Fast dating using least-squares criteria and algorithms

Citation for published version:

Digital Object Identifier (DOI):
10.1093/sysbio/syv068

Link:
Link to publication record in Edinburgh Research Explorer

Document Version:
Peer reviewed version

Published In:
Systematic biology

General rights
Copyright for the publications made accessible via the Edinburgh Research Explorer is retained by the author(s) and / or other copyright owners and it is a condition of accessing these publications that users recognise and abide by the legal requirements associated with these rights.

Take down policy
The University of Edinburgh has made every reasonable effort to ensure that Edinburgh Research Explorer content complies with UK legislation. If you believe that the public display of this file breaches copyright please contact openaccess@ed.ac.uk providing details, and we will remove access to the work immediately and investigate your claim.
ONLINE APPENDIX

Fast dating using least-squares criteria and algorithms

Thu Hien TO¹, Matthieu JUNG¹,³, Samantha LYCETT² and Olivier GASCUEL¹,*

¹Institut de Biologie Computationnelle, LIRMM, UMR 5506 CNRS – Université Montpellier 2, Montpellier, France
²University of Edinburgh, Institute of Evolutionary Biology, Ashworth Laboratories, Edinburgh, United Kingdom
³Current address: IGBMC (Institut de Génétique et de Biologie Moléculaire et Cellulaire), INSERM, U596, CNRS, UMR7104, Université de Strasbourg, Illkirch, France

*Corresponding author: gascuel@lirmm.fr

Unicity of the minimum in both constrained and unconstrained settings p. 2
LD algorithm p. 3
QPD algorithm p. 5
Lagrange linear-time algorithm to find the stationary point of Equation (8) p. 7
Convergence of the QPD algorithm p. 11
Root estimation p. 13
Supp. Figure S1. Summary results with the 750/3×25 simulated data set p. 15
Supp. Figure S2. Summary results with the 750/11×10 simulated data set p. 16
Supp. Figure S3. Summary results with the 995/3×25 simulated data set p. 17
Supp. Figure S4. Summary results with the 995/11×10 simulated data set p. 18
Supp. Figure S5. Error in rate estimation when using the true phylogeny p. 19
Supp. Table S1: Computing times with simulated data p. 20
Supp. Table S2: Simulation results with a strict molecular clock p. 21
Supp. Table S3: Simulation results with a lognormal, relaxed molecular clock p. 22
Supp. Table S4: Topological errors with simulated data sets p. 23
Supp. Table S5. Origins and dates of influenza A H1N1pdm09 strains p. 24
\( \phi \) (Eq. 2) has a unique minimum in both constrained and unconstrained settings

\( \Psi \) in Equation (2b) is a quadratic form (O’Meara 2000) and there exists a matrix \( Q \), a vector \( v \) and a positive constant \( k \) such that \( \Psi(x) = \frac{1}{2} x^T Q x + v^T x + k \), where \( x = (\omega, \beta_1, ..., \beta_{n-1}) \). The quadratic term

\[
\frac{1}{2} x^T Q x = \sum_{i=2}^{n-1} w_i (\beta_i - \beta_{a(i)})^2 + \sum_{i=n}^{2n-1} w_i (\omega t_i - \beta_{a(i)})^2
\]

is always positive for any \( x \). So \( Q \) is positive definite, and thus \( \Psi \) is strictly convex and has a unique global minimum. It is easily seen that the ranges of \( \phi \) (in Eq. 2) and \( \Psi \) are the same, and each vector \((\omega, t_1, ..., t_{n-1})\) corresponds to one and only one vector \((\omega, \beta_1, ..., \beta_{n-1})\). So, \( \phi \) also has a unique global minimum.

The function \( \Psi \) remains strictly convex on the restricted domain of constraints. Indeed, by definition of strict convexity: \((\Psi(X) + \Psi(X'))/2 > \Psi((X + X')/2)\) for every two vectors \( X = (\omega, \beta_1, ..., \beta_{n-1}) \) and \( X' = (\omega', \beta'_1, ..., \beta'_{n-1}) \); moreover, if \( X \) and \( X' \) satisfy the constraints, then \((X + X')/2\) also satisfies the constraints. So \( \Psi \) is strictly convex and has a unique minimum on the domain of constraints. Therefore, \( \phi \) also has a unique global minimum on the domain of constraints.
LD Algorithm

Input: a rooted tree with: branch lengths $b_2, ..., b_{2n-1}$; the dates of the leaves $t_n, ..., t_{2n-1}$; and $\omega_{\text{min}}$

Output: $\hat{\omega} \geq \omega_{\text{min}}$ and $\hat{t}_1, ..., \hat{t}_{n-1}$ that minimize Eq. (2)

Calculate the system of Equations (5)

pos $\leftarrow$ post-order of the internal nodes

For $i$ from pos[1] to pos[n - 1] and $i \neq 1$

\[ t_i = x_i a(i) + y_i + \frac{v_i}{\hat{\omega}} \quad (6.i) \]

Else if $s_1(i)$ (resp. $s_2(i)$) is not a leaf, it has been already processed and we replace Eq. (6. $s_1(i)$) (resp. Eq. (6. $s_2(i)$)) into Eq. (5) to obtain Eq. (6.i)

pre $\leftarrow$ pre-order of the internal nodes

For $i$ from pre[1] to pre[n - 1]

If $i = 1$ then replace Eq. (6. $s_1(1)$) and Eq. (6. $s_2(1)$) into Eq. (5.1) to obtain $t_1 = u_1 + \frac{v_1}{\hat{\omega}}$ (Eq. (7.1))

Else replace $t_i = u_i + \frac{v_i}{\hat{\omega}} (7.i)$ into Eq. (6. $s_1(i)$) and into Eq. (6. $s_2(i)$), to obtain Eq. (7. $s_1(i)$) and Eq. (7. $s_2(i)$)

Replace $n - 1$ equations Eq. (7.i) into Eq. (2) to compute $\hat{\omega}$ that minimizes Eq. (2).

If $\hat{\omega} < \omega_{\text{min}}$ then $\hat{\omega} = \omega_{\text{min}}$.

Replace $\hat{\omega}$ into Eq. (7.i) to compute $\hat{t}_1, ..., \hat{t}_{n-1}$

This algorithm takes as input a binary rooted tree with branch lengths and the sampling date associated to each tip. The outputs are the substitution rate and the dates of all internal nodes (including the root) that minimize Equation (2). All calculations are performed in linear time.
starts by calculating the system of Equations (5), and then resolves this system with recursive replacements using post-order and pre-order traversals (Tarjan 1983) of the input, rooted tree $R$.

By using a post-order (from the leaves up to the root) in the first loop, in each iteration $i$, if $s$ is a descendant of $i$, then either $s$ is a leaf or the iteration $s$ has already been achieved. If $s$ is a leaf, then $t_s$ is known. Otherwise, $x_s$, $y_s$ and $z_s$ of Equation (6.s) have been computed in iteration $s$. In both cases, by replacing the value of $t_s$ or the Equation (6.s) into Equation (5.i), we deduce $x_i, y_i$ and $z_i$ of Equation (6.i).

Next, by using a pre-order (from the root down to the leaves) in the second loop, in each iteration $i$, either $i$ is the root or the iteration of $a(i)$, the ancestor of $i$, has already been achieved. If $i = 1$ (the root), by replacing Equations (6.s (1)) and (6.s (2)) into Equation (5.1), we infer two constants $u_1$ and $v_1$ such that $t_1 = u_1 + v_1/\omega$, as required by Equation (7.1). Otherwise, $u_{a(i)}$ and $v_{a(i)}$ have been computed in $a(i)$ iteration, so by replacing Equation (7.a(i)) into Equation (6.1), we deduce $u_i$ and $v_i$ of Equation (7.1).

Afterwards, the $(n - 1)$ Equations (7) for every $i$ are replaced into Equation (2) to obtain $\tilde{\omega}$. If $\tilde{\omega} < \omega_{min}$, then $\omega_{min}$ is the optimal rate under $(\tilde{\omega} \geq \omega_{min} > 0)$ constraint. This property follows from the strict convexity of the equation obtained from Equation (2) after replacement of the $(n - 1)$ Equations (7). We thus assign $\tilde{\omega} = \omega_{min}$ in this case.

Lastly, $\tilde{\omega}$ is used to obtain all time estimates $\tilde{t}_i$ using Equations (7).
QPD ALGORITHM

QPD Algorithm

Input: a rooted tree with: branch lengths $b_2, \ldots, b_{2n-1}$; the dates of the leaves $t_{n}, \ldots, t_{2n-1}$; and $\omega_{\min}$

Output: $\hat{\omega} \geq \omega_{\min}$ and $\hat{t}_j, \ldots, \hat{t}_{n-1}$ that minimizes $\phi$ (Eq. 2) and satisfy the temporal constraints

$x = (\omega, t_1, \ldots, t_{n-1}) \leftarrow$ the solution of LD; $C = \emptyset$;

$pos \leftarrow$ a post-order of internal nodes of the input tree;

For $i = pos[1]$ to $pos[n-1]$

| If $t_{s_1(i)} < t_{s_2(i)}$ then | ; bottom-up initialization of the active set $C$
| If $t_i > t_{s_1(i)}$ then $t_i \leftarrow t_{s_1(i)}$; add $s_1(i)$ into $C$; | ; $x$ is made feasible
| Else If $t_i > t_{s_2(i)}$ then $t_i \leftarrow t_{s_2(i)}$; add $s_2(i)$ into $C$. |

For $k = 0$ to MaxNumberIterations

Use Lagrange Algorithm to compute the stationary point $(x^*, \lambda^*)$ of Eq. (8) with respects to $C$

If ($x^*$ is feasible) then

If $(\lambda^*_h \geq 0$ for every $h \in C)$ then return solution ; KKT conditions, $x^*$ is feasible and optimal

Else let $\lambda^*_h$ be the most negative component of $\lambda^*$; remove $h$ from $C$ ; $h$ is useless

Else

Let $V = \{i | t^*_i < t^*_{a(i)} \}$ ; $V$ is the set of violated constraints in $x^*$

If there exists $i \in V$ s.t. $t_i = t_{a(i)}$ then add $i$ into $C$ ; $x$ remains unchanged

Else Let $\Delta_i = (t^*_i - t^*_{a(i)})/(t_i - t_{a(i)})$ for all $i$ in $V$

Let $p$ be the index such that $\Delta_p$ is minimum over all $\Delta_i$ terms, and $\alpha = 1/(1 - \Delta_p)$

Add $p$ into $C$ ; $p$ is the “first violated” constraint

Set $x \leftarrow x + \alpha(x^* - x)$ ; $x$ is feasible and as close as possible to $x^*$

Return $x$. ; $x$ is feasible and nearly optimal
This (active set, Nocedal and Wright 2006) algorithm takes as input a binary rooted tree with branch lengths and the sampling dates associated to the tips. The outputs are the substitution rate and the dates of all internal nodes (including the root) which minimize Equation (2), such that
\[ \mathcal{A}_i = \omega(\mathcal{A}_i), \] for every \( i \), and \( \omega \geq \omega_{\text{min}} \). QPD runs in \( O(f \times n) \), where \( f \) is the number of iterations needed to reach the optimal feasible solution \( (x^*, \lambda^*) \), when KKT conditions are fulfilled, and \( n \) is the number of taxa. We observed \( f \ll n \) in all of our experiments (e.g. 4-5 with 110-taxon simulated trees, and 69 with the 891-taxon influenza data set), and the maximum number of iterations (set to 1,000 in our implementation) was never reached, by far.

The starting point \( x \) is obtained by LD, and the active set \( C \) is computed from the constraints being violated in \( x \), using a post-order tree traversal. In each iteration, we compute the optimal solution of Equation (2) such that \( \mathcal{A}_i = \omega(\mathcal{A}_i) \) for all \( i \in C \). This is equivalent to searching for the stationary point \( (x^*, \lambda^*) \) of the Lagrange Equation (8). To achieve this task we use a slightly modified version of LD where the branches corresponding to violated constraints are collapsed; see Lagrange Algorithm below. If the KKT conditions are satisfied \( (x^* \) is feasible and \( \lambda^* \) does not contain any negative elements), then the algorithm stops and returns \( x^* \). Otherwise the active set \( C \) is updated as follows. If \( x^* \) is feasible and \( \lambda^* \) contains negative elements, then the constraint corresponding to the most negative value is relaxed. Otherwise, we search for a violated constraint to add into \( C \) and a new feasible point \( x \) being closer to the optimum. To this purpose, \( x \) is moved as close as possible to \( x^* \) following the direction \( d = x^* - x \). In other words, we have to find the largest number \( \alpha \) in \([0,1]\) such that \( x + \alpha d \) is feasible. Note that if a constraint \( i \) is valid in \( x^* \), then it is also valid in \( x + \alpha d \) for every \( \alpha \) in \([0,1]\). Thus, we only need to consider the constraints \( i \) that are violated in \( x^* \). Let \( V = \{ i \in 2, \ldots, 2n - 1 | t^*_i < t^a(\mathcal{A}_i) \} \), then \( t_i + \alpha(t^*_i - t_i) \geq t(a(\mathcal{A}_i) + \alpha(t^a(\mathcal{A}_i) - t(a(\mathcal{A}_i))) \) must hold for every \( i \in V \). If there exists \( i \in V \) such that \( t_i = t(a(\mathcal{A}_i)) \), then \( \alpha = 0 \) and \( i \) is the new constraint added into \( C \). Otherwise, \( t_i > t(a(\mathcal{A}_i)) \) for all \( i \in V \) (because \( x \) is feasibl), thus let \( \Delta_i = (t^*_i - t(a(\mathcal{A}_i)))/(t_i - t(a(\mathcal{A}_i))) \), the feasibility condition becomes \( \alpha \leq 1/(1 - \Delta_i) \) for every \( i \in V \). Let \( p \) be the index such that \( \Delta_p \) is minimum over all \( \Delta_i \) terms with \( i \in V \). Then, \( \alpha = 1/(1 - \Delta_p) \), \( x \) becomes \( x + \alpha d \), and \( p \) is added into the active set. We repeat these calculations until the KKT conditions are satisfied (then \( x^* \) is returned) or the given maximum number of iterations is reached (then \( x \) is returned). We prove below that this algorithm converges to the optimal solution when the number of iterations is large enough.
LAGRANGE LINEAR-TIME ALGORITHM TO FIND THE STATIONARY POINT OF EQUATION (8)

\[ \text{Lagrange algorithm} \]

Input: The Lagrange function (8) and the active set \( \mathcal{C} \)

Output: The stationary point \( (x^*, \lambda^*) \) of (8)

Calculate the system of equations (10,11,12)

**For** each internal node \( i \) such that \( i \) is not in \( \mathcal{C} \), and either \( s_1(i) \) or \( s_4(i) \) is in \( \mathcal{C} \)

Calculate the set \( \mathcal{C}_i \)

*If* \( \mathcal{C}_i \) contains only internal nodes, then collapse all branches in \( \mathcal{C}_i \)

*Else* assign the dates of all nodes in \( \mathcal{C}_i \) by the date of the leaf contained in \( \mathcal{C}_i \)

Use the generalized version of LD algorithm to compute \( x^* \) on the collapsed tree

**For** each \( \mathcal{C}_i \)

**For** each \( j \) in \( \mathcal{C}_i \) with respect to the order defined by rules 1,2,3,4

Calculate \( \lambda_j^* \) by using \( x^* \) and the already computed elements of \( \lambda^* \) in Equation (10,j)

Return \( x^* \) and \( \lambda^* \)

The stationary point \( (x^*, \lambda^*) \) of the Lagrange Equation (8) is computed by solving the system of equations in which the partial derivatives of \( \Gamma \) (Eq. (8)) are null (the weight terms \( w_i \) are ignored in the formulae for simplicity):

\[
\frac{\partial \Gamma(x, \lambda)}{\partial t_1} = 2\omega \left( b_{s_1(i)} - \omega t_{s_1(i)} + \omega t_1 \right) + 2\omega \left( b_{s_2(i)} - \omega t_{s_2(i)} + \omega t_1 \right) + \lambda_{s_1(i)} + \lambda_{s_2(i)} = 0, \tag{10.1}
\]
This system of equations is solved under the conditions that:

\[ t_i = t_{a(i)}, \forall i \in C, \text{ and } \lambda_i = 0, \forall i \notin C. \]  \hspace{1cm}(12)

The “Lagrange” algorithm (summarized above) solves Equations (10, 11, 12) in linear time. Through the first loop, the active constraints of \( C \) are imposed on the tree. Then we use a generalized version of LD on the collapsed tree to compute \( x^* \). Lastly, we compute \( \lambda^* \) by using \( x^* \) and recursive replacements following the specific order of \( C_i \) in Equations (10). Let us detail these calculations:

**Computing \( x^* \):** the key remark is that if \( i \) is in \( C \), then the equation \( \frac{\partial \Gamma}{\partial t_i} \) contains \(-\lambda_i\), and the equation \( \frac{\partial \Gamma}{\partial t_{a(i)}} \) contains \(+\lambda_i\). So \( \lambda_i \) is eliminated by taking the sum of these two equations, and we can eliminate all Lagrange multipliers in all Equations (10.i) by taking the sum of appropriate sets of equations. Each set corresponds to a maximal set of nodes whose dates are equal due to active constraints. Let call \( i \) the “top” of such a set if \( i \) is not in \( C \) but either \( s_1(i) \) or \( s_2(i) \) are in \( C \). We then define for each top \( i \) a set of active constraints \( C_i \) by the following rules:

1. every child of \( i \) that is in \( C \) is in \( C_i \);
2. if \( j \in C \) and \( a(j) \in C_i \), then \( j \in C_i \). It is easy to see that these sets are pairwise disjoint. Moreover, each \( C_i \) contains at most one leaf all along QPD iterations, as we shall prove recursively. In the initial active set \( C \) each internal node has at most one child in \( C \), so each \( C_i \) contains at most one leaf. Moreover, if \( C_i \) contains a leaf \( k \), then \( t_k \leq t_{k'} \) for every leaf \( k' \neq k \) below \( i \). Suppose that \( k' \) has its parent in \( C_i \); so in the next iteration, if \( k \)
is still in \( C_i \) then \( k' \) is not added into \( C_i \), because the constraint \( k' \) is not violated \( (t_{k'} \geq t_{a(k')} = t_k) \). If \( k \) is removed from \( C_i \), then whether \( k' \) is added into \( C_i \) or not, \( C_i \) contains at most one leaf. If the parent of \( k' \) is not in \( C_i \), then if \( k' \) is added into \( C_i \) in the next iteration, it must be contained in a \( C_j \) where \( j \neq i \). Therefore, each \( C_i \) contains at most one leaf in every iteration of QPD. If \( C_i \) contains a leaf, then the dates of all nodes in \( C_i \) are equal to the date of this leaf, which is known. So it is not necessary to compute the dates of these nodes in this case.

Consider now the case where \( C_i \) only contains internal nodes. The sum of the equations \( \partial \Gamma / \partial t_j \) for all \( j \in C_i \) is

\[
-2\omega(b + \omega t_i + \omega t_{a(i)}) + 2\omega \sum_{s \in S(i)} (b_s - \omega t_s + \omega t_i) = 0,
\]

where \( S(i) \) consists of the children of the lowest nodes of \( C_i \). This equation does not contain any Lagrange multiplier, and has the same form as Equations (5) but on a tree where all branches in \( C_i \) are collapsed. Therefore, by using on the new tree a generalized version of LD able to process unresolved trees, we can compute in linear time, for every internal node \( i \) such that \( i \notin C \), the constants \( u_i \) and \( v_i \) such that \( t_i = u_i + v_i / \omega \). The later equations and the value of all known \( t_i \) are then replaced into (11) to calculate \( \omega^* \), and then \( t_i^* \).

**Computing \( \lambda^* \):** we can also compute \( \lambda^* \) in linear time by replacing \( x^* \) into Equations (10.i) following a specific order which ensures that each time there is one and only one unknown \( \lambda_i \) in the equation at hand, so that we can easily deduce its value. This order is defined on the \( C_i \) sets, distinguishing two cases:

1. If \( C_i \) contains internal nodes only, then one simply uses a bottom-up order: for every \( j_1, j_2 \in C_i \), if \( j_1 \) is below \( j_2 \), then \( j_1 < j_2 \) (rule 1).
2. If \( C_i \) contains a leaf \( k \), then as we proved above, \( k \) is the only leaf in \( C_i \). Denote by \( C_i^1 \) the set of nodes in \( C_i \) that are not above or equal to the leaf \( k \), and \( C_i^2 \) the set of nodes in \( C_i \) that are above or equal to \( k \). The order of the elements in \( C_i \) respects the following rules: the nodes in \( C_i^1 \) follow a bottom-up order (rule 2); the nodes in \( C_i^2 \) follow a top-down order (rule 3); the nodes in \( C_i^1 \) are smaller than the nodes in \( C_i^2 \) (rule 4).
Then, $\lambda_j$ is computed following the order defined above: for every $j \in C_i$, either $j$ is the smallest node in $C_i$ (so the first one in $C_i$ to be computed) or all Lagrange multipliers of nodes smaller than $j$ in $C_i$ have already been calculated. If $C_i$ only contains internal nodes (case (1)), then $\forall j \in C_i, j$ always has two direct descendant $s_1(j), s_2(j)$. Moreover, $\lambda_{s_1(j)}$ (resp. $\lambda_{s_2(j)}$) is either 0 (not in the active set) or has already been calculated (because $s_1(j), s_2(j)$ are smaller than $j$ by rule 1). In other words, $\lambda_j$ is the only unknown variable in Equation (10.j) and is easily computed. Let us now deal with case (2) and let $k$ be the only leaf in $C_i$. If $j$ is not above or equal to $k$, then $\lambda_{s_1(j)}$ (resp. $\lambda_{s_2(j)}$) is either 0 or has already been calculated (because $s_1(j), s_2(j)$ are smaller than $j$ by rule 2); therefore, $\lambda_j$ is determined by Equation (10.j). If $j$ is above or equal to $k$, then let $j'$ be the sibling node of $j$. So $j'$ is not above or equal to $k$. Hence, $\lambda_{s_1(j)}$ (resp. $\lambda_{s_2(j)}$) is either 0 or has already been calculated (they are smaller than $j$ by the rule 3 (resp. rule 4)). Therefore, $\lambda_j$ is the only unknown variable in Equation (10.j) and thus is easily computed.
CONVERGENCE OF THE QPD ALGORITHM

Let us consider the problem of finding the minimal solution of Ψ (Eq. 2b), subject to the constraints \( \omega \geq \omega_{\text{min}}, \beta_i \geq \beta_{a(i)} \) for \( i = 2, ..., n - 1 \), and \( \omega t_i \geq \beta_{a(i)} \) for \( i = n, ..., 2n - 1 \). Ψ is convex quadratic, so the active set method is ensured to converge to its unique global minimum on the constraint domain (Nocedal and Wright 2006). By choosing the starting feasible point and the initial active set of Ψ similarly to the one of φ in QPD algorithm (i.e., the minimal solution of Ψ without constraints, and then deriving \( C \) from the set of violated constraints, see above), we prove here the equivalence of the active sets and feasible points of Ψ and φ in every iteration of both algorithms. This implies the convergence of the QPD algorithm, which is the active set method applied to φ (Eq. 2).

Let \( x = (\omega_x, t_1, ..., t_{n-1}) \) be the variable vector of φ (Eq. 2), and \( y = (\omega_y, \beta_1, ..., \beta_{n-1}) \) be the variable vector of Ψ (Eq. 2b). We say that \( x \) and \( y \) are equivalent if and only if \( \omega_x = \omega_y \) and \( \beta_i = \omega_y t_i \) for every \( i \) from 1 to \( n - 1 \).

First, it is easy to see that \( y = (\omega, \beta_1, ..., \beta_{n-1}) \) is the minimal solution of Ψ (Eq. 2b), without constraints, if and only if \( x = (\omega, \beta_1/\omega, ..., \beta_{n-1}/\omega) \) is the minimal solution of φ (Eq. 2), without constraints. In other words, the minimal solution of φ and Ψ, without constraints, are equivalent, and is easy to define for Ψ a starting point equivalent to φ’s and an initial active set exactly the same as φ’s (see QPD algorithm above).

Second, given an active set \( C \), it is easy to see that \( y = (\omega, \beta_1, ..., \beta_{n-1}) \) is the minimal solution of Ψ (Eq. 2b) subjects to \( C \) (i.e., \( \forall i \in W, \beta_i = \beta_{a(i)} \) if \( i < n \), and \( \omega t_i = \beta_{a(i)} \) if \( i \geq n \)) if and only if \( x = (\omega, \beta_1/\omega, ..., \beta_{n-1}/\omega) \) is the minimal solution of φ (Eq. 2) subjects to \( C \) (i.e., \( \forall i \in C, t_i = t_{a(i)} \)). Hence, if \( (x^*, \lambda_x^*) \) (resp. \( (y^*, \lambda_y^*) \)) is the stationary point of the Lagrange function of φ (Eq. 8) (resp. Ψ, Eq. (8b)) with respects to \( C \), then \( x^* \) and \( y^* \) are equivalent. We will prove that \( \lambda_x^* = \omega^* \lambda_y^* \). The Lagrange function of Ψ with respects to \( C \) is:

\[
\mathcal{L}(y, \lambda) = \Psi(y) - \sum_{i \in C, i < n} \lambda_i (\beta_i - \beta_{a(i)}) - \sum_{i \in C, i \geq n} \lambda_i (\omega t_i - \beta_{a(i)}).
\] (8b)
By taking the partial derivatives of $\mathcal{L}$ in Equation (8b), we have a system of equations similar to Equations (10, 11, 12) but without $\omega$ factor before each term as in Equations (10). By replacing $x^*$ and $y^*$ (which are equivalent) into these equations, we deduce that $\lambda_x^* = \omega^* \lambda_y^*$.

Finally, we prove that if the two starting active sets $\mathcal{C}_x$, $\mathcal{C}_y$ are the same, then they remain the same after each update. Suppose that they are the same at the $k^{th}$ iteration, then by using the claim above, we have that $x^*$, $y^*$ are equivalent and $\lambda_x^* = \omega^* \lambda_y^*$. So the KKT condition on $(y^*, \lambda_y^*)$ is satisfied if and only if it is satisfied on $(x^*, \lambda_x^*)$. The indices of the most negative components of $\lambda_x^*$ and $\lambda_y^*$ are the same, so the constraints to relax are the same. It remains to prove that the two new active constraints added into $\mathcal{C}_x, \mathcal{C}_y$ are also the same. Since $x^*$ and $y^*$ are equivalent, then the set of constraints that are violated in $x^*$ are the same as in $y^*$. In other words, let $V_x^* = \{i = 2, ..., 2n - 1 | t_i^* < t_a(i)\}$, and $V_y^* = \{i = 2, ..., n - 1 | \beta_i^* < \beta_a(i)\} \cup \{i = n, ..., 2n - 1 | \omega^* t_i < \beta_a(i)\}$, then $V_x^* = V_y^*$. If there exists $i \in V_x^*$ such that $t_i^* = t_a(i)$, then $\beta_i^* = \beta_a(i)$ if $i < n$ or $\omega^* t_i = \beta_a(i)$ if $i \geq n$, and $i$ is the new constraint to be added into $\mathcal{C}_x$ and $\mathcal{C}_y$. Otherwise, let $\Delta_i = (t_i^* - t_a(i)) / (t_i - t_a(i))$ and $\nabla_{i*} = (\omega^* / \omega) \Delta_i$ for all $i \in V_x^*$. As explained in in QPD description (above), the constraint $p$ to be added into $\mathcal{C}_x$ corresponds to the $\Delta_p$ index that is minimum over all $\Delta_i$ with $i \in V_x^*$. Similarly, we can prove that the constraint $q$ to be added into $\mathcal{C}_y$ corresponds to the $\nabla_q$ index that is minimum of all $\nabla_i$ with $i \in V_y^*$. Because $\omega^*$ and $\omega$ are positive, and $V_x^* = V_y^*$, then we always have $p = q$. In other words, we add the same constraint into the active sets $\mathcal{C}_x, \mathcal{C}_y$. Hence, the two active sets are the same at the $(k + 1)^{th}$ iteration, and they are the same until the end, when the KKT condition is satisfied for both. In conclusion, the QPD algorithm is converged.
ROOT ESTIMATION

Without constraints

Optimizing Equation (9) without constraints can also be solved in linear time by slightly modifying the algorithm LD. Indeed, by calculating the first order derivatives of $\varphi$ with respect to each variable, we have a system of equations:

$$
\begin{align*}
\mu &= \frac{\omega}{2b}(t_{r1} - t_{r2}) + \frac{1}{2}, \\
t_r &= \frac{1}{2}(t_{r1} + t_{r2} - \frac{1}{\omega} b), \\
t_i &= \frac{1}{3}\left[\left( t_{s1(i)} - \frac{1}{\omega} b_{s1(i)} \right) + \left( t_{s2(i)} - \frac{1}{\omega} b_{s2(i)} \right) + \left( t_r + \frac{1}{\omega} \mu b \right) \right], \quad i = r_1 \\
t_i &= \frac{1}{3}\left[\left( t_{s1(i)} - \frac{1}{\omega} b_{s1(i)} \right) + \left( t_{s2(i)} - \frac{1}{\omega} b_{s2(i)} \right) + \left( t_r + \frac{1}{\omega} (1 - \mu) b \right) \right], \quad i = r_2 \\
t_i &= \frac{1}{3}\left[\left( t_{s1(i)} - \frac{1}{\omega} b_{s1(i)} \right) + \left( t_{s2(i)} - \frac{1}{\omega} b_{s2(i)} \right) + \left( t_a(i) + \frac{1}{\omega} b_i \right) \right], \quad i \neq r_1, r_2 \\
\omega &= \frac{\mu b(t_{r1} - t_r) + (1 - \mu) b(t_{r2} - t_r) + \sum_{i \neq r_1, r_2} b_i(t_i - t_a(i))}{(t_{r1} - t_r)^2 + (t_{r2} - t_r)^2 + \sum_{i \neq r_1, r_2} (t_i - t_a(i))^2},
\end{align*}
$$

which simplifies into:

$$
\begin{align*}
\mu &= \frac{\omega}{2b}(t_{r1} - t_{r2}) + \frac{1}{2}, \\
t_r &= \frac{1}{2}(t_{r1} + t_{r2} - \frac{1}{\omega} b), \\
t_i &= \frac{1}{2}\left[\left( t_{s1(i)} - \frac{1}{\omega} b_{s1(i)} \right) + \left( t_{s2(i)} - \frac{1}{\omega} b_{s2(i)} \right) \right], \quad i = r_1, r_2 \\
t_i &= \frac{1}{3}\left[\left( t_{s1(i)} - \frac{1}{\omega} b_{s1(i)} \right) + \left( t_{s2(i)} - \frac{1}{\omega} b_{s2(i)} \right) + \left( t_a(i) + \frac{1}{\omega} b_i \right) \right], \quad i \neq r_1, r_2 \\
\omega &= \frac{\mu b(t_{r1} - t_r) + (1 - \mu) b(t_{r2} - t_r) + \sum_{i \neq r_1, r_2} b_i(t_i - t_a(i))}{(t_{r1} - t_r)^2 + (t_{r2} - t_r)^2 + \sum_{i \neq r_1, r_2} (t_i - t_a(i))^2}.
\end{align*}
$$

Just as with Equation (2), the above system has a unique solution and can be resolved in linear time using the same approach as LD (i.e., simplify progressively by recursive replacements upon post-order and then pre-order, see above). Hence, the function $\varphi$ has only one critical point. By considering its second order derivatives, it is easy to see that this point corresponds to the minimum solution. To solve this problem when $\mu$ is bounded between 0 and 1, we first calculate
the minimum solution in the general case. If the minimum value of $\mu$ is not in $[0, 1]$, then we compute the minimum solution of Equation (9) in the cases where $\mu = 0$ and $\mu = 1$, and then take the one which makes $\varphi$ smaller. By running this procedure on all branches, the time complexity is quadratic in the number of tips.

**With constraints**

We use again the active set method (see QPD algorithm) to optimize $\varphi$ in Equation (9), subject to the constraints $\omega \geq \omega_{\text{min}}$, $t_i \geq t_{a(i)}$, and $0 \leq \mu \leq 1$. In this case, $x = (\omega, t_1, ..., t_{n-1}, \mu)$. Let $\mathcal{Z}$ be the Lagrange function of Equation (9). To find the stationary point of $\mathcal{Z}$, we use an algorithm similar to the Lagrange algorithm (see above). The Lagrange multipliers are computed in the same way, using that $\lambda_\mu$ can be computed directly from equation $\frac{\partial \mathcal{Z}}{\partial \mu} = 0$.

To avoid exploring all branches, which could require heavy computing times with large data sets, we pre-estimate the root position without temporal constraints. Then, we use the method with constraints to search for the local minimum around that position. Let $o$ be the branch that contains the pre-estimated root position, and $E$ be a set that initially contains the branch $o$ only. Denote by $\varphi_i$ the optimum value of $\varphi$ in Equation (9), when the root is situated on the branch $i$. Let $r$ be the branch that contains the estimated root ($r$ is initialized with $o$), and $\varphi^* = \varphi_r$ the corresponding objective function value. Consider a branch $i$ in $E$; add into $E$ all neighbors $j$ of $i$ such that $\varphi_j < \varphi_i$, and then remove $i$ from $E$. If the smallest value of these neighboring branches is smaller than $\varphi^*$, then update $\varphi^*$ with this value and $r$ with the corresponding branch. Continue exploring the neighbors of all branches in $E$ and stop when $E$ is empty. It is easy to check that the explored branches form the largest subtree in which the value of $\varphi$ at $o$ is the greatest, and that this value decreases from $o$ to every terminal branch of that subtree. Moreover, $r$ corresponds to the terminal branch having the smallest value. In practice, starting from the pre-estimated root without constraints, only a few branches need to be explored to reach to the optimal one with constraints. For example, with our simulated data of 110 taxa, we usually have to consider 4-5 branches in average, and in most cases the pre-estimated root is on the same branch as that of the final root. However, both fast and complete options are available in LSD, as we observed a few rare cases where the fast option performed poorly.
Supplementary Figure S1. Summary results with the 750/3×25 simulated data set. See legend of Figure 2.
**Supplementary Figure S2. Summary results with the 750/11×10 simulated data set.** See legend of Figure 2.
Supplementary Figure S3. Summary results with the 995/3×25 simulated data set. See legend of Figure 2.
Supplementary Figure S4. Summary results with the 995/11×10 simulated data set. See legend of Figure 2.
Supplementary Figure S5. Error in rate estimation when using the true topology. The true, rooted topology is used by all methods. With distance methods (LD*, QPD*, LF*, RTT*) the branch lengths are estimated by PhyML. BEAST (BSMC*: strict clock, BRMC*: lognormal, relaxed clock) sample the branch lengths (and the dates and the substitution rate) from sequence data. We measure here the relative error in estimating the substitution rate. We see that the high estimation error of BEAST is not due to topological errors, and that the ranking of methods is basically preserved, compared to the (realistic case) where an approximate (FastME, PhyML or BEAST) topology is used to estimate the substitution rate (and other date parameters).
<table>
<thead>
<tr>
<th></th>
<th>750/11 \times 10</th>
<th>995/3 \times 25</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Phylogeny inference</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DNAdist+FastME</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>PhyML</td>
<td>8mn</td>
<td>4mn</td>
</tr>
<tr>
<td><strong>Dates and rate estimation</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LD</td>
<td>0.1</td>
<td>&lt;0.1</td>
</tr>
<tr>
<td>LD*</td>
<td>&lt;0.1</td>
<td>&lt;0.1</td>
</tr>
<tr>
<td>QPD</td>
<td>0.2</td>
<td>&lt;0.1</td>
</tr>
<tr>
<td>QPD*</td>
<td>&lt;0.1</td>
<td>&lt;0.1</td>
</tr>
<tr>
<td>Root-to-tip</td>
<td>&lt;0.1</td>
<td>&lt;0.1</td>
</tr>
<tr>
<td>Root-to-tip*</td>
<td>&lt;0.1</td>
<td>&lt;0.1</td>
</tr>
<tr>
<td>LF*</td>
<td>3.5</td>
<td>1.5</td>
</tr>
<tr>
<td>BEAST with a strict molecular clock</td>
<td>4h</td>
<td>3h</td>
</tr>
<tr>
<td>BEAST with a relaxed molecular clock</td>
<td>17h</td>
<td>12h</td>
</tr>
</tbody>
</table>

*outgroup-based rooted tree

**Table S1. Computing time with simulated data sets.** The time is expressed in seconds, except otherwise specified. Two types of trees are used: 750/11 \times 10 with a death ratio of 750/1,000 and 110 leaves; 995/3 \times 25 with a death ratio of 995/1,000 and 75 leaves. All calculations have been performed on our server: Intel(R) Xeon(R) X5650 @ 2.67GHz, single core, no parallelization.
Supplementary Table S2. Detailed estimation results with data simulated with a strict molecular clock (SMC). This table displays the errors of the various methods in estimating the root date (tMRCA) and the substitution rate, for the four type of trees. Two values are given in each cell: the relative error on top, and the relative bias on bottom. See paper for the definitions of these measures and the details and options of the tested methods. See also legend of Figure 2. The formula used to compute the summary value of the relative error is:

$$\frac{1}{400} \sum_{i=1}^{100} \left( \frac{x - x_i^A}{x_A^2} \right)^2 + \sum_{i=1}^{100} \left( \frac{x - x_B^i}{x_B^2} \right)^2 + \sum_{i=1}^{100} \left( \frac{x - x_C^i}{x_C^2} \right)^2 + \sum_{i=1}^{100} \left( \frac{x - x_D^i}{x_D^2} \right)^2$$

For the substitution rate: $x = x_A = x_B = x_C = x_D = 0.006$ is the true substitution rate; $x_A^i$ is the estimated value for the ith data set (among 100) with tree type $X$. For the tMRCA: the four tree types are denoted as $A = 750\{3\times25}, B = 750\{11\times10}, C = 995\{3\times25}, and D = 995\{11\times10}; x = 0$ is the true root date; $x_A^i$ is the estimated root date with the ith data set (among 100) with tree type $X$; $x_A$ is the date (in years) of the most recent tips with tree type $X$, that is, $x_A = 66.833416, x_B = 29.366668, x_C = 32.466935, x_D = 22.493387$. For the global relative bias we use the standard arithmetic mean of the four relative biases of the four data sets.
Supplementary Table S3. Detailed estimation results with data simulated with a lognormal relaxed molecular clock (RMC). See legend of Figure 2 and Table S2.
### Supplementary Table S4. Topological errors with simulated data sets.

This table displays the Robinson and Foulds topological distance between the true and inferred trees, for the various tree building methods and tree models. This distance is equal to the number of bipartitions (branches) that are in one tree but not in the other, divided by the total number of bipartitions; it is equal to 0.0 when both trees are identical, and 1.0 when they do not share any bipartition in common. Results are averaged over the 100 data sets corresponding to each tree model. BSMC: BEAST with strict molecular clock model; BRMC: BEAST with lognormal relaxed molecular clock. As expected likelihood-based methods (PhyML and BEAST) have a better topological accuracy than (distance-based) FastME. BEAST (incorporating a tree model constraining branch lengths) is slightly more accurate than PhyML.

<table>
<thead>
<tr>
<th></th>
<th>Strict molecular clock (SMC)</th>
<th>Relaxed molecular clock (RMC)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PhyML</td>
<td>FastME</td>
</tr>
<tr>
<td>750/3×25</td>
<td>0.06</td>
<td>0.12</td>
</tr>
<tr>
<td>750/11×10</td>
<td>0.08</td>
<td>0.14</td>
</tr>
<tr>
<td>995/3×25</td>
<td>0.08</td>
<td>0.12</td>
</tr>
<tr>
<td>995/11×10</td>
<td>0.17</td>
<td>0.19</td>
</tr>
<tr>
<td><strong>Summary</strong></td>
<td><strong>0.10</strong></td>
<td><strong>0.14</strong></td>
</tr>
</tbody>
</table>
Supplementary Table S5. Origins and dates of influenza A H1N1pdm09 strains.