v]$ and $j \in T[w]$ are constrained in eq.(14), because only paths π_{ij} that are edge-disjoint with π_{xy} can be considered. The recursion for the partition function of the last common ancestor node of x and y, denoted k, is also constrained, because π_{xy} must go through k. Calculation of the partition functions for the children of k is therefore not needed to compute Z_k. Thus,

$$Z_u = \begin{cases} \exp(-\beta\omega_{xy}) \cdot R_{vx} \cdot R_{wy} \text{ if } u = k \\ \\ Z_v \cdot Z_w + \\ \sum\limits_{\substack{i \in T[v], j \in T[w] \\ \pi_{xy} \cap \pi_{ij} = \emptyset}} \exp(-\beta\omega_{ij}) \cdot R_{vi} \cdot R_{wj} \text{ otherwise} \end{cases} \tag{17}$$

In resource requirements, this backward recursion is comparable to the forward recursion in eq.(3): $Z(\pi_{xy})$ and thus also P_{xy} can be calculated in $\mathcal{O}(n^3)$ time, because the number of leaf-pairs that have to be considered is still in $\mathcal{O}(n^2)$. There is an additional factor $\mathcal{O}(n)$ arising from the need to determine if the path π_{xy} is edge-disjoint with another path, which however does not increase overall time complexity. Furthermore, $\mathcal{O}(n^2)$ space is needed.

The computation of partition functions is a much more complex problem in the case of trees with multifurcations since it would require us in particular to compute partition functions for the interleaved matching problems. These are not solved by means of dynamic programming; instead, they use a greedy algorithm acting on augmenting paths in the auxiliary graphs. These algorithms therefore do not appear to give rise to efficient partition function versions.

The TARGETING software

We implemented the polynomial algorithms for the MPP in the program TARGETING. The TARGETING program is written in C and uses Ed Rothberg's implementation [17] of the Gabow algorithm [10] to solve the *Maximum Weight Matching Problem* on general graphs. The software also provides an user-friendly interface and can solve the special weight variants as well. The source code can be obtained under the GNU Public License at http://www.bioinf.uni-leipzig.de/Software/Targeting/.

Concluding Remarks

In this contribution, we introduced a polynomial algorithm for the *Maximal Pairing Problem* (MPP) as well as some variants. The efficient generalization of the dynamic programming approach to trees with multifurcations is non-trivial, since a straightforward approach yields run-times that are exponential in the maximal degree of the input tree. A polynomial-time algorithm can be constructed by interleaving the dynamic programming steps with the solution of auxiliary maximum weighted matching problems. This generalized algorithm for the MPP is implemented in the software package TARGETING, providing a user-friendly and efficient way to solve the MPP as well as some of its variants.

Future work in this area is likely to focus on developing algorithms for the variants of the MPP on multifurcating trees. In particular, the interleaving of dynamic programming for the MPP and the greedy approach for the auxiliary matching problems does not readily generalize to a partition function algorithm for arbitrary trees. The concept of unique matchings as discussed in [18] may be of relevance in this context.

The MPP solver presented here has applications in a broad variety of research areas. The method of phylogenetically independent comparisons relies on relatively few assumptions [1–3] and is frequently used in evolutionary biology, in particular in anthropology, comparative phylogenetics and, more generally, in studies that test evolutionary hypotheses [19–22]. As highlighted earlier, another application area lies in the design of studies in which tedious and expensive data collection is the limiting factor, so that a careful selection (phylogenetic targeting) becomes an economic necessity [5]. As noted in [13], alternative applications can be found in molecular phylogenetics, for example in the context of estimating relative frequencies of different nucleotide substitutions or the determination of the fraction of invariant sites in a particular gene.

Appendix

Pseudocode

Below, we include some pseudocode for the computation of the optimal path-system for an arbitrary tree T.

```
Require: $\omega_{xy} \geq 0$ $\forall$ pairs $x, y \in L$ and precomputed array $n(u,x)$ $\forall$ $u \in J$ and $x \in L$
$S_x = R_{xx} = Q_{x,x} = 0$ $\forall x \in L$
$R_{ux} = R_{u_x x} + S_{u_{\bar{x}}}$ if $|\mathrm{chd}(u)| = 2$ and $R_{ux} = R_{u_x x} + Q_{u_x,x}$ if $|\mathrm{chd}(u)| > 2$ $\forall u \in J$ and $x \in L$.
for all $u \in J$ in post-order tree traversal do
  if $|\mathrm{chd}(u)| = 2$ then
    $\{v, w\} \leftarrow chd(u)$
    $S_{u1} = S_v + S_w$
    for all paths $\pi_{xy}$ with $x \in T[v]$ and $y \in T[w]$ do
      determine the path $\pi_{xy}$ that maximizes
      $S_{u2} = \omega_{xy} + R_{v,x} + R_{w,y}$
    end for
    if $S_{u2} > S_{u1}$ then
      $F_u = (x, y)$
    else
      $F_u = \emptyset$
    end if
    $S_u = \max(S_{u1}, S_{u2})$
  else
    for all pairs $v', v'' \in \mathrm{chd}(u)$ do
      determine the path $\pi_{xy}$ that maximizes $\tilde{Q}_{v',v''}$ and set $F_{v'v''} = (x, y)$ and $\omega_{v',v''} = \tilde{Q}_{v',v''}$
    end for
    for all pairs $v, v^* \in \mathrm{chd}(u)$ do
      $\omega_{v,v^*} = S_v$
    end for
    use computed edge weights for the following MWM problems
    $S_u = \mathtt{MWM}(\Gamma(\mathrm{chd}(u)))$
    for $i = 1$ to $|\mathrm{chd}(u)|$ do
      $k \leftarrow$i-th child from $u$
      compute $\delta = \mathtt{MWM}(\Gamma(chd(u) \setminus k))$
      for all leaves $x \in T[k]$ do
        $Q_{ux} = \delta$
      end for
    end for
    tabulate solution of all MWM problems
  end if
end for
```

The following algorithm summarizes backtracing. It starts at the root of the tree, but consider any vertex u:

```
if $|\mathrm{chd}(u)| = 0$ then
  return
end if
if $|\mathrm{chd}(u)| = 2$ then
  $\{v, w\} \leftarrow chd(u)$
  if $F_u = \emptyset$ then
    call backtracing for $T[v]$ (using the solution of the MWM for $S_v$ if $|\mathrm{chd}(v)| > 2$)
    repeat for $T[w]$
  else
    add $F_u = (x, y) = \pi_{xy}$ to solution set
    $k = v$ {path from $v$ to $x$}
    while $k \neq x$ do
      if $|\mathrm{chd}(k)| = 2$ then
        call backtracing for $T[\bar{k}_x]$
      else
        call backtracing for $T[k] \setminus T[k_x]$ (using the solution of the MWM for $Q_{kx}$)
      end if
      $k = k_x$
    end while
```

```
        repeat for $k = w$ {path from $w$ to $y$}
    end if
else
    $\{v_1, v_2, \ldots, v_n\} \leftarrow chd(u)$
    take the appropriate tabulated MWM $M$
    for all edges $(v_i, v_j)$ of $M$ do
        add $F_{v_i,v_j} = (x, y) = \pi_{xy}$ to solution set
        $k = v_i$ {path from $v_i$ to $x$}
        while $k \neq x$ do
            see case differentiation for the binary case
            $k = k_x$
        end while
        repeat for $k = v_j$ {path from $v_j$ to $y$}
    end for
    for all edges $(v_i, v_i^*)$ of $M$ do
        call backtracing for $T[v_i]$ (using the solution
        of the MWM for $S_{v_i}$ if $|\mathrm{chd}(v_i)| > 2$)
    end for
end if
```

A useful inequality

Consider an algorithm that operates on a rooted tree with n leaves requiring $\mathcal{O}((d_u)^\alpha)$ time for each interior vertex with d_u children. A naïve estimate immediately yields the upper bound $\mathcal{O}(n^{\alpha+1})$. Using the following lemma, however, we can obtain a better upper bound. Although Lemma 0.1 is probably known, we could not find a reference and hence include a proof for completeness.

Lemma 0.1 *Let T be a phylogenetic tree with n leaves, u an interior vertex, d_u the degree of u, and $\alpha > 1$. Then*

$$\sum_u (d_u)^\alpha \leq n^\alpha \tag{18}$$

Proof Let h denote the total number of interior vertices. Each leaf or interior vertex except the root is child of exactly one interior vertex. Thus $\sum_u d_u = n + (h - 1)$. For fixed h, we can employ the method of Lagrange multipliers to maximize the objective function $F(d_{u_1}, d_{u_2}, ..., d_{u_h}) = \sum_u (d_u)^\alpha$ subject to the constraint $\sum_u d_u = n + (h - 1) = c \leq 2n - 1$. The Lagrange function is then

$$\Lambda(d_{u_1}, d_{u_2}, \ldots, d_{u_h}, \lambda) = \sum_u (d_u)^\alpha + \lambda(\sum_u (d_u)^\alpha - c)\,. \tag{19}$$

Setting the partial derivatives of $\Lambda = 0$ yields the following system of equations:

$$\begin{aligned} \frac{\partial \Lambda}{\partial d_{u_i}} &= \alpha \cdot (d_{u_i})^{\alpha-1} + \lambda\ \forall u_i, i \in \{1, h\} \\ \frac{\partial \Lambda}{\partial \lambda} &= \sum_u (d_u)^\alpha - c \end{aligned} \tag{20}$$

, which is solved by $d_{u_1} = d_{u_2} = \ldots = d_{u_h} = d$ for all $i \in \{1, h\}$. The above sum is maximal when T is a full d-ary tree for some d. The constraint can thus be expressed as $h \cdot d = n + h - 1$ and $F = hd^\alpha$ which is maximized by making d as large as possible (i.e., n) and hence minimizing the number h of interior vertices (i.e., 1). Hence, $F(n) = n^\alpha$. ■

Authors' contributions

Both authors designed the study and developed the algorithms. C.A. implemented the TARGETING software. Both authors collaborated in writing the manuscript.

Acknowledgments

The authors thank Charles Nunn for useful discussions and the motivation for phylogenetic targeting. This work was supported in part by the Sächsische Aufbaubank, proj. BBZ-007, as part of the research program THERANOSTIK, and the bioinformatics initiative of the Deutsche Forschungsgemeinschaft.

References

1. Felsenstein J: **Phylogenies and the comparative method**. *Amer. Nat.* 1985, **125**:1–15.
2. Maddison WP: **Testing Character Correlation using Pairwise Comparisons on a Phylogeny**. *J. Theor. Biol.* 2000, **202**:195–204.
3. Ackerly DD: **Taxon sampling, correlated evolution, and independent contrasts**. *Evolution* 2000, **54**:1480–1492.
4. Arnold C, Nunn C: **Phylogenetic Targeting of Research Effort in Evolutionary Biology**. *American Naturalist* 2010. [In review].
5. Arnold C, Nunn CL: **Phylogenetic Targeting Website** 2010. [http://phylotargeting.fas.harvard.edu].
6. Bininda-Emonds OR, Cardillo M, Jones KE, MacPhee RD, Beck RM, Grenyer R, Price SA, Vos RA, Gittleman JL, Purvis A: **The delayed rise of present-day mammals**. *Nature* 2007, **446**:507–512.
7. Burleigh JG, Hilu KW, Soltis DE: **Inferring phylogenies with incomplete data sets: a 5-gene, 567-taxon analysis of angiosperms**. *BMC Evol. Biol.* 2009, **9**:61.

8. Arnold C, Matthews LJ, Nunn CL: **The 10kTrees Website: A New Online Resource for Primate Phylogeny**. *Evol. Anthropology* 2010. in press.
9. Sanderson MJ, Driskell AC: **The challenge of constructing large phylogenetic trees**. *Trends Plant Sci.* 2003, **8**:374–379.
10. Gabow H: **Implementation of Algorithms for Maximum Matching on Nonbipartite Graphs**. *PhD thesis*, Stanford University 1973.
11. Galil Z, Micali S, Harold G: **An $O(EV \log V)$ algorithm for finding a maximal weighted matching in general graphs**. *SIAM J. Computing* 1986, **15**:120–130.
12. Gabow HN, Tarjan RE: **Faster scaling algorithms for general graph matching problems**. *J. ACM* 1991, **38**:815–853.
13. Purvis A, Bromham L: **Estimating the transition/transversion ratio from independent pairwise comparisons with an assumed phylogeny**. *J. Mol. Evol.* 1997, **44**:112–119.
14. Mückstein U, Hofacker IL, Stadler PF: **Stochastic Pairwise Alignments**. *Bioinformatics* 2002, **S153–S160**:18.
15. McCaskill JS: **The equilibrium partition function and base pair binding probabilities for RNA secondary structures**. *Biopolymers* 1990, **29**:1105–1119.
16. Steffen P, Giegerich R: **Versatile and declarative dynamic programming using pair algebras**. *BMC Bioinformatics* 2005, **6**:224.
17. Rothenberg E: **Solver for the Maximum Weight Matching Problem** 1999. [http://elib.zib.de/pub/Packages/mathprog/matching/weighted/].
18. Gabow HN, Kaplan H, Tarjan RE: **Unique Maximum Matching Algorithms**. *J. Algorithms* 2001, **40**:159–183.
19. Nunn CL, Baton RA: **Comparative Methods for Studying Primate Adaptation and Allometry**. *Evol. Anthropology* 2001, **10**:81–98.
20. Goodwin NB, Dulvy NK, Reynolds JD: **Life-history correlates of the evolution of live bearing in fishes**. *Phil. Trans. R. Soc. B: Biol. Sci.* 2002, **357**:259–267.
21. Vinyard CJ, Wall CE, Williams SH, Hylander WL: **Comparative functional analysis of skull morphology of tree-gouging primates**. *Am. J. Phys. Anthropology* 2003, **120**:153–170.
22. Poff NLR, Olden JD, Vieira NKM, Finn DS, Simmons MP, Kondratieff BC: **Functional trait niches of North American lotic insects: traits-based ecological applications in light of phylogenetic relationships**. *J. North Am. Benthological Soc.* 2006, **25**:730–755.