A MASSIVELY PARALLEL BRANCH-&-BOUND ALGORITHM FOR THE BALANCED MINIMUM EVOLUTION PROBLEM

Daniele Catanzaro, Martin Frohn, Olivier Gascuel, Raffaele Pesenti

REPRINT | 3274

UCLouvain

CORE | LIDAM
Louvain Institute of Data Analysis and Modeling in economics and statistics

Computers & Operations Research 158 (2023) 106308

Contents lists available at ScienceDirect

Computers and Operations Research

journal homepage: www.elsevier.com/locate/cor

A massively parallel branch-&-bound algorithm for the balanced minimum evolution problem

Daniele Catanzaro [a], Martin Frohn [b,*], Olivier Gascuel [c], Raffaele Pesenti [d]

[a] CORE, Université Catholique de Louvain, Voie du Roman Pays 34, L1.03.01, B-1348 Louvain-la-Neuve, Belgium
[b] Department of Mathematics and Computer Science, Eindhoven University of Technology, De Groene Loper 5, 5612 AZ Eindhoven, Netherlands
[c] Institut de Systématique, Evolution, Biodiversité (ISYEB - UMR 7205 CNRS & Muséum National d'Histoire Naturelle), Paris, France
[d] Department of Management, Università Ca' Foscari, San Giobbe, Cannaregio 837, I-30121 Venezia, Italy

ARTICLE INFO

Keywords:
Combinatorial optimization
Integer programming
Parallel branch-and-bound
Network design
Balanced minimum evolution problem

ABSTRACT

We build upon recent theoretical advances in the *Balanced Minimum Evolution Problem* (BMEP) to design a new massively parallel exact solution algorithm that proves to be up to one order of magnitude faster than the current state-of-the-art under the same computing settings and environment.

1. Introduction

An *Unrooted Binary Tree* (UBT) is a tree having $n \geq 3$ leaves, $n-2$ internal vertices, each of degree 3, and $2n-3$ edges, n of which are external, i.e., incident to leaves, and $n-3$ are internal, i.e., incident to two internal vertices. An UBT can be encoded through a *Path-Length Matrix* (PLM) τ, i.e., a square symmetric matrix of order n, whose generic entry τ_{ij} denotes the number of edges on the (unique) path between leaves i and j (Catanzaro et al., 2022). Fig. 1 shows an example of an UBT with 5 leaves and its associated PLM τ.

The *Balanced Minimum Evolution Problem* (BMEP) is a *Network Design Problem* (Pop, 2012) that is defined over UBTs and that has been much studied in the literature on molecular phylogenetics (Gascuel, 2005; Catanzaro et al., 2022). Given a set $\Gamma = \{1, 2, \ldots, n\}$ of $n \geq 3$ vertices, hereafter referred to as *taxa*, and a square symmetric matrix $\mathbf{D}$ of order n, whose generic entry d_{ij} denotes a distance between the pair of taxa $i, j \in \Gamma$ (see Fig. 2), the BMEP consists in finding an UBT T that has Γ as leafset and that minimizes the *length function*

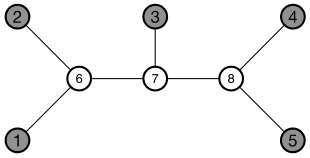
$$L(T) = \sum_{i \in \Gamma} \sum_{j \in \Gamma \setminus \{i\}} \frac{d_{ij}}{2^{\tau_{ij}}}. \tag{1}$$

In the context of molecular phylogenetics, taxa encode distinct aligned molecular sequences (such as DNA, RNA, or codon sequences) extracted from selected species; the entries d_{ij} of $\mathbf{D}$ encode estimates of the *evolutionary distances* (Felsenstein, 2004; Page and Holmes, 1998) between pairs of taxa; and the optimal UBT (usually referred to as the optimal *phylogeny*) encodes evolutionary relationships inferred from the evolutionary distances of taxa. The optimal solution T^* to the BMEP provably minimizes the cross-entropy associated with the molecular sequences of taxa and if $\mathbf{D}$ encodes consistent estimates of the evolutionary distances of taxa then T^* provides a *statistically consistent estimate* of their evolutionary relationships (Catanzaro et al., 2020a; Felsenstein, 2004; Gascuel, 2005; Gascuel and Steel, 2006; Catanzaro et al., 2022). This information, in turn, is central in many branches of medicine and life sciences (Catanzaro et al., 2022). Curious readers interested in an introduction to the problem, to its biological motivations, as well as in a review of the recent advances in its combinatorics and optimization aspects may refer to Catanzaro et al. (2022).

Besides being a network design problem defined over UBTs, the BMEP can be also seen as: (i) a *Network Realization Problem* (NRP) (Batagelj et al., 1990; Buneman, 1971, 1974); (ii) a restriction of the *Minimum Evolution Problem* (Catanzaro et al., 2015); (iii) a version of the *Euclidean Steiner tree problem* (Cheng and Du, 2001; Gusfield, 1984; Lu et al., 2003; Prömel and Steger, 2002; Du et al., 2000; Hwang et al., 1992; Cieslik, 1998; Li and Mao, 2016; Wu and Chao, 2004; Du and Hu, 2008); and (iv) a generalization of the *Quadratic Assignment Problem* (QAP) (Çela, 1997), in which one of the two matrices involved in the Shur product must be determined within a factor of the set of matrices $\tau = \{\tau_{ij}\}$ that encode UBTs with n leaves (Aringhieri et al., 2011). The BMEP is $\mathcal{NP}$-hard and inapproximable within c^n, for some constant $c > 1$, unless $\mathcal{P} = \mathcal{NP}$ (Fiorini and Joret, 2012). This

* Corresponding author.
E-mail addresses: daniele.catanzaro@uclouvain.be (D. Catanzaro), m.frohn@tue.nl (M. Frohn), olivier.gascuel@mnhn.fr (O. Gascuel), pesenti@unive.it (R. Pesenti).

https://doi.org/10.1016/j.cor.2023.106308
Received 9 January 2023; Received in revised form 3 April 2023; Accepted 7 June 2023
Available online 10 June 2023



$$\tau = \begin{pmatrix} 0 & 2 & 3 & 4 & 4 \\ 2 & 0 & 3 & 4 & 4 \\ 3 & 3 & 0 & 3 & 3 \\ 4 & 4 & 3 & 0 & 2 \\ 4 & 4 & 3 & 2 & 0 \end{pmatrix}$$

Fig. 1. An example of an UBT with 5 leaves (marked in dark gray) and the associate PLM τ. The generic entry τ_{ij} of τ denotes the number of edges on the path between the pair of leaves i and j. For example, $\tau_{12} = 2$, $\tau_{24} = 4$, and $\tau_{35} = 3$.

negative result persists even in the case the underlying UBT is fixed and just the assignment of taxa to the leaves of U that minimizes (1) must be identified (Frohn, 2020). The problem is instead solvable in polynomial-time if the input distance matrix $\mathbf{D}$ is *additive*, i.e., if its entries satisfy the following condition (Gascuel, 2005):

$$d_{ij} + d_{kr} \leq \max\{d_{ik} + d_{jr}, d_{ir} + d_{jk}\} \quad \forall\, i, j, k, r \in \Gamma.$$

In the case $\mathbf{D}$ is just *metric*, i.e., if its entries satisfy just the triangle inequality, then the optimal solution to the BMEP can be approximated within a factor of two (Fiorini and Joret, 2012).

The last 20 years registered extensive research efforts focused on the characterization of several theoretical and algorithmic aspects of the BMEP, with the view of developing exact and approximate solution algorithms of practical use. In particular, the literature includes advances in the statistical consistency of the BMEP (Gascuel, 2005; Desper and Gascuel, 2004; Gascuel and Steel, 2006), in its connections with information theory (Catanzaro et al., 2020a), in its combinatorics (Catanzaro et al., 2012; Semple and Steel, 2004; Cueto and Matsen, 2011; Haws et al., 2011; Catanzaro et al., 2020b; Forcey et al., 2015, 2017), as well as on the development of implicit enumeration algorithms (Catanzaro et al., 2012; Pardi, 2009; Aringhieri et al., 2011) and heuristics (Pardi, 2009; Fiorini and Joret, 2012; Gascuel, 2005). This article further adds to this literature in that it presents a massively parallel exact solution algorithm that sets a new benchmark in this specific area.

The current state-of-the-art exact solution algorithm for the BMEP was proposed by Catanzaro et al. (2012) in 2012. The theoretical foundations of this algorithm (briefly summarized in Section 3) lie in the deep connections between the BMEP and information theory (Parker and Ram, 1996; Catanzaro et al., 2020a). These connections allowed the authors to identify some fundamental combinatorial equalities characterizing UBTs and to derive both tight lower bounds on the optimal solution to the problem and a valid integer linear programming formulation able to solve instances of the BMEP containing roughly two dozen taxa within a reference time limit of one hour. In biological and medical applications, the time limit can be arbitrarily extended as far as it remains practical. Finding the optimum to a given instance of the BMEP, or approximating it as close as possible, is instead highly desirable due to the previously mentioned relationships between the optimality of a solution and its statistical consistency (Catanzaro et al., 2022). Practical instances of the BMEP may include tens, hundreds, or even thousands of taxa (see, e.g., SARS-CoV-2 genomes available at https://www.ncbi.nlm.nih.gov/genbank/sars-cov-2-seqs/ and Lefort et al., 2015; Gascuel and Steel, 2006; Pardi, 2009; Desper and Gascuel, 2008; Auch et al., 2006; Yang, 2014; Pardi and Gascuel, 2012; Pardi et al., 2010; Pardi and Gascuel, 2016; Criscuolo and Michel, 2009; Jung, 2012; Bordewich et al., 2009). Developing exact solution algorithms able to assist in practical phylogenetic analyses is therefore of primary importance at least when a certificate of statistical consistency of the predicted evolutionary relationships of taxa is demanded. Despite the recent advances in the polyhedral combinatorics of the BMEP (Catanzaro et al., 2012; Semple and Steel, 2004; Cueto and Matsen, 2011; Haws et al., 2011; Catanzaro et al., 2020b; Forcey et al., 2015, 2017), however, the scientific community still struggles to design algorithms having a performance superior to Catanzaro et al. (2012). This article derives from the effort to push to the limit the performance of the current state-of-the-art by building upon the results described in Catanzaro et al. (2012), Semple and Steel (2004), Catanzaro et al. (2020b, 2015), Aringhieri et al. (2011), Catanzaro and Pesenti (2019), Catanzaro et al. (2020a), Gascuel and Steel (2006) and Gascuel and Steel (2016) and by taking advantage of specific tree-coding schemes, mixed integer programming, and the multicore (shared-memory) features of modern CPUs. The use of massive parallelism in the context of the BMEP is quite recent. We introduced it in Catanzaro and Pesenti (2019) to enumerate by brute-force all of the vertices of the BMEP polyhedron, by enabling so the analysis of its polyhedral combinatorics discussed in Catanzaro et al. (2020b). In that work, we encoded phylogenies as Prüfer codes (Caminiti et al., 2007) and we showed algorithms and methods to parallelize this enumeration both on CPUs and GPUs. We show in this article that the combination of massive parallelism with new theoretical results on the polyhedral combinatorics of the BMEP (Catanzaro et al., 2020b) and on its relationships with information theory (Catanzaro et al., 2020a) allows us to define a new reference in terms of performance that can be concisely summarized as follows: up to one order of magnitude faster than the current state-of-the-art exact sequential algorithm in the same computing environment on generic instances of the BMEP and up to 25% more taxa solved within a prefixed time limit.

The paper is organized as follows. In the next section, we introduce some notation and definitions that will prove useful throughout the article as well as some fundamental equations that describe the combinatorial properties of phylogenies. In Section 2, we introduce some notation and discuss fundamental properties of path-lengths. In Section 3, we briefly recall, for the sake of completeness, the current state-of-the-art sequential exact solution algorithm for the BMEP. In Sections 4 and 5, we discuss possible strategies to obtain lower bounds and upper bounds on the optimal solution to the BMEP, respectively. In Section 6, we develop a new massively parallel exact solution algorithm for the BMEP. Finally, in Section 7, we report on its performance on a set of benchmark instances taken from the literature and with respect to the current state-of-the-art.

2. Notation and properties of path-lengths

We introduce here some notation, definitions, and basic results from the literature that will prove useful throughout the article. We start by denoting $V(T)$ and $E(T)$ as the set of vertices and the set of edges of an UBT T respectively. Moreover, for a given subset of taxa $S \subseteq \Gamma$, we call an UBT T of S a *sub-phylogeny* of Γ and write T_S instead of T to indicate that we involve just the taxa in S, i.e., T_S is defined as a pair constituted by an UBT with $|S|$ leaves. We denote $V(T_S)$ and $E(T_S)$

Taxa	Sequence		
Taxon 1	A	A	A
Taxon 2	A	C	C
Taxon 3	C	G	C
Taxon 4	C	C	G
Taxon 5	G	A	G

$$\mathbf{D} = \begin{pmatrix} 0 & 2 & 3 & 3 & 2 \\ 2 & 0 & 2 & 2 & 3 \\ 3 & 2 & 0 & 2 & 3 \\ 3 & 2 & 2 & 0 & 2 \\ 2 & 3 & 3 & 2 & 0 \end{pmatrix}$$

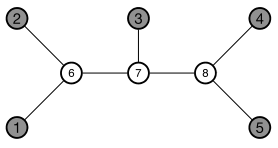


Fig. 2. A small example of a set Γ of five DNA sequences (*taxa*) and of a possible distance matrix $\mathbf{D}$ associated with Γ. In this specific case, the entries of $\mathbf{D}$ encode Hamming distances between pairs of molecular sequences. The UBT on the right encodes a phylogeny of Γ and provably minimizes the length function (1).

as the sets of internal vertices and edges of T_S, respectively. Moreover, whenever the context will be sufficiently clear for the sake of notation we will drop the index S from T. For example, we shall write just T whenever $S = \Gamma$.

Given two integers α and β, with $\alpha < \beta$, we denote $[\alpha, \beta]$ as the discrete interval constituted by the integers included between α and β, and similarly to Parker and Ram (1996), we define a *sequence* as a collection of nonnegative real values such as $\mathbf{s} = [s_1, s_2, \dots, s_m]$, $s_j \in \mathbb{R}$. Repetition of values in the sequence is permitted, as the reals s_j do not need to be pairwise distinct. We denote by $\uparrow(\mathbf{s})$ and $\downarrow(\mathbf{s})$ the sequences obtained from $\mathbf{s}$ by sorting its entries in non-decreasing order and non-increasing order, respectively, and we shall extend this notation to any vector in $\mathbb{R}^m$.

Given an UBT T of Γ and a taxon $i \in \Gamma$, we denote Γ_i as the set $\Gamma \setminus \{i\}$ and we define the *path-length sequence* τ_i as the ordered sequence $\mathbf{s}$ such that each entry s_j of $\mathbf{s}$ encodes the length of the unique path in T connecting taxon i to each taxon $j \in \Gamma_i$, i.e., $\tau_i = [\tau_{ij} \in [2, n-1] : j \in \Gamma_i]$. The ith row of the PLM τ associated with T can be seen as the path-length sequence τ_i when the ith entry in that row is neglected. Also note that τ_i describes the UBT T as seen from (or *rooted in*) taxon i. If a PLM τ contains a row $\tau_i = [2, 3, \dots, (n-2), (n-1), (n-1)]$, then we call the UBT encoded by τ a *caterpillar*.

Fixed a taxon $i \in \Gamma$, we denote Θ_i as the set of the path-length sequences τ_i encoding UBTs of Γ rooted in i. Moreover, we denote Θ as the set of PLMs τ encoding UBTs with n leaves and, for a fixed $\tau \in \Theta$, we define $2^{-\tau}$ as the matrix obtained from τ by setting its non-diagonal entries to $2^{-\tau_{ij}}$. Finally, we define $2^{-\Theta} = \{2^{-\tau} : \tau \in \Theta\}$. Characterizing the set Θ may enable compact mathematical programming formulations for the BMEP. At present, however, this task remains an open problem. The literature just provides the necessary conditions that a symmetric square matrix τ of order n must satisfy to encode the PLM of an UBT. Specifically, because UBTs are (non-oriented) acyclic graphs, the entries of τ must satisfy the following conditions:

$$\tau_{ij} \in [2, n-1] \qquad \forall\, i, j \in \Gamma : i \neq j \tag{2}$$

$$\tau_{ii} = 0 \qquad \forall\, i \in \Gamma \tag{3}$$

$$\tau_{ij} = \tau_{ji} \qquad \forall\, i, j \in \Gamma : i < j \tag{4}$$

$$\tau_{ij} + \tau_{jk} - \tau_{ik} \geq 2 \qquad \forall\, i, j, k \in \Gamma : i \neq j \neq k. \tag{5}$$

Condition (5) imposes the satisfaction of the *triangle inequality* (the right-hand side of (5) arises from the constraints $\tau_{ij} \geq 2$ in condition (2)). A further necessary condition that τ must satisfy is known as *Kraft's equality* (Catanzaro et al., 2012, 2020b):

$$\sum_{j \in \Gamma_i} 2^{-\tau_{ij}} = \frac{1}{2} \quad \forall\, i \in \Gamma. \tag{6}$$

Kraft's equality captures in a closed form both the connectivity of a rooted UBT and the degree constraint on its internal vertices. An UBT corresponding to a given path-length sequence $\boldsymbol{\tau}_i = [\tau_{ij} \in [2, n-1] : j \in \Gamma_i]$, for some $i \in \Gamma$, can be easily reconstructed by sorting $\boldsymbol{\tau}_i$ in ascending order and by drawing the path-lengths from i to all of the remaining taxa in Γ_i. However, it is worth noting that no bijective relation between the set Θ_i and the set of UBTs can be defined because a path-length sequence may correspond to multiple distinct UBTs. For example, Fig. 3 shows that there exist three possible UBTs that can be reconstructed from the path-length sequence $\boldsymbol{\tau}_1 = [3, 3, 4, 4, 4, 4]$.

It is worth noting that, up to a factor 2, each entry $2^{-\tau_{ij}}$ in (6) can be interpreted as a probability. This insight has been exploited in Catanzaro et al. (2012, 2020b,a) to identify a further condition that the entries of any PLM τ must satisfy to belong to Θ, namely

$$\sum_{i \in \Gamma} \sum_{j \in \Gamma_i} \tau_{ij} 2^{-\tau_{ij}} = (2n-3). \tag{7}$$

Eq. (7), usually referred to as the *phylogenetic manifold*, relates the path-lengths τ_{ij} to their *path-length densities* $2^{-\tau_{ij}}$ (Parker and Ram, 1996), and encodes the geometric locus of the matrices $2^{-\tau}$ having constant cross-entropy (see Catanzaro et al., 2020a, for more information). Conditions (2)–(7) are independent and necessary to characterize Θ (Catanzaro et al., 2020b).

3. On the state-of-the-art exact solution algorithm for the BMEP

The space of path-lengths τ_{ij} contains the minimal necessary information to formulate the BMEP in terms of mathematical programming. Kraft equalities (6) and the phylogenetic manifold (7), however, have a nonlinear expression in τ_{ij}. A possible attempt to overcome this issue was proposed by Forcey et al. (2015) through the introduction of variables

$$x_{ij} = 2^{n-1-\tau_{ij}} \quad \forall\, i, j \in \Gamma, i \neq j. \tag{8}$$

In this space, indeed, Kraft equalities (6) become linear:

$$\sum_{j \in \Gamma_i} x_{ij} = 2^{n-2} \quad \forall\, i \in \Gamma.$$

The expression of the phylogenetic manifold, however, remains nonlinear and it seems hard to find a space in which both Kraft equalities (6) and the phylogenetic manifold (7) linearize at the same time. The discretization approach proposed in Catanzaro et al. (2012) constitutes a possible way around this issue. Specifically, denoted $\mathcal{L} = \{2, \dots, n-1\}$ as the set of values taken by the generic path-length τ_{ij}, path-lengths can be modeled using the following binary decision variables

$$x_{ij}^{\ell} = \begin{cases} 1 & \text{if } \tau_{ij} = \ell \\ 0 & \text{otherwise} \end{cases} \quad \forall\, i, j \in \Gamma, i \neq j,\ \ell \in \mathcal{L}.$$

Thus, provided the satisfaction of the following *convexity constraint*

$$\sum_{\ell \in \mathcal{L}} x_{ij}^{\ell} = 1 \quad \forall\, i, j \in \Gamma,\ i \neq j \tag{9}$$

ensuring the existence of exactly one path-length connecting each distinct pair of taxa $i, j \in \Gamma$ in any feasible solution to the problem, in this space the objective function (1) can be written as

$$z = \sum_{i \in \Gamma} \sum_{j \in \Gamma_i} d_{ij} \left(\sum_{\ell \in \mathcal{L}} 2^{-\ell} x_{ij}^{\ell} \right)$$

and conditions (4), (6), and (7) become

$$x_{ij}^{\ell} = x_{ji}^{\ell} \qquad \forall\, i, j \in \Gamma, i \neq j, \forall\, \ell \in \mathcal{L} \tag{10}$$

$$\sum_{j \in \Gamma_i} \sum_{\ell \in \mathcal{L}} 2^{-\ell} x_{ij}^{\ell} = \frac{1}{2} \qquad \forall\, i \in \Gamma \tag{11}$$

$$\sum_{i \in \Gamma} \sum_{j \in \Gamma_i} \sum_{\ell \in \mathcal{L}} \ell 2^{-\ell} x_{ij}^{\ell} = (2n-3). \tag{12}$$

The resulting formulation

Formulation 1.

$$\min \quad z = \sum_{i \in \Gamma} \sum_{j \in \Gamma_i} d_{ij} \left(\sum_{\ell \in \mathcal{L}} 2^{-\ell} x_{ij}^{\ell} \right) \tag{13}$$

$$s.t. \quad \sum_{\ell \in \mathcal{L}} x_{ij}^{\ell} = 1 \qquad \forall\, i, j \in \Gamma, i \neq j \tag{14}$$

$$x_{ij}^{\ell} = x_{ji}^{\ell} \qquad \forall\, i, j \in \Gamma, i \neq j, \forall\, \ell \in \mathcal{L} \tag{15}$$

$$\sum_{j \in \Gamma_i} \sum_{\ell \in \mathcal{L}} 2^{-\ell} x_{ij}^{\ell} = \frac{1}{2} \qquad \forall\, i \in \Gamma \tag{16}$$

$$\sum_{i \in \Gamma} \sum_{j \in \Gamma_i} \sum_{\ell \in \mathcal{L}} \ell 2^{-\ell} x_{ij}^{\ell} = (2n-3) \tag{17}$$

$$x_{ij}^{\ell} \in [2, n-1] \qquad \forall\, i, j \in \Gamma, i \neq j, \forall\, \ell \in \mathcal{L} \tag{18}$$

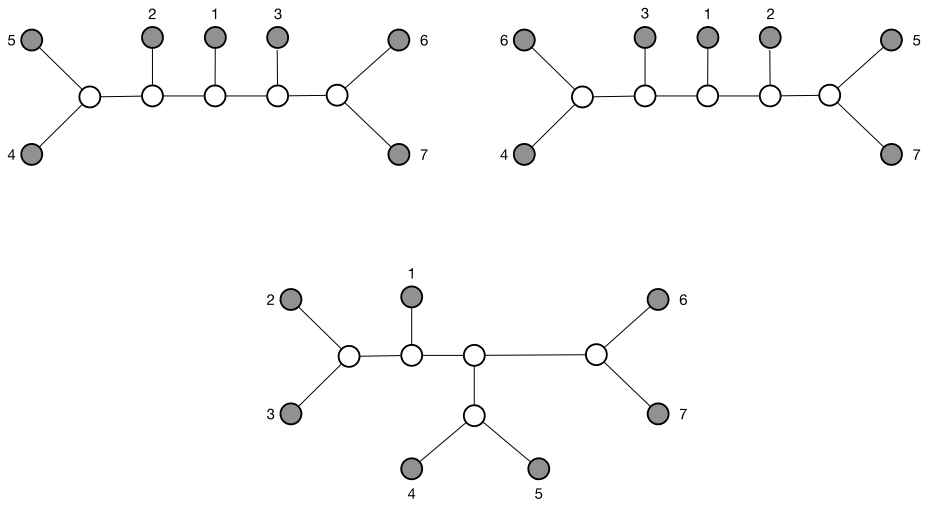
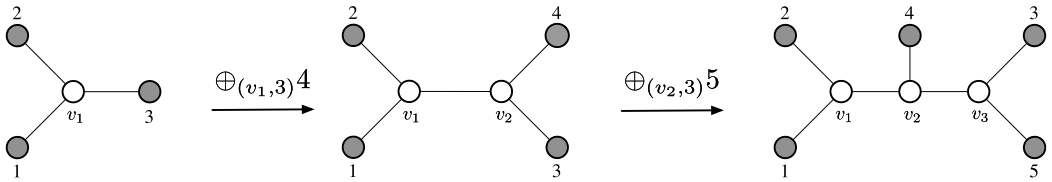


Fig. 3. An example of the three distinct UBTs that can be reconstructed from the path-length sequence $\tau_1 = [3, 3, 4, 4, 4, 4]$.



$\tau_{12} = \tau_{13} = \tau_{23} \geq 2$

$\tau_{34} \geq 2$
$\tau_{13} = \tau_{14} = \tau_{23} = \tau_{24} \geq 3$

$\tau_{35} \geq 2$
$\tau_{34} = \tau_{45} \geq 3$
$\tau_{15} = \tau_{25} \geq 4$

Fig. 4. An example of two consecutive insertions on a sub-phylogeny of three taxa. The inequalities indicate restrictions on path-lengths between pairs of taxa for further insertions.

is not valid in general for the BMEP as constraints (9)–(12) are just necessary but possibly not sufficient. A possible way to recover validity consists of using specific branching rules on variables x ensuring the enumeration of all and only UBTs of Γ. The branching rules proposed in Catanzaro et al. (2012) simulate Hendy and Penny (1982)'s *Stepwise Addition Strategy* (SAS), which provably enumerates all of the possible UBTs of Γ by using the concept of *insertion* of a taxon on a given sub-phylogeny of Γ. Specifically, given a subset of taxa $S \subset \Gamma$, a sub-phylogeny T_S of Γ, a taxon $t \in \Gamma \setminus S$, an internal vertex $v \in I \setminus V_i(T_S)$, and an edge $(a, b) \in E(T_S)$, an *insertion* of taxon t into the edge $e = (a, b)$ of T_S is the sub-phylogeny $T_{S\cup\{t\}}$ defined by

$$E\left(T_{S\cup\{t\}}\right) = E\left(T_S\right) \setminus \{(a,b)\} \cup \{(a,v),(v,b),(v,t)\}.$$

For the sake of notation, we denote the above insertion operation as

$$T_{S\cup\{t\}} = T_S \oplus_e t.$$

An example of two consecutive insertions on a sub-phylogeny of three taxa is shown in Fig. 4. Taxon 4 is initially inserted into edge $(v_1, 3)$ of the given sub-phylogeny (i) by introducing a new internal vertex v_2; (ii) by removing the edge $(v_1, 3)$; and (iii) by adding edges (v_1, v_2), $(v_2, 3)$ and $(v_2, 4)$. Subsequently, taxon 5 is inserted into the new sub-phylogeny of four taxa by adding internal vertex v_3; by removing edge $(v_2, 3)$; and by adding edges (v_2, v_3), $(v_3, 3)$, $(v_3, 5)$.

The opposite of an insertion of a taxon t on a sub-phylogeny $T_{S\cup\{t\}}$ is referred to as a *removal* and, for an edge $e = (a, b) \in E(T_{S\cup\{t\}})$ and an internal vertex $v \in I \setminus V_i(T_{S\cup\{t\}})$, it is defined by

$$E\left(T_S\right) = E\left(T_{S\cup\{t\}}\right) \setminus \{(a,v),(v,b),(v,t)\} \cup \{(a,b)\}.$$

As for the insertion, we denote the removal operations as

$$T_{S\cup\{t\}} \ominus_e t.$$

Roughly speaking, a removal of taxon t can be obtained by deleting the last three edges from $E(T_{S\cup\{t\}})$ and by restoring the edge (a, b). The recursive application of the above definitions of an insertion and a removal of a taxon on a sub-phylogeny of Γ allows one to enumerate the phylogenies of Γ, e.g., using Algorithm 1. A graphical example of its execution is shown in Fig. 5.

Note that, fixed a pair of distinct taxa $i, j \in \Gamma$, a subset $S \subset \Gamma$, and a sub-phylogeny T_S of Γ, the net result of an insertion of taxon $t \notin S$ on some edge of T_S is the exclusion of specific values for the path-lengths τ_{ij}. Specifically, denoted $P_{ij}(T_S)$ as the set of the possible length values that can be taken by the unique path p_{ij} in a sub-phylogeny derived from T_S by a series of insertions, it holds that

$$P_{ij}(T_S) = \begin{cases} \{\ell \in \mathcal{L} \,:\, \tau_{ij} \leq \ell \leq \tau_{ij} + |\Gamma \setminus S|\} & \text{if } i,j \in S; \\ \{\ell \in \mathcal{L} \,:\, 2 \leq \ell \leq \max_{s\in S} \tau_{is} + |\Gamma \setminus S|\} & \text{if } i \in S, j \in \Gamma \setminus S; \\ \{\ell \in \mathcal{L} \,:\, 2 \leq \ell \leq \max_{s,t\in S} \tau_{st} + |\Gamma \setminus S|\} & \text{if } i,j \in \Gamma \setminus S. \end{cases} \tag{19}$$

Algorithm 1: Enumerate - Stepwise Addition Strategy

Input: An ordered set of taxa $\Gamma = \{1, \ldots, n\}$; a subset $S \subset \Gamma$; a sub-phylogeny T_S of Γ

```
if |S| = |Γ| then
    return;
t ← |S| + 1;
for each edge e ∈ E(T_S) do
    Enumerate(Γ, S, T_S ⊕_e t);
    T_S = T_S ⊖_e t;
```

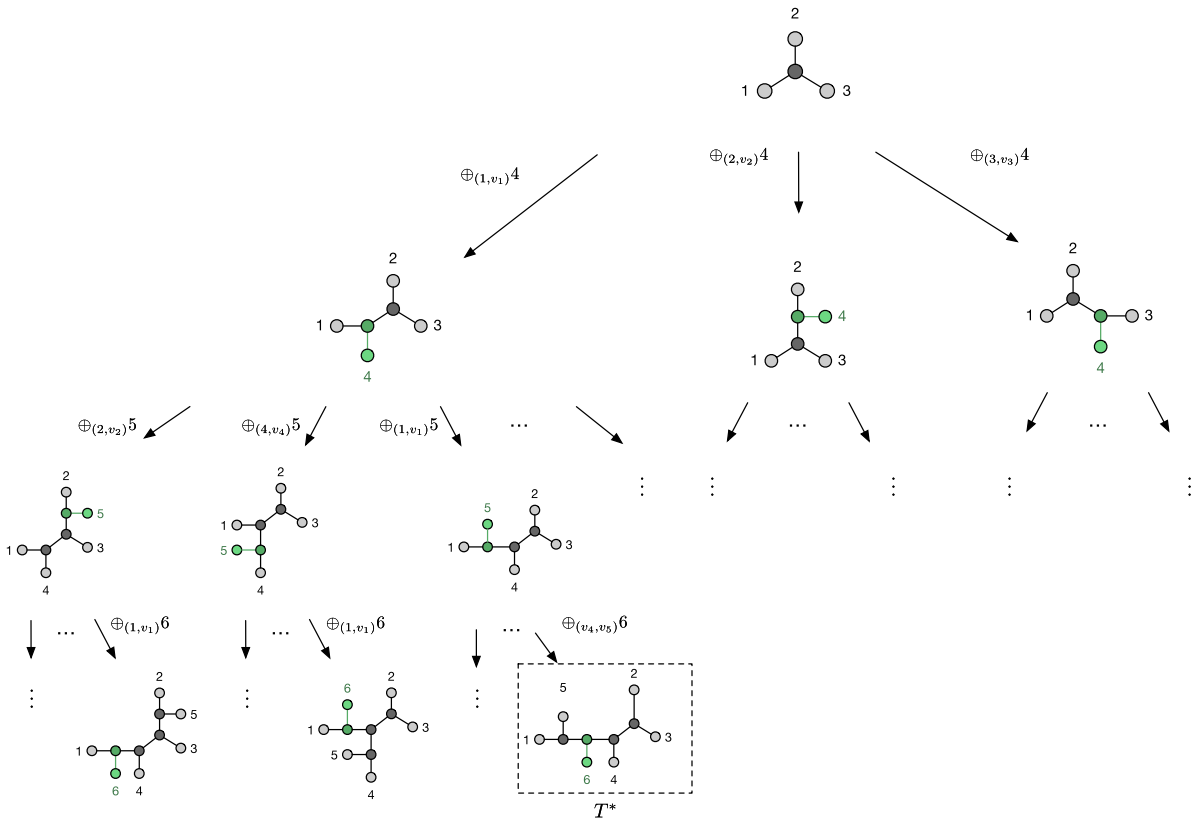


Fig. 5. An example of the enumeration tree of a SAS for a set Γ of six taxa.

This fact implies that, following an insertion of taxon t on T_S and fixed a pair of distinct taxa i and j in Γ, all of the x_{ij}^{ℓ} variables whose ℓ indices fall out of the index sets $P_{ij}(T_S)$ can be set to zero, by giving rise to the above mentioned SAS-based branching rules. For example, by referring to the insertion of taxon 5 in Fig. 4, we have that $x_{13}^3 = x_{23}^3 = x_{34}^2 = x_{15}^2 = x_{15}^3 = x_{25}^2 = x_{25}^3 = x_{45}^2 = 0$ and $x_{12}^{n-2} = x_{14}^{n-1} = x_{35}^{n-1} = x_{35}^{n-2} = x_{45}^{n-1} = 0$.

4. Lower bounds on the optimal solution to the BMEP

Identifying tight bounds on the optimal solution to a given discrete optimization problem is essential to develop effective exact solution algorithms of practical use. In this section, we report on known lower bounds on the optimal solution to the BMEP, we introduce new ones, and for the first time we compare them on benchmark instances of the problem.

The earliest lower bounds on the optimal solution to the BMEP was introduced by Pardi (2009) in 2009. Given a subset of taxa $S = \{1, 2, \dots, t-1\} \subset \Gamma$, a sub-phylogeny T_S of Γ, and a taxon $t \in \Gamma \setminus S$, the author first defined the quantities

$$\lambda_t^{ij} = \frac{1}{2}\left(d_{it} + d_{jt} - d_{ij}\right) \qquad \forall\, i, j \in S, i < j$$

as well as the vector $\lambda(t)$ having λ_t^{ij}, $i, j \in S$, $i < j$, as generic entry, sorted in non-decreasing order. Then, denoted $\lambda(t)_k$ as the kth entry of $\lambda(t)$, Pardi showed that

Proposition 1.

$$L\left(T_S \oplus_{(a,b)} t\right) - L\left(T_S\right) \geq \sum_{k=1}^{t-3} \frac{1}{2^k}\lambda(t)_k + \frac{1}{2^{t-3}}\lambda(t)_{(t-2)} \tag{20}$$

and

$$L\left(T^*\right) - L\left(T_S\right) \geq \sum_{t \notin S}\left(\sum_{k=1}^{t-3} \frac{1}{2^k}\lambda(t)_k + \frac{1}{2^{t-3}}\lambda(t)_{(t-2)}\right). \tag{21}$$

Pardi's bound (21) currently constitutes one of the best non-mathematical programming-based lower bounds for the BMEP (see, Tables 1 and 2 for more information).

A second set of lower bounds on the optimal solution to the BMEP was introduced by Catanzaro et al. (2020a) and is based on the fact that the length function (1) is invariant under a particular rescaling of the input distance matrix $\mathbf{D}$. Specifically, denote K as a non-negative constant, $\mathbf{I}$ as the identity matrix of order n, $2^{-\mathbf{D}}$ as the $n \times n$ matrix having as generic entry $2^{-d_{ij}}$, $i, j \in \Gamma$, and $\hat{\mathbf{D}} = 2^{-\mathbf{D}} - \mathbf{I}$ as a rescaled form of the input distance matrix $\mathbf{D} \in \mathbb{R}^{n \times n}$ for which $\hat{\mathbf{D}}$ is doubly stochastic. As observed in Catanzaro et al. (2020a), $\hat{\mathbf{D}}$ exists and can be computed in polynomial time. Denote also $\hat{d}_{ij}$ as the generic entry of $\hat{\mathbf{D}}$ and consider the probability distributions

$$p_i = \left\{2^{1-\tau_{ij}} \;:\; j \in \Gamma_i\right\} \quad \text{and} \quad q_i = \left\{2^{-\hat{d}_{ij}} \;:\; j \in \Gamma_i\right\}$$

as well as their *Kullback–Leibler divergence*

$$D_{KL}(p_i \,\|\, q_i) = \sum_{j \in \Gamma_i}\left(-p_{ij}\log_2\left(q_{ij}\right) + p_{ij}\log_2\left(p_{ij}\right)\right).$$

Then, the length function (1) can be rewritten as

$$L(T) = \frac{1}{2}\sum_{i \in \Gamma}\left(D_{KL}\left(p_i \,\|\, q_i\right)\right) + \frac{3n-6}{2} + K$$

and the BMEP becomes a cross-entropy minimization problem (Catanzaro et al., 2020a). This reformulation enables the following result:

Proposition 2. *For any fixed scalar $\alpha \in [0, 1]$, the following lower bound holds on the optimal solution to the BMEP:*

$$L\left(T^*\right) \geq \frac{1}{2}\sum_{i \in \Gamma}\min_{\tau_i \in \Theta_i}\sum_{j \in \Gamma_i}\frac{\hat{d}_{ij} - \alpha(\tau_{ij} - 1)}{2^{\tau_{ij}-1}} + \alpha\frac{3n-6}{2} + K. \tag{22}$$

Alternative lower bounds on the optimal solution to the BMEP can be obtained by means of mathematical programming. Specifically,

consider the problem of minimizing the BMEP length function subject to Kraft equalities (16), i.e.,

Formulation 2.

$$\min \quad z = \sum_{i\in\Gamma}\sum_{j\in\Gamma_i} d_{ij}2^{-\tau_{ij}}$$

$$s.t. \quad \sum_{j\in\Gamma_i} 2^{-\tau_{ij}} = \frac{1}{2} \qquad \forall\, i \in \Gamma$$

$$\tau_{ij} \in [2, n-1] \qquad \forall\, i,j \in \Gamma, i \neq j.$$

Formulation 2 can obtained from Formulation 1, by removing all constraints but Kraft's, the convexity, and the integrality ones. We observe that the optimal solution to Formulation 2 coincides with the right-hand side of (22) when both $\alpha = 0$ and $\hat{\mathbf{D}} = \mathbf{D}$. Moreover, we also observe that from a combinatorial perspective, the optimal solution to Formulation 2 can be interpreted as a forest of caterpillar UBTs each rooted in taxon $i \in \Gamma$. In particular, the following proposition holds:

Proposition 3. *Let $\mathbf{d}_i$ denote the ith row of $\mathbf{D}$ and let τ_i be a path-length sequence in Θ_i. Then,*

(i) if τ_i is sorted in non-decreasing order then $\sum_{k=1}^{n} d_{ik}\cdot 2^{-\tau_{ik}} \geq \sum_{k=1}^{n} \uparrow (d_i)_k \cdot 2^{-\tau_{ik}}$;

(ii) $\tau_i' = [2,3,\ldots,(n-2),(n-1),(n-1)] = \arg\min\left\{\sum_{k=1}^{n} \uparrow (d_i)_k \cdot 2^{-\tau_{ik}} \;:\; \tau_i \in \Theta_i\right\}$.

Proof.

Part i. Assume that $\mathbf{d}_i$ is not sorted in non-decreasing order, otherwise the statement trivially follows. Then, there exist two indices, say $p, q \in [1,n]$, such that $p < q$ and $d_{ip} > d_{iq}$. As $2^{-\tau_{ip}} \geq 2^{-\tau_{iq}}$, then $d_{ip}2^{-\tau_{ip}} + d_{iq}2^{-\tau_{iq}} \geq d_{iq}2^{-\tau_{ip}} + d_{ip}2^{-\tau_{iq}}$. Hence, $\sum_{k=1}^{n} d_{ik}2^{-\tau_{ik}}$ gets smaller if d_{ip} and d_{iq} are swapped in d_i. This argument can be iterated until the entries of d_i are sorted in non-decreasing way.

Part ii. Let $\tau_i' = [2,3,\ldots,(n-1),(n-1)]$ denote a path-length sequence encoding an UBT with n leaves and rooted in i. Observe that

$$\sum_{k=1}^{l-1} 2^{-\tau'_{ik}} \geq \sum_{k=1}^{l-1} 2^{-\tau_{ik}} \quad \forall\, l = 2,\ldots,n \;\forall\, \tau_i \in \Theta_i. \tag{23}$$

Without loss of generality assume that d_i is already sorted in non-decreasing order. Then, we have that

$$\begin{aligned}
\sum_{k=1}^{n} d_{ik}\cdot 2^{-\tau_{ik}} &= d_{i1}\cdot\sum_{k=1}^{n} 2^{-\tau_{ik}} + \sum_{l=2}^{n}\left((d_l - d_{l-1})\cdot\sum_{k=l}^{n} 2^{-\tau_{ik}}\right)\\
&\overset{(6)}{=} \frac{d_{i1}}{2} + \sum_{l=2}^{n}\left((d_l - d_{l-1})\cdot\left(\frac{1}{2} - \sum_{k=1}^{l-1} 2^{-\tau_{ik}}\right)\right)\\
&\overset{(23)}{\geq} \frac{d_{i1}}{2} + \sum_{l=2}^{n}\left((d_l - d_{l-1})\cdot\left(\frac{1}{2} - \sum_{k=1}^{l-1} 2^{-\tau'_{ik}}\right)\right)\\
&\overset{(6)}{=} d_{i1}\cdot\sum_{k=1}^{n} 2^{-\tau'_{ik}} + \sum_{l=2}^{n}\left((d_l - d_{l-1})\cdot\sum_{k=l}^{n} 2^{-\tau'_{ik}}\right)\\
&= \sum_{k=1}^{n} d_{ik}\cdot 2^{-\tau'_{ik}}.
\end{aligned}$$

Thus, the statement follows. □

A second mathematical programming-based lower bound for the optimal solution to the BMEP can be obtained by minimizing the length function on the phylogenetic manifold (17), i.e., by solving the problem

Formulation 3.

$$\min \quad z = \sum_{i\in\Gamma}\sum_{j\in\Gamma_i} d_{ij}2^{-\tau_{ij}}$$

$$s.t. \quad \sum_{i\in\Gamma}\sum_{j\in\Gamma_i} \tau_{ij}2^{-\tau_{ij}} = 2n-3$$

$$2 \leq \tau_{ij} \leq n-1 \qquad \forall\, i,j \in \Gamma, i \neq j.$$

Formulation 3 can be obtained from Formulation 1 by relaxing the integrality constraints on τ_{ij} variables and by eliminating all constraints but the convexity and the phylogenetic manifold ones. We note that the optimal solution to Formulation 3 has a particular geometric interpretation. Specifically, let $\mathbf{O}$ denote a $n\times n$ matrix having $2^{-(n-1)}$ as a generic non-diagonal entry and 0 otherwise. Let $\mathbf{o}$ denote a $n(n-1)$-vector obtained from $\mathbf{O}$ by excluding the diagonal entries and by appending its rows one after the other. It is easy to realize that this vector identifies a point that is internal to the phylogenetic manifold. Let $\mathbf{d}$ denote the $n(n-1)$-vector obtained from $\mathbf{D}$ by excluding its diagonal entries and by appending its rows one after the other. Note that in general the point mapped by $\mathbf{d}$ can be internal, external or on the phylogenetic manifold itself. Finally, let $2^{-\tau}$ denote a $n(n-1)$-vector whose generic entry is $2^{-\tau_{ij}}$. Then, it is easy to see that solving Formulation 3 is equivalent to minimize the scalar product $\mathbf{d}\cdot 2^{-\tau}$ subject to having $2^{-\tau}$ mapping a point on the manifold. It is easy to see that this situation occurs when the angle between $\mathbf{d}$ and $2^{-\tau}$ is zero and that the point on the manifold for which this situation holds corresponds to the intersection between (the possible prolongation of) the segment $\overline{\mathbf{od}}$ and the phylogenetic manifold. In general, this intersection point is not integral; the BMEP, therefore, can be interpreted as the problem of finding a point $2^{-\tau}$ that maps an UBT and that is as close as possible to this intersection point. It is also worth noting that it is not possible to assess a priori the tightness of the lower bound provided by Formulation 2 with respect to the one provided by Formulation 3. In fact, consider the following input distance matrices

$$\mathbf{D} = \begin{pmatrix}
0 & 0.49 & 0.95 & 1.39 & 1.1 & 1.14 & 0.81\\
0.49 & 0 & 0.81 & 1.18 & 1.64 & 1.85 & 0.39\\
0.95 & 0.81 & 0 & 1.55 & 3.26 & 3.07 & 1.48\\
1.39 & 1.18 & 1.55 & 0 & 3.39 & 4.06 & 1.43\\
1.1 & 1.64 & 3.26 & 3.39 & 0 & 0.26 & 1.41\\
1.14 & 1.85 & 3.07 & 4.06 & 0.26 & 0 & 2.03\\
0.81 & 0.39 & 1.48 & 1.43 & 1.41 & 2.03 & 0
\end{pmatrix} \quad \text{and}$$

$$\mathbf{D}' = \begin{pmatrix}
0 & 3.45 & 3.45 & 3.21 & 3.45 & 3.1 & 3\\
3.45 & 0 & 2.19 & 1.88 & 1.76 & 2.42 & 2.12\\
3.45 & 2.19 & 0 & 0.29 & 1.88 & 1.76 & 2.42\\
3.21 & 1.88 & 0.29 & 0 & 1.56 & 1.47 & 2.19\\
3.45 & 1.76 & 1.88 & 1.56 & 0 & 0.93 & 0.71\\
3.1 & 2.42 & 1.76 & 1.47 & 0.93 & 0 & 0.73\\
3 & 2.12 & 2.42 & 2.19 & 0.71 & 0.73 & 0
\end{pmatrix}$$

and observe that the values of the optimal solutions to Formulations 2 and 3 are 3.272 and 3.357, respectively, when considering the distance matrix $\mathbf{D}$. In contrast, the values of the optimal solutions to Formulations 2 and 3 are 4.4 and 3.726, respectively, when considering $\mathbf{D}'$. We obtain a lower bound that is provably tighter than both Formulations 2 and 3 when merging both formulations and including also the symmetry constraint (15), i.e., when considering Formulation 1.

We refer to the value of the optimal solution to Formulation 1 as the *Linear Programming Lower Bound* (LPLB). Tables 1 and 2 report on the values of the lower bounds obtained when considering Catanzaro et al.'s benchmark instances of the BMEP (Catanzaro et al., 2006). The tables show that Formulation 1 always provides the tightest lower bound on the optimal solution to the BMEP. The tightness of the LPLB may justify its use in an implicit enumeration solution algorithm for the BMEP. It is worth noting here that the burden required to compute each of these lower bounds on a specific node of the search tree is per se negligible. However, this burden becomes considerable in the context of an implicit enumeration search and this phenomenon is particularly true when considering Formulation 1: as shown in Catanzaro et al. (2012), the iterated calls to the simplex algorithm throughout the search tree makes the computation of the LPLB so time consuming

Table 1
Values of the lower bounds at the root node of the search tree obtained when considering Catanzaro et al.'s benchmark instances of the BMEP Catanzaro et al. (2006). In order to include inequalities (22) in the comparison we set $K = 0$ (a valid lower bound for any K) and we choose an optimal scalar α by complete enumeration and for a machine epsilon equal to $\epsilon = 10^{-6}$.

Data set	Taxa	Optimum	Ineq. (21)	Ineq. (22)	Form. 3	Form. 1
Primates12	10	124.968	117.552	113.486	72.422	121.762
	11	332.834	308.570	283.810	184.726	324.211
	12	802.589	740.163	692.126	455.455	782.755
M17	10	105.134	100.189	90.639	72.929	104.033
	11	261.233	246.490	219.211	170.613	258.146
	12	541.632	508.402	456.433	349.652	535.065
	13	1,181.598	1,098.695	1,009.782	780.718	1,162.043
	14	2,408.066	2,236.155	2,050.144	1,545.662	2,368.216
	15	4,998.295	4,630.741	4,288.528	3,200.613	4,915.784
	16	10,225.560	9,371.332	8,752.214	6,455.794	10,051.994
	17	20,788.112	19,031.724	17,800.311	12,949.645	20,447.707
M18	10	190.331	164.735	166.916	145.096	186.288
	11	396.942	330.568	342.986	289.819	386.718
	12	805.937	667.791	686.222	567.978	784.056
	13	1,758.111	1,382.077	1,501.844	1,231.903	1,700.618
	14	3,599.678	2,763.052	3,008.283	2,431.544	3,469.282
	15	7,746.218	5,793.984	6,607.869	5,305.496	7,477.146
	16	15,973.016	11,754.719	13,721.437	11,093.488	15,411.761
	17	32,345.662	22,911.055	27,721.161	22,455.971	31,202.619
	18	66,029.112	45,790.686	56,264.712	44,556.665	63,683.315
SeedPlant25	10	91.458	77.449	84.284	77.494	86.549
	11	206.250	173.268	185.399	170.437	195.988
	12	427.816	356.940	391.828	360.248	407.022
	13	943.774	761.057	848.138	771.943	893.893
	14	1,929.219	1,553.194	1,749.409	1,591.185	1,831.570
	15	4,085.763	3,287.307	3,699.822	3,338.153	3,895.098
	16	8,353.457	6,334.814	7,446.740	6,598.428	7,872.086
	17	17,314.429	12,441.439	15,387.072	13,493.130	16,239.100
	18	36,156.527	25,344.199	31,811.347	27,666.698	33,654.096
	19	75,261.323	52,409.879	65,976.509	56,998.660	70,014.666
	20	160,913.646	109,511.370	139,432.212	119,602.162	149,517.567
M43	10	105.020	99.003	97.534	85.251	103.652
	11	209.282	196.591	189.736	164.251	204.268
	12	434.326	401.064	395.582	335.465	426.557
	13	895.424	823.990	812.050	680.809	879.878
	14	1,808.970	1,650.212	1,619.119	1,328.144	1,777.358
	15	3,965.234	3,569.271	3,517.331	2,852.487	3,904.670
	16	8,219.835	7,356.954	7,403.837	6,004.717	8,095.798
	17	16,798.759	14,985.303	15,134.213	12,336.357	16,493.312
	18	35,383.158	31,350.532	32,020.622	25,772.321	34,874.323
	19	71,769.903	63,521.426	65,160.787	52,611.255	70,743.386
	20	157,472.424	138,290.054	141,187.979	112,460.414	155,142.306
RbcL55	10	152.482	140.757	131.171	111.541	149.733
	11	328.645	303.619	286.910	243.473	323.349
	12	685.972	628.710	605.745	511.399	670.868
	13	1,502.870	1,335.970	1,328.099	1,104.865	1,467.890
	14	3,094.888	2,684.557	2,729.168	2,242.954	3,024.746
	15	6,448.259	5,558.499	5,697.917	4,678.343	6,289.820
	16	13,455.292	11,352.353	11,908.270	9,709.872	13,119.319
	17	27,804.246	22,685.482	24,499.350	19,643.587	26,987.288
	18	56,175.236	44,449.031	48,648.020	38,573.673	54,544.159
	19	114,564.570	88,318.788	98,119.885	76,318.718	111,223.724
	20	234,189.871	171,925.735	195,874.159	150,025.379	225,473.659

that an implicit enumeration algorithm based on Pardi's bound (Pardi, 2009) may prove to be even faster despite its poorer quality. One way to decrease the computational burden necessary to compute the LPLB consists of (i) reducing the size of Formulation 1 and (ii) reducing the number of times the LPLB is computed during the search tree. A dummy approach to implement point (i) consists of removing the symmetry constraints and declaring the variables x just for $i < j$. Finding an efficient way to implement point (ii) is instead a less trivial task. One option proposed in Catanzaro et al. (2012) consists of considering the Lagrangian relaxation of the reduced Formulation 1, i.e.,

Formulation 4 (*Lagrangian Lower Bound (LagLB)*).

$$\min \quad z_{lag}(T_S, \mu, \lambda) = \sum_{i,j \in \Gamma, i<j} \left(\sum_{\ell \in \mathcal{L} \setminus \{1\}} \left(2d_{ij} - \mu_i - \mu_j - 2\ell\lambda\right) 2^{-\ell} x_{ij}^{\ell} \right) + \sum_{i \in \Gamma} \frac{\mu_i}{2} + (2n-3)\lambda$$

$$s.t. \quad \sum_{\ell \in \mathcal{L} \setminus \{1\}} x_{ij}^{\ell} = 1 \quad \forall i, j \in \Gamma, i < j$$

$$x_{ij}^{\ell} \geq 0 \quad \forall i, j \in \Gamma, i < j, \forall \ell \in \mathcal{L} \setminus \{1\}.$$

Formulation 4 decomposes into a family of $n(n-1)/2$ independent subproblems whose analytical solution is trivial and fast to compute, provided suitable choices for the Lagrangian multipliers μ and λ. If the LPLB has been computed at a given node v of the search tree, then the shadow-prices of constraints (16) and (17), respectively, may constitute licit choices for μ and λ, respectively, and the lower bounds for the children nodes of v can be then computed by solving Formulation 4, hereafter denoted as LagLB, at those nodes. We have observed in computational experiments that the quality of the LagLB degrades

Table 2
Continuation of Table 1.

Data set	Taxa	Optimum	Ineq. (21)	Ineq. (22)	Form. 3	Form. 1
M62	10	163.876	158.801	141.791	117.649	162.117
	11	350.425	338.429	302.397	248.494	345.178
	12	753.821	725.293	647.649	525.766	738.269
	13	1,580.764	1,514.172	1,358.747	1,091.071	1,545.823
	14	3,345.564	3,192.456	2,883.060	2,287.762	3,267.091
	15	7,161.078	6,830.675	6,164.045	4,843.656	6,972.194
	16	14,980.693	14,288.715	13,011.075	10,160.761	14,599.862
	17	31,293.082	29,723.927	27,300.251	21,186.236	30,511.422
	18	66,187.974	62,386.458	58,765.435	45,177.687	64,773.333
	19	145,642.424	136,472.180	127,537.285	96,765.463	142,447.095
	20	296,651.044	277,545.708	261,991.533	203,310.439	290,179.592
Rana64	10	41.223	39.010	36.463	33.271	38.859
	11	87.162	81.538	79.042	74.004	82.854
	12	183.042	166.237	167.510	158.827	174.551
	13	382.700	342.310	347.048	331.373	362.074
	14	773.810	681.998	691.220	657.890	730.466
	15	1,603.120	1,394.179	1,410.437	1,320.593	1,499.345
	16	3,290.745	2,788.659	2,822.970	2,563.864	3,034.799
	17	6,745.447	5,722.298	5,710.143	5,038.760	6,205.151
	18	15,435.268	13,014.861	13,116.981	11,113.052	14,652.750
	19	36,052.455	30,943.295	30,728.166	24,488.000	34,437.478
	20	81,105.131	69,694.729	70,151.715	53,684.172	77,259.420
M82	10	53.170	45.653	47.333	42.694	50.834
	11	106.443	85.710	90.742	77.799	101.633
	12	225.836	181.823	197.266	171.660	215.831
	13	543.127	438.867	463.930	390.489	521.854
	14	1,238.322	972.214	1,035.872	862.604	1,182.460
	15	2,515.808	1,956.781	2,107.198	1,782.431	2,358.434
	16	5,098.459	3,886.246	4,292.998	3,602.747	4,837.115
	17	10,483.717	7,779.587	8,840.097	7,364.237	9,985.172
	18	21,440.172	15,299.374	17,982.026	15,001.009	20,318.424
	19	43,487.275	30,240.065	36,198.347	30,063.243	41,198.996
	20	87,392.343	59,127.952	71,064.345	57,422.867	82,092.395

quickly in function of the depth of a descendant node with respect to v. In particular, the LagLB is already poorer than any of the previous bounds for the children of a generic node v of the search tree. However, if the computation of the LPLB and of the LagLB is alternated during the search tree then it is possible to reach a good trade off in terms of tightness of the lower bound and solution times.

5. Upper bounds on the optimal solution to the BMEP

A possible approach to obtain an upper bound on the optimal solution to the BMEP consists of using Hendy and Penny (1982)' SAS in a greedy fashion. The basic idea consists of evaluating first all of the possible insertions of a taxon into a given sub-phylogeny and subsequently selecting the insertion that results minimal with respect to a given criterion $C(D,T)$ (see Algorithm 2). An alternative to the greedy SAS is the *agglomerative approach* (Gascuel, 2005), i.e., an algorithm that starts from an infeasible solution to the BMEP (e.g., a star tree with n terminals (taxa)) and iteratively cluster taxa according to a specific optimality criterion, until a phylogeny of Γ is obtained. Algorithm 3 formalizes the idea at the core of an agglomerative approach. The algorithm starts with the star tree and inserts a new internal edge (u_1,u_2), by replacing the internal vertex u. Such an edge is determined by minimizing a selection criterion on all bipartitions $P=(N_1,N_2)$ of the neighborhood of u. Then, the neighborhood of u_1 and u_2 are set to be $N_1\cup\{u_2\}$ and $N_2\cup\{u_1\}$, respectively. The algorithm iterates this process until obtaining a phylogeny of Γ, i.e., until agglomerating $(2n-3)$ edges. Fig. 6 shows two consecutive steps of the algorithm. In particular, starting from a star tree with eight taxa, a new tree can be obtained by inserting an edge between the star tree on taxa $\{1,2,3,6,7,8\}$ and the pair $\{4,5\}$. The subsequent step may consist, e.g., of inserting an internal edge between the subtrees on taxa $\{1,7,8\}$ and $\{2,3,4,5,6\}$. Observe that both Algorithms 2 and 3 are correct greedy algorithms for the BMEP. Given a sub-phylogeny of $S\subseteq\Gamma$, a possible greedy selection criterion for Algorithm 2 consists of minimizing

$$C(D,T)=L(T_S)+\sum_{t\in\Gamma\setminus S}\min_{\substack{i,j\in S\\ i<j<t}}\frac{1}{2}\left(d_{it}+d_{jt}-d_{ij}\right)$$

already introduced by Pardi (2009). A possible greedy selection criterion for Algorithm 3 consists of picking a vertex u according to the bipartition $P=(N_1,N_2)$, $|N_1|>|N_2|=2$, which minimizes the change in Semple and Steel's extension (Semple and Steel, 2004) of length function (1) for unrooted non-binary trees. This method is known in the literature as the *Neighbor-Joining algorithm* (NJ) (Saitou and Nei, 1987; Studier and Keppler, 1988; Gascuel, 1994; Gascuel and Steel, 2006). After having minimized the greedy selection criterion, we carry on a local search based on the *Nearest Neighbor Interchange* (NNI) described in Allen and Steel (2001) to improve the quality of the upper bound provided by both algorithms. For the sake of space, we refer the reader interested in a comprehensive discussion on NNI to Allen and Steel (2001). Here it suffices to say that this local search swaps subtrees

Algorithm 2: Stepwise addition strategy - Greedy search

Input: Ordered set of taxa $\Gamma=\{1,\dots,n\}$, distance matrix D for Γ
Output: A phylogeny of Γ

```
t ← 3;
T ← The only sub-phylogeny of Γ for {1, 2, 3};
while t < n do
    t ← t + 1;
    ê ← argmin{C(D, T ⊕_e t) : e ∈ E(T)};
    T ← T ⊕_ê t;
return T;
```

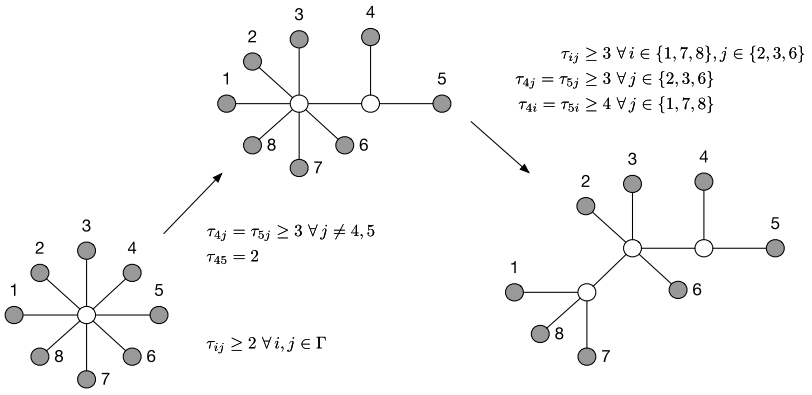


Fig. 6. An example of two steps of the agglomerative approach in the case of eight taxa.

adjacent to a given internal edge of a partial phylogeny until no further improvement is possible.

6. A massively parallel implicit enumeration algorithm

The exploration strategy at the core of Hendy and Penny's SAS (Hendy and Penny, 1982) is naturally prone to parallelization. Indeed, the solution subspaces generated by the SAS through new insertions on a given sub-phylogeny constitute a partition of the original solution subspace of the BMEP and can therefore be explored independently from one another. This is the case, e.g., for the subspaces constituted by the phylogenies of Γ derived from the respective three sub-phylogenies obtained by inserting the fourth taxon on the initial star tree in Fig. 5. Strangely enough, however, this particular feature of Hendy and Penny's SAS (Hendy and Penny, 1982) has never been exploited in the literature on the BMEP to speed up the exact resolution of its instances. In this section, we close this gap, by studying possible parallelization paradigms that may prove particularly effective to this end.

We start by observing that the number of computing cores available in a shared memory environment is usually even. In contrast, the number of sub-phylogenies for a fixed subset of taxa $S \subseteq \Gamma$ is odd, precisely equal to $(2|S| - 5)!!$, and exponentially growing in function of the cardinality of S. At best the number of cores and initial sub-phylogenies differ only by one but mismatches for the number of subsequently constructed sub-phylogenies makes inefficient any parallelization scheme of the implicit enumeration search that consists of just delegating the exploration of a sub-phylogeny (hence of a subspace) to the available computing cores. Moreover, because entire subspaces could be pruned during the implicit enumeration search, in such parallelization schemes some computing cores could go idle by causing so a useless waste of computing resources. An efficient parallelization scheme, therefore, must include both a scalable exploration strategy of the solution space and a dynamic balance of the workload to avoid as much as possible the absence of cores in idle status. In order to design such a scheme, we first define a *job* as a triplet (Γ, T_S, F) such that T_S is a sub-phylogeny of Γ and $F \subseteq E(T_S)$ a subset of edges in T_S. Moreover, we also introduce a *queue of jobs* Q whose length may be dynamically changed in function of the available computing cores that are in the idle state. Then, a possible parallel exploration strategy of the solution space of the BMEP can be outlined as in Algorithm 4. Specifically, Algorithm 4 first performs an initialization phase, by allocating shared data structures for all the available cores in the system. Shared data contain, from among others, information about the branches processed by a core and information related to the status of a core (e.g., idle or not). Each core can read and modify the records relative to its own identifier. All of the other cores instead can just read such information, independently from one another and in an asynchronous way, by simplifying so the implementation of the stopping criterion of Algorithm 4. The initialization phase terminates with the following steps: the set Γ is ordered and the best upper bound on the optimal solution to the BMEP is calculated and stored as described in Section 5. Algorithm 4 then constructs an initial sub-phylogeny T_S for the first 3 taxa in Γ, it computes and stores the best lower bound for T_S (see Section 3), and it finally creates a queue of jobs Q, by considering all of the possible insertions of the fourth taxon on T_S. Finally, Algorithm 4 runs the parallel exploration of the subspaces of BMEP solutions derived by all of the possible insertions of the remaining taxa on the sub-phylogenies in Q and stops once that Q is empty. Fig. 7 provides a high-level view of the exploration strategy implemented in Algorithm 4. We remark here that the functions Push and Pop are classical queue operators. Specifically, Push$(\Gamma, T, E(T))$ adds a job in the queue Q and Pop(Q) removes and returns the first element J of Q, respectively. Finally, the function SAS$(D, \Gamma, T, F, T^*, Q)$ implements Algorithm 5.

Algorithm 3: Agglomerative approach - Greedy search

Input: Set of taxa $\Gamma = \{t_1, \ldots, t_q\}$, distance matrix D for Γ,
Output: A phylogeny of Γ

1 $V \leftarrow \Gamma \cup \{v_1\}$;
2 $E \leftarrow \{\{t_i, v_1\} \; : \; t_i \in \Gamma\}$;
3 **while** $|E| < (2n - 3)$ **do**
4 $u \leftarrow$ A vertex of degree at least four in (V, E) that minimizes a selection criterion $C(D, P)$ for a bipartition $P = (N_1, N_2)$ of its neighbors;
5 $V \leftarrow V \setminus \{u\} \cup \{u_1, u_2\}$;
6 $E \leftarrow E \setminus \{(u, v) \; : \; (u, v) \in E\} \cup \{(u_1, v) \; : \; v \in N_1\} \cup \{(u_2, v) \; : \; v \in N_2\} \cup \{\{u_1, u_2\}\}$;
7 **return** (V, E);

The global search strategy starts at lines 9 and 10. Note that at the beginning and at the end of the while loop, all cores need to be synchronized to ensure both a proper data initialization and a report of the results. This is indicated by the dashed arrows in Fig. 7. Observe also that the stopping criterion at line 9 of Algorithm 4 can be slightly changed to account for a time-based stopping criterion (e.g., the computing time exceeded the maximum running time allowed). A proper parallel implementation of Algorithm 4 involves determining how jobs are retrieved from Q and how they are processed and added to Q. These steps are written in bold in Fig. 7. Retrieving and deleting a job from the queue Q can be trivially implemented by semaphores, i.e., by making

Algorithm 4: Stepwise addition strategy - Parallel initialization and termination

Input: Distance matrix D for Γ, queue Q of jobs, ordered set of taxa $\Gamma = \{1, \ldots, n\}$
Output: A phylogeny T^* of Γ

1 Allocate and initialize shared memory;
2 $T^* \leftarrow$ the best upper bound;
3 $T_S \leftarrow$ the only sub-phylogeny of Γ for $S = \{1, 2, 3\}$;
4 Calculate and store the best lower bound for T_S;
5 **for** $e \in E(T_S)$ **do**
6 $T \leftarrow T_S \oplus_e 4$;
7 Push $(\Gamma, T, E(T))$ into Q;
8 **while** Q *not empty* **do**
9 Execute in parallel $(\Gamma, T, F) = \text{Pop}(Q)$ and SAS$(D, \Gamma, T, F, T^*, Q)$ on an idle core;
10 **return** T^*;

Algorithm 5: Stepwise addition strategy - Implicit enumeration of phylogenies on one core (SAS)

Input: Distance matrix D for Γ, the set of taxa Γ, a sub-phylogeny T_S of Γ, edge set F with $F \subseteq E(T_S)$, phylogeny T^* of Γ, queue Q

1 $t \leftarrow |S| + 1$;
2 **for** $e \in F$ **do**
3 $T \leftarrow T_S \oplus_e t$;
4 Calculate $L(T)$;
5 Update time limit information in the shared memory;
6 **if** $S = \Gamma$ ***and*** $L(T) < L(T^*)$ **then**
7 $T^* \leftarrow T_S$;
8 **else**
9 continue $\leftarrow$ FALSE;
10 **if** $LagLB(T_S) < L(T^*)$ **then**
11 **if** $LPLB(T_S) < L(T^*)$ **then**
12 continue $\leftarrow$ TRUE;
13 **if** *continue* **then**
14 **if** *There exist idle cores* ***or*** *queue Q contains less jobs than there are overall cores* **then**
15 Partition $E(T) = F_1 \cup F_2$ such that $||F_1| - |F_2|| \leq 1$;
16 inserted $\leftarrow$ FALSE;
17 **while** Q *not full* ***and*** *time limit not exceeded* ***and*** *not inserted* **do**
18 inserted $\leftarrow$ Push (Γ, T, F_2) into Q;
19 SAS$(D, \Gamma, T, F, T^*, Q)$;

sure that access to the queue is given to exactly one core at a time. The processing and adding of jobs to Q require instead a bit more attention. Both operations are implied in Fig. 7 by the statement *Run a SAS on the local data* and executed by SAS(D, Γ, T, F, T^*) in Algorithm 4. This line calls for the execution of the steps outlined in Algorithm 5.

The input of Algorithm 5 includes the distance matrix **D** and the job (Γ, T_S, F). For every newly enumerated sub-phylogeny T, line 4 of Algorithm 5 computes the objective function value of the BMEP. Subsequently, line 5 of the algorithm checks if the time limit has been exceeded. This check is necessary because Algorithm 5 is employed by a parallel algorithm but the for-loop at line 2 runs independently from any action taken by other cores. Then, line 6 checks if a new best-so-far solution to the BMEP has been found and in the positive case stores it in T^*. This updating step must be carried out using semaphores to ensure consistency and avoid overwriting. Next, lines 9 to 12 introduce a bounding procedure for all sub-phylogenies T_S with $S \subset \Gamma$. Specifically, the function LagLB(T_S) at line 10 computes LagLB and is initially called to solve Formulation 4 for the considered subphylogeny T_S (see Section 3). In the case no pruning is possible, then Algorithm 5 computes the more expensive lower bound LPLB for the same subphylogeny, by calling the function LPLB(T_S) at line 11. If LPLB(T_S) can be improved without exceeding the intermediate optimal objective function value $L(T^*)$, then Algorithm 5 continues with our SAS because a series of branchings of T_S might produce an optimal solution to the BMEP. Line 21 calls SAS$(D, \Gamma, T, F, T^*, Q)$ after the bounding procedure at lines 9 to 12 to complete our extended SAS for the sub-phylogeny T_S and to ensure that our parallel algorithm in Fig. 7 terminates after reporting an optimal solution to the BMEP. However, to decrease the overall running time, lines 14 to 19 seek to rebalance the workload for every core by continuously monitoring the length of the queue Q. In particular, if Q is already very long and all other cores are running Algorithm 5, adding new jobs to the queue Q may prove pointless. Instead, if the length of Q is short or there are idle cores, then the adding of at least one new job in Q may prove beneficial. In such case, the algorithm partitions $E(T)$ into two sets F_1 and F_2 of similar size, by giving rise to two new jobs, say (Γ, T, F_1) and (Γ, T, F_2), one of which is processed by the current core and the other is added to the queue Q.

7. Computational experiments

In the previous section, we saw that the parallel implicit enumeration algorithm requires the specification of: the order in which the taxa in Γ are processed, the order in which edges in F are processed at line 2 of Algorithm 5, and the order in which the jobs in the queue Q are extracted and processed. In this section, we consider and compare alternative implementation settings for these orders on benchmark sets of biological and artificial instances of the BMEP provided by Catanzaro et al. (2006), Desper and Gascuel (2002) and Gascuel (2005).

We considered three possible settings to select a processing order of the taxa in Γ, namely: (i) the order corresponding to the rows of the input distance matrix D (hereafter, this order is denoted as None); (ii) the order provided by shortest Hamiltonian tour obtained when solving the Traveling Salesman Problem (TSP) on the instance encoded by D; and (iii) the order obtained by shortcutting the tree returned by Algorithms 2 and 3, respectively. Concerning the set F of edges of a sub-phylogeny, we considered the case in which F is (i) ordered in increasing way with respect to the initial star-tree or (ii) ordered in a non-decreasing way by pre-calculating the lower bound one would obtain in the next recursion step by branching on an edge (hereafter, this order is denoted as NDLB). Finally, concerning the order in which the jobs in Q are extracted and processed, we considered the case in which Q is ordered *Non-Increasing in the Number of Edges* (NINE); the case in which Q is ordered *Non-Decreasing in the Number of Edges* (NDNE); and the classic cases in which the jobs are extracted in *First-In First Out* (FIFO) and *First-In Last-Out* (FILO), respectively. By combining these implementation settings we derived 32 possible implementations of the parallel implicit enumeration algorithm described in the previous section, each of which has been denoted as PSx, $x \in [1, 32]$, and specified in Table 3.

7.1. Implementation details

All the implementations described in this section have been coded in ANSI C++, by relying on FICO Xpress Optimizer libraries v33.01.05 for linear programming and on OpenMP (Chandra et al., 2001) version 3.0 for parallelism. The codes have been compiled by the GGC compiler version 4.8.5, libc version 2.17, and run on a 2 × 8 Core E5-2667v3 Processor at 3.2 GHz and 256 GB RAM, operating system CentOS Linux 7 release 7.9.2009 (kernel linux 3.10.0-1160.71.1.e17.x86_64).

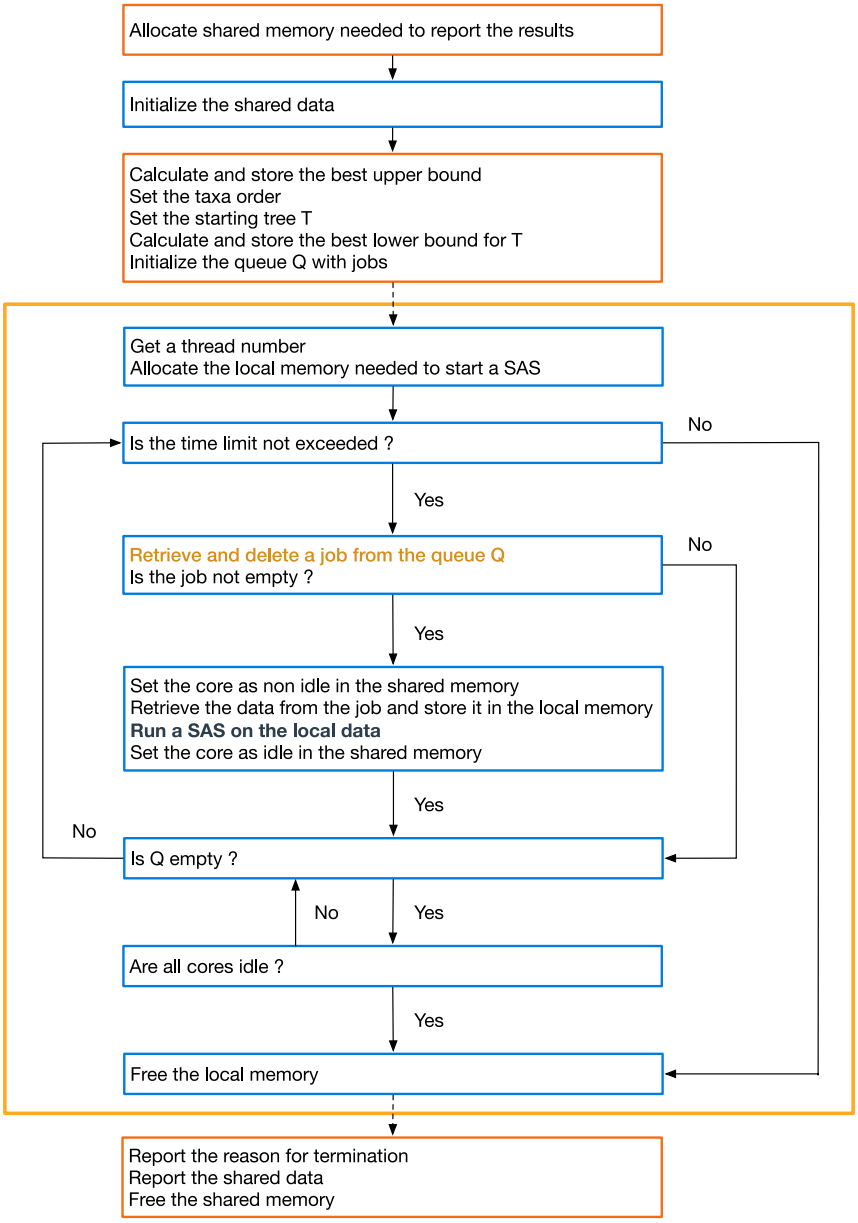


Fig. 7. A description of Algorithm 4 in a shared memory system. We are given a set of cores that can access a shared memory and that can run in parallel independently. Statements inside red and blue boxes are processed by a fixed master core and every core, respectively. The yellow box contains all statements that are part of the global search strategy indicated at lines 8 and 9 of Algorithm 4.

We assumed one hour as maximum runtime per instance and rescaled the objective function by a factor 2^n to reduce possible numerical stability problems. Codes and data used in the experiments can be downloaded at https://github.com/mfrohn/BMEPparallel.

7.2. Implementation settings

The systematic analysis of the computational results relative to the 32 different implementation settings considered in this section exceeds the page limits of the journal and has been therefore omitted. The interested reader, however, can find it online at https://github.com/mfrohn/BMEPparallel (in particular, in Figures 1 to 5 of the file "Supplementary Material"). In this section, we just limit to summarize the most important findings arising from these massive computational experiments.

We first observed that the choice of the processing order of jobs in Q has an impact of multiple magnitudes smaller than both the choice for the processing order of the taxa in Γ and the processing order of F. Moreover, no queue order yields a statistically significant advantage compared to all others. We also observed that ordering F in increasing fashion leads overall to better runtimes compared to the NDLB order. Specifically, we observed that the worst case and 75% quartile runtimes improve dramatically for increasing n. This leads to a twofold conclusion: on the one hand, the additional computational overhead for the calculation of lower bounds required at each iteration to determine the order of F affects negatively the runtime for small n. On the other hand, the calculation of lower bounds throughout the tested algorithms dominates all other contributions to the total computation time. Preprocessing the lower bounds for future iterations does not scale well with a higher number of cores and increasing n

Table 3
Settings for the algorithm PSx: the taxa in Γ may be ordered according to (i) the input order of the rows of the distance matrix D (denoted as "None"); (ii) the order provided by shortest Hamiltonian tour obtained when solving the Traveling Salesman Problem (TSP) on the instance encoded by D; and (iii) the order obtained by shortcutting the tree returned by Algorithms 2 and 3, respectively. The set F of edges of a sub-phylogeny can be either (i) ordered in increasing way with respect to the initial star-tree or (ii) non-decreasing ordered by pre-calculating the lower bound one would obtain in the next recursion step by branching on an edge (NDLB). The order of Q can be *Non-Increasing in the Number of Edges* (NINE) or *Non-Decreasing in the Number of Edges* (NDNE). FIFO and FILO stand for the usual *First-In First Out* and *Firs-In Last-Out* strategies, respectively.

Algorithm	Order of Γ	Order of F	Order of Q	Algorithm	Order of Γ	Order of F	Order of Q
PS01	None	Increasing	FIFO	PS17	Algorithm 2	Increasing	FIFO
PS02	None	Increasing	FILO	PS18	Algorithm 2	Increasing	FILO
PS03	None	Increasing	NINE	PS19	Algorithm 2	Increasing	NINE
PS04	None	Increasing	NDNE	PS20	Algorithm 2	Increasing	NDNE
PS05	None	NDLB	FIFO	PS21	Algorithm 2	NDLB	FIFO
PS06	None	NDLB	FILO	PS22	Algorithm 2	NDLB	FILO
PS07	None	NDLB	NINE	PS23	Algorithm 2	NDLB	NINE
PS08	None	NDLB	NDNE	PS24	Algorithm 2	NDLB	NDNE
PS09	TSP	Increasing	FIFO	PS25	Algorithm 3	Increasing	FIFO
PS10	TSP	Increasing	FILO	PS26	Algorithm 3	Increasing	FILO
PS11	TSP	Increasing	NINE	PS27	Algorithm 3	Increasing	NINE
PS12	TSP	Increasing	NDNE	PS28	Algorithm 3	Increasing	NDNE
PS13	TSP	NDLB	FIFO	PS29	Algorithm 3	NDLB	FIFO
PS14	TSP	NDLB	FILO	PS30	Algorithm 3	NDLB	FILO
PS15	TSP	NDLB	NINE	PS31	Algorithm 3	NDLB	NINE
PS16	TSP	NDLB	NDNE	PS32	Algorithm 3	NDLB	NDNE

because when a precalculated lower bound is employed by a core the corresponding node of the search-tree might be already pruned by a different core. The computational results relative to the processing order of the taxa in Γ lead instead to less net conclusions. The best worst-case performance is always reached by the order derived from Algorithm 3, but the lowest best-case and 25% quartile runtime is achieved by other orders of Γ for most n. However, since the order derived from Algorithm 3 performs statistically significant better than all other configurations for $n = 20$ we decided to keep this order as the best choice for further experiments. Overall, the analysis indicates that the configurations PS25-PS28 of Table 3 perform the best.

We also considered further modifications of lines 14 to 19 in Algorithm 5 to strengthen the performance of configurations PS25 to PS28. First, we investigate the number p of sets of the partition $E(T) = \bigcup_{i=1}^{p} F_i$ with default value $p = 2$ in Algorithm 5. To this end, we consider configurations PS25 to PS28 for $p \in \{3, \dots, 9\}$. We keep the assumption of similar sized subsets at line 15, i.e., $\left||F_i| - |F_j|\right| \leq 1$ for all $i, j = 1, \dots, p$. Then, we consider the modifications of configurations PS25 to PS28 listed in the upper half of Table 4.

The computational results for configurations PS33 to PS60 of Table 4 are shown in the Supplementary Material. Here we can observe that no choice of the parameter p yields a statistically significant advantage compared to all other evaluated choices. However, for sufficiently large n, increasing the number of partitions sets p seems to slow down the execution of the tested algorithms. This observation is consistent with the idea that sufficiently small partition sets lead to an overabundance of jobs in the queue. This increase in computation time for managing the distribution of jobs can not be compensated by the decrease in idle time for each core. Hence, we only consider $p = 2$ and $p = 3$ as part of the best performing configurations in the worst case.

As a second extension of the partitioning process of the edge set F in Algorithm 5 we introduce a positive constant integer c to define the minimum cardinality

$$c^*(p) = \min\left\{\max\left\{c, \left\lfloor \frac{|E(T)|}{p} \right\rfloor\right\}, |E(T)|\right\}$$

of partition sets F_i, $i = 1, \dots, p$. Employing $c^*(p)$ as a lower bound on the number of edges included in one job ensures that the balancing of workloads can be dynamically tailored to the size of sub-phylogenies. Overall, any partition of edges $E(T)$ aims at making at least as many jobs available in the queue as there are cores in the shared memory system. The computational results for configurations PS25 to PS28, PS33 to PS36, and PS61 to PS84 of Table 4 are shown in the Supplementary Material. Again, we observe that no specification is significantly the best among all tested configurations. Smaller choices for the constant c seem to be preferable for increasing n.

7.3. Comparing PS76 versus the state-of-the-art

The implementation setting PS76 offers the best worst case performance among all analyzed configurations PS01 to PS84 for $n = 20$. Tables 5 and 6 summarize the numerical results with respect to the solution time and the number of branches taken by configuration PS76 to solve Catanzaro et al.'s benchmark biological instances of the BMEP (Catanzaro et al., 2012). In general, we can see that the multi-core version solves the BMEP instances faster than Catanzaro et al.'s state-of-the-art solution algorithm (Catanzaro et al., 2012). Moreover, both tables show that using a single core can only beneficial when dealing with small sets of taxa. This is explained by the fact that the computational overhead of managing multiple cores for configuration PS76 on a few taxa is more expensive than the potential for improvements in the solution time through parallelization. Furthermore, we observe that configuration PS76 is twice as fast on multiple cores as on a single core 88% of the time. The relative improvements when comparing configuration PS76 on multiple cores and a single core also show that the parallel version is at least one order of magnitude faster than the single-core version 23% of the time. This is particularly interesting for instances like *M82* on 20 taxa for which the BMEP was not solvable before within one hour. Moreover, the clear difference in runtime between the two solvers is illustrated in Fig. 8. Thus, configuration PS76 increases the number of taxa for which the BMEP can be solved in a reasonable amount of time.

7.3.1. Comparing PS76 versus FastME

A computational experiment that may attract the attention of the biological community concerns how far the solutions provided by popular heuristics for the BMEP, such as FastME (Lefort et al., 2015), can be with respect to the optimum. To address this issue we considered the sets of molecular sequences contained in FastME's supplementary material (Lefort et al., 2015) (see http://www.atgc-montpellier.fr/fastme/paper.php). We configured FastME to start from an initial phylogeny constructed by the BIONJ algorithm (see Gascuel, 1997) and then perform local searches on it based on NNI and SPR interchanges (see Allen and Steel, 2001). We then carried out the following steps to use the solution provided by FastME as the initial upper bound on the optimal solution to the BMEP in configuration PS76.

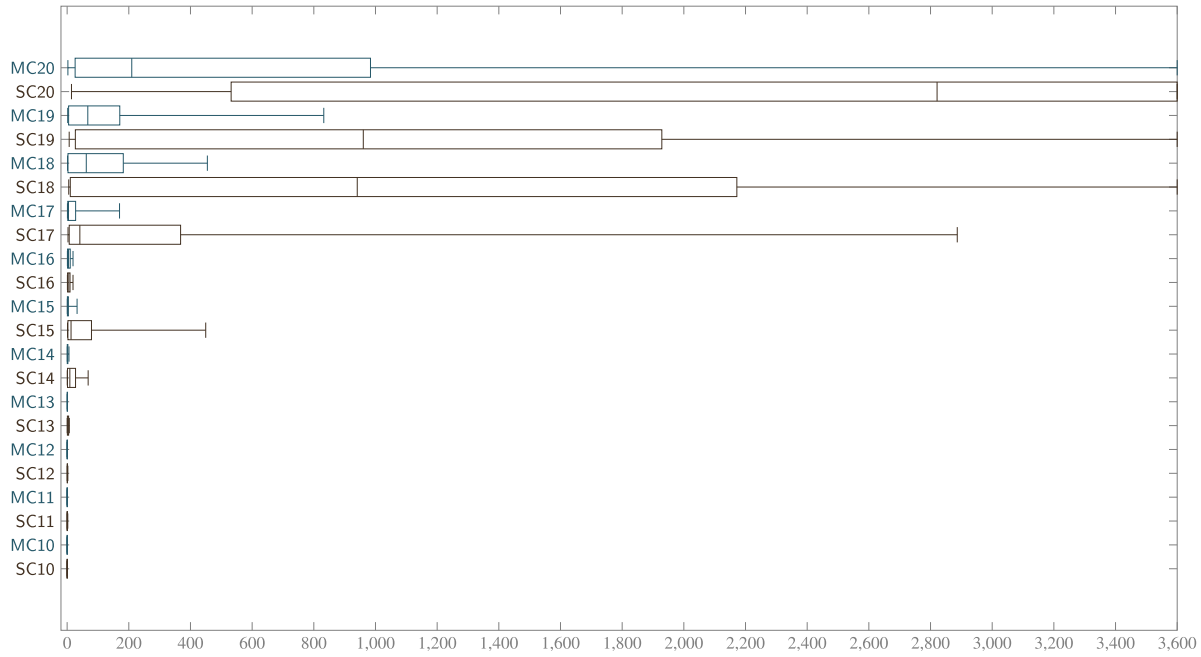


Fig. 8. A boxplot of the runtime in seconds for the algorithm PS76 on a single core and on 32 cores, e.g., SC10 is the runtime of the single-core version of PS76 on 10 taxa and MC17 is the runtime of the multi-core version of PS76 on 17 taxa. The plots are only shown for at most 3,600 s of runtime to improve the readability.

Table 4
Settings for the algorithm PSx when assuming that the order of Γ derives from Algorithm 3 and the order of F is increasing with respect to the initial star-tree. FIFO and FILO stand for the usual *First-In First Out* and *First-In Last-Out* strategies, respectively.

Algorithm	Order of Q	Value of p	Value of c	Algorithm	Order of Q	Value of p	Value of c
PS33	FIFO	3	–	PS47	NINE	6	–
PS34	FILO	3	–	PS48	NDNE	6	–
PS35	NINE	3	–	PS49	FIFO	7	–
PS36	NDNE	3	–	PS50	FILO	7	–
PS37	FIFO	4	–	PS51	NINE	7	–
PS38	FILO	4	–	PS52	NDNE	7	–
PS39	NINE	4	–	PS53	FIFO	8	–
PS40	NDNE	4	–	PS54	FILO	8	–
PS41	FIFO	5	–	PS55	NINE	8	–
PS42	FILO	5	–	PS56	NDNE	8	–
PS43	NINE	5	–	PS57	FIFO	9	–
PS44	NDNE	5	–	PS58	FILO	9	–
PS45	FIFO	6	–	PS59	NINE	9	–
PS46	FILO	6	–	PS60	NDNE	9	–
PS61	FIFO	2	2	PS73	FIFO	3	2
PS62	FILO	2	2	PS74	FILO	3	2
PS63	NINE	2	2	PS75	NINE	3	2
PS64	NDNE	2	2	PS76	NDNE	3	2
PS65	FIFO	2	3	PS77	FIFO	3	3
PS66	FILO	2	3	PS78	FILO	3	3
PS67	NINE	2	3	PS79	NINE	3	3
PS68	NDNE	2	3	PS80	NDNE	3	3
PS69	FIFO	2	4	PS81	FIFO	3	4
PS70	FILO	2	4	PS82	FILO	3	4
PS71	NINE	2	4	PS83	NINE	3	4
PS72	NDNE	2	4	PS84	NDNE	3	4

1. For each set of molecular sequences, we calculated the default distance matrix, i.e., the model introduced in Felsenstein (1984) for DNA data and in Le and Gascuel (2008) for protein data, respectively.
2. For each distance matrix, we constructed an $n \times n$ submatrix including the n *most dissimilar taxa*, i.e., the first n taxa in the order of non-increasing phylogenetic diversity (Steel, 2005).
3. For each distance matrix on the n most dissimilar taxa
   (i) we run FastME to calculate a phylogeny;
   (ii) we run configuration PS76 with two modifications: select the phylogeny produced by FastME as the initial upper bound on the optimal solution to the BMEP and insert taxa in non-increasing phylogenetic diversity order (Steel, 2005). We call this configuration PS*.

The results of this procedure for $n = 20$ are shown in Table 7. The experiments show that FastME solved to optimality 12 instances of the BMEP; nothing can be said about the optimality of the solution found for the remaining 7 instances as PS* was unable to terminate within the given time limit (one hour of computing time). Because practical data sets often include much more than 20 taxa, it is natural to wonder how far configuration PS* can go within a given time limit. Table 8 addresses this issue. In particular, the table reports on the biggest and

Table 5
Numerical results obtained for the algorithm PS76 on a single core and on 32 cores. The last column shows the improvement in the solution time of the multi-core approach relative to the time required to terminate PS76 on one core. The improvement measures the difference in solution time between the single-core and multi-core version. Bold indicates the approach that performed the best.

				Single-core		Multi-core		
Data set	Taxa	Optimum	Gap (%)	Time (s)	Branches	Time (s)	Branches	Imp.
Primates12	10	124.968	2.57	**0.073**	216	0.191	216	−0.118
	11	332.834	2.59	**0.082**	344	0.196	344	−0.114
	12	802.589	2.47	**0.169**	1,720	0.173	579	−0.004
M17	10	105.134	1.05	**0.082**	472	0.104	609	−0.022
	11	261.233	1.18	**0.174**	999	0.182	1,037	−0.008
	12	541.632	1.21	0.570	3,296	**0.249**	3,574	0.321
	13	1,181.598	1.65	3.283	18,479	**0.622**	30,738	2.661
	14	2,408.066	1.65	1.216	8,721	**0.534**	19,548	0.682
	15	4,998.295	1.65	5.311	33,737	**0.892**	40,605	4.419
	16	10,225.560	1.70	5.511	295,509	**4.011**	205,280	1.500
	17	20,788.112	1.64	8.034	36,857	**1.711**	74,402	6.323
M18	10	190.331	2.11	0.154	909	**0.141**	1,136	0.013
	11	396.942	2.57	0.550	5,392	**0.247**	5,909	0.303
	12	805.937	2.71	0.637	4,383	**0.259**	5,187	0.378
	13	1,758.111	3.27	4.888	36,317	**1.006**	49,614	3.882
	14	3,599.678	3.62	18.704	125,553	**1.297**	80,173	17.407
	15	7,746.218	3.47	115.178	664,734	**4.573**	300,169	110.605
	16	15,973.016	3.51	**15.133**	1,129,882	19.102	1,354,570	−3.969
	17	32,345.662	3.53	228.149	919,580	**17.537**	1,039,192	210.612
	18	66,029.112	3.55	1,152.986	4,176,462	**68.131**	3,716,006	1,084.855
SeedPlant25	10	91.458	5.51	0.207	1,904	**0.172**	2,142	0.035
	11	206.250	5.10	0.389	3,880	**0.234**	4,098	0.155
	12	427.816	4.87	0.506	3,038	**0.294**	3,343	0.212
	13	943.774	5.30	4.809	30,295	**0.760**	34,960	4.049
	14	1,929.219	5.06	16.181	92,293	**1.656**	99,754	14.525
	15	4,085.763	4.66	20.043	101,135	**2.533**	152,207	17.510
	16	8,353.457	5.76	7.422	397,996	**7.087**	385,206	0.335
	17	17,314.429	6.21	74.456	237,259	**6.565**	292,273	67.891
	18	36,156.527	6.92	> 3,600	12,817,828	**454.788**	24,714,319	n.a.
	19	75,261.323	6.97	> 3,600	12,555,016	**832.327**	44,306,981	n.a.
	20	160,913.646	9.11	> 3,600	10,899,618	> 3,600	140,412,968	n.a.
M43	10	105.020	1.29	**0.105**	433	0.173	462	−0.068
	11	209.282	2.39	0.354	2,321	**0.183**	2,539	0.171
	12	434.326	1.79	0.330	1,577	**0.237**	1,952	0.093
	13	895.424	1.74	0.906	3,339	**0.507**	10,135	0.399
	14	1,808.970	1.75	1.467	6,317	**0.688**	14,598	0.779
	15	3,965.234	1.53	2.603	12,742	**1.064**	40,716	1.539
	16	8,219.835	1.51	1.369	33,584	**1.368**	36,137	0.001
	17	16,798.759	1.82	5.978	18,950	**2.144**	59,408	3.834
	18	35,383.158	1.44	5.819	20,880	**2.344**	64,230	3.475
	19	71,769.903	1.43	6.678	21,233	**3.061**	80,764	3.617
	20	157,472.424	1.48	28.296	80,160	**7.697**	216,567	20.599

smallest value of n for which configuration PS* does terminate and does not terminate within one hour of computing time, respectively. Interestingly, this analysis shows that FastME was unable to optimally solve instances from the data sets proteic_M3755_77x9918_2008-BMGE and nucleic_M2573_346x897_2006.

In order to get a better view of the average performance of configuration PS* in comparison to FastME we also looked at simulations based on random trees with parameter values chosen in order to cover some features of real data sets. Specifically, we considered a 24-taxa tree set of size 2,000 that was generated at moderate evolutionary rates (see Desper and Gascuel, 2002) for detailed specifications). Table 9 shows the results obtained when considering these instances. We observe that configuration PS* can not solve 19.90% of the instances within one hour and FastME can not solve 13.25% of the instances up to optimality, respectively. Again, a certificate of optimality for FastME is derived from configuration PS* only if configuration PS* terminates (within one hour). The FastME solutions are obtained in at most one second. Moreover, configuration PS* proves that at least 67.80% of the solutions produced by FastME are optimal. Thus, we can find a significant set of inputs for which FastME can be improved. However, identifying this set can not be done quickly so far due to the average high running time of configuration PS*. The number of taxa of the input instances considered in our experiments for which we can solve the BMEP up to optimality is limited by the performance of the parallel exact solution algorithms we have discussed. However, as we show in the next section, the range of values for the estimated evolutionary distances d_{ij}, $i, j \in \Gamma$, also plays a crucial role in being able to prove optimality or not.

8. Concluding remarks

The recent COVID19 pandemic highlighted the need to equip epidemiologists and, more in general, medicine and life sciences researchers and practitioners with estimation models and algorithms able to track the evolution of pathogens and, more in general, of taxa in the most reliable and accurate way possible (Catanzaro et al., 2022). The optimal solution to the BMEP may constitute an answer to these needs, thanks to its particular mathematical and statistical properties (see Catanzaro et al., 2022; Gascuel, 2005; Gascuel and Steel, 2006). Developing models and algorithms able to exactly solve instances of the BMEP as large as possible is, therefore, highly desirable in practical applications. Despite the recent advances in the polyhedral combinatorics of the BMEP (Catanzaro et al., 2012; Semple and Steel, 2004; Cueto and Matsen, 2011; Haws et al., 2011; Catanzaro et al., 2020b; Forcey et al., 2015, 2017), however, the scientific community still struggles to

Table 6
Continuation of Table 5.

				Single-core		Multi-core		
Data set	Taxa	Optimum	Gap (%)	Time (s)	Branches	Time (s)	Branches	Imp.
	10	152.482	1.79	0.193	1,066	**0.126**	1,309	0.067
	11	328.645	1.61	0.219	1,221	**0.172**	1,817	0.047
	12	685.972	2.20	1.214	6,434	**0.275**	5,340	0.939
	13	1,502.870	2.33	3.251	19,092	**0.649**	23,228	2.602
	14	3,094.888	2.27	52.929	320,526	**5.985**	402,286	46.944
RbcL55	15	6,448.259	2.46	66.892	293,413	**3.829**	216,056	63.063
	16	13,455.292	2.50	5.235	306,146	**5.007**	290,968	0.228
	17	27,804.246	2.94	787.426	3,413,959	**57.475**	3,630,747	729.951
	18	56,175.236	2.90	3,190.887	11,874,059	**296.129**	15,559,660	2,894.758
	19	114,623.865	2.97	1,840.835	6,186,132	**186.286**	8,492,081	1,654.549
	20	234,189.871	4.63	> 3,600	11,970,044	**1,198.891**	45,427,832	n.a.
	10	163.876	1.07	**0.079**	245	0.136	366	−0.057
	11	350.425	1.50	0.239	1,198	**0.188**	1,460	0.051
	12	753.821	2.06	2.342	18,078	**0.518**	19,168	1.824
	13	1,580.764	2.21	**0.222**	774	0.331	5,868	−0.109
	14	3,345.564	2.35	**0.422**	1,966	0.458	8,902	−0.036
M62	15	7,161.078	2.64	2.184	11,389	**0.773**	27,122	1.411
	16	14,980.693	2.54	1.442	61,985	**1.322**	57,958	0.120
	17	31,293.082	2.50	6.991	31,200	**2.251**	102,974	4.740
	18	66,187.974	2.14	14.649	63,153	**3.043**	140,898	11.606
	19	145,642.424	2.19	80.066	306,081	**9.918**	518,041	70.148
	20	296,651.044	2.81	2,042.545	9,494,687	**80.650**	5,413,747	1,961.895
	10	41.223	5.74	0.304	1,592	**0.204**	1,609	0.100
	11	87.162	4.94	0.423	1,333	**0.176**	1,322	0.247
	12	183.042	4.64	0.452	1,499	**0.234**	1,575	0.218
	13	382.700	5.39	0.666	2,367	**0.304**	2,788	0.362
	14	773.810	5.60	0.847	2,690	**0.444**	3,040	0.403
Rana64	15	1,603.120	6.47	1.265	4,066	**0.554**	7,756	0.711
	16	3,290.745	7.78	0.888	14,485	**0.886**	15,684	0.002
	17	6,745.447	8.01	2.883	9,554	**1.021**	18,071	1.862
	18	15,435.268	5.07	5.188	16,797	**1.390**	28,289	3.798
	19	36,052.455	4.48	8.875	27,505	**1.825**	40,105	7.050
	20	81,105.131	4.74	14.097	39,201	**2.483**	61,282	11.614
	10	53.170	4.39	0.331	2,904	**0.209**	3,616	0.122
	11	106.443	4.52	2.109	20,295	**0.455**	23,658	1.654
	12	225.836	4.43	1.881	14,637	**0.474**	17,072	1.407
	13	543.127	3.92	7.082	53,225	**1.010**	61,686	6.072
	14	1,238.322	4.51	68.153	454,644	**4.710**	408,391	63.443
M82	15	2,515.808	6.26	449.456	1,985,919	**32.263**	2,020,183	417.193
	16	5,098.459	5.13	19.058	1,085,704	**19.008**	1,053,939	0.050
	17	10,483.717	4.76	2,886.735	14,280,434	**169.935**	12,522,457	2,716.800
	18	21,440.172	5.53	940.869	3,794,152	**62.124**	3,603,313	878.745
	19	43,487.275	6.36	1,957.877	7,475,481	**124.122**	6,702,387	1,833.755
	20	87,392.343	8.21	> 3,600	12,562,902	**338.515**	17,631,694	n.a.

Table 7
Numerical results obtained for FastME and configuration PS* for $n = 20$ reported in the objective function $2^n \cdot L(T)$. For FastME, small deviations in $2^n \cdot L(T)$ compared to the results reported in the supplementary material in Lefort et al. (2015) arise from a different use of floating-point arithmetic, i.e., rounding errors. For configuration PS*, the gap at the root node of the search, the running time, and the number of branchings processed are shown.

Data set	FastME	PS*	Gap (%)	Time (s)	Branches
B-HA-573-585-BMGE	530,072.565	530,072.565	2.57	11.45	411,028
B-NA-684-472-BMGE	554,220.083	554,220.083	3.45	10.96	604,262
B-NS1-284-344-BMGE	600,436.354	600,436.354	4.69	3.21	116,224
proteic_M2577_40 × 12260_2005	4,793,364.626	4,793,364.626	6.36	43.91	2,779,707
proteic_M2624_139 × 348_2006-BMGE	3,569,327.875	3,569,327.875	1.43	26.94	1,477,087
proteic_M2883_91 × 7386_2007-BMGE	2,208,105.905	n.a.	6.87	> 3,600	n.a.
proteic_M3068_50 × 1000_2006	24,638,785.283	n.a.	2.48	> 3,600	n.a.
proteic_M3755_77 × 9918_2008-BMGE	5,063,227.006	5,063,227.006	1.00	228.71	12,371,673
proteic_M3756_77 × 11234_2008	6,639,947.471	n.a.	2.49	> 3,600	n.a.
B-HA-986-1908-BMGE	432,217.763	432,217.763	8.42	90.06	3,446,320
B-NA-633-1751-BMGE	288,685.631	288,685.631	6.08	15.89	599,867
B-NS-629-1111-BMGE	284,088.517	284,088.517	5.43	24.67	1,239,801
eudicots-BMGE	1,297,053.321	n.a.	2.56	> 3,600	n.a.
euros1	964,061.058	n.a.	3.75	> 3,600	n.a.
euros2	716,214.753	716,214.753	1.08	132.94	8,434,937
nucleic_M2573_346 × 897_2006	1,324,915.878	1,324,915.878	2.25	43.40	2,520,330
nucleic_M2839_470 × 829_2006	3,599,570.957	3,599,570.957	3.89	1,718.55	104,776,984
nucleic_M3862_362 × 1207_2008	3,102,271.026	n.a.	2.60	> 3,600	n.a.
rosids	1,038,195.249	n.a.	3.91	> 3,600	n.a.

Table 8
Continuation of Table 7 where the number of taxa and the number of primal solution updates are shown. In the last column the number of updates of the intermediate optimal solution (Up.) during the execution of PS* is shown.

Data set	n	FastME	PS*	Gap (%)	Time (s)	Branches	Up.
B-HA-573-585-BMGE	25	17,880,701	17,880,701	2.44	546.34	21,598,004	0
	26	35,655,750	n.a.	2.55	> 3,600	n.a.	0
B-NA-684-472-BMGE	24	9,467,537	9,467,537	4.25	2,758.68	89,888,929	0
	25	18,911,659	n.a.	4.23	> 3,600	n.a.	0
B-NS1-284-344-BMGE	29	341,239,418	341,239,418	6.69	2,124.72	44,871,572	0
	30	689,965,539	n.a.	7.34	> 3,600	n.a.	0
proteic_M2577_40 × 12260_2005	23	41,973,145	41,973,145	7.30	814.48	38,575,777	0
	24	86,565,511	n.a.	7.74	> 3,600	n.a.	0
proteic_M2624_139 × 348_2006-BMGE	25	125,549,247	125,549,247	2.89	703.86	19,630,921	0
	26	257,595,763	n.a.	2.96	> 3,600	n.a..	0
proteic_M3755_77 × 9918_2008-BMGE	22	21,115,546	21,113,170	1.06	> 3,600	117,055,834	2
	23	43,191,028	n.a.	1.08	> 3,600	n.a.	0
B-HA-986-1908-BMGE	21	869,072	n.a.	7.35	> 3,600	n.a.	0
B-NA-633-1751-BMGE	22	1,270,085	1,270,085	7.22	1,449.53	78,057,725	0
	23	2,660,926	n.a.	8.23	> 3,600	n.a.	0
B-NS-629-1111-BMGE	26	19,225,291	19,225,291	8.19	1,320.43	49,798,676	0
	27	39,704,665	n.a.	8.44	> 3,600	n.a.	0
euros2	22	2,949,200	2,949,200	1.10	84.00	3,802,128	0
	23	5,959,679	n.a.	1.76	> 3,600	n.a.	0
nucleic_M2573_346 × 897_2006	27	171,662,192	171,652,350	2.92	1,587.63	43,125,380	1
	28	345,752,913	345,731,096	2.88	> 3,600	95,013,870	2
nucleic_M2839_470 × 829_2006	21	7,507,514	7,507,514	4.14	3,141.93	170,364,597	0
	22	15,439,953	n.a.	4.27	> 3,600	n.a.	0

Table 9
Numerical results obtained for FastME and configuration PS* for the random instances on 24 taxa given by the 24-taxa trees set discussed in Desper and Gascuel (2002). The relative gap is the difference between the optimal solution of FastME and configuration PS* relative to PS*.

Average gap at the root node of the search	5.66%
Configuration PS* terminated (within one hour)	80.10%
Average number of branchings	26, 206, 315.71
Average number of branchings when configuration PS* terminated	9, 963, 219.75
Average solution time	1, 021.86 s
Average solution time when configuration PS* terminated	381.35 s
FastME does not solve the BMEP up to optimality	13.25%
Average number of primal updates when FastME does not solve the BMEP up to optimality	2.23
Relative gap when FastME does not solve the BMEP up to optimality	0.05%
FastME does solve the BMEP up to optimality	$\geq$ 67.80%

design exact solution algorithms outperforming the current state-of-the-art algorithm described in Catanzaro et al. (2012). Hence, in this paper, we investigated ways to push to the limit the performance of the current state-of-the-art by combining it with the recent theoretical advances in the combinatorics, information theory, and optimization aspects of the BMEP described in Catanzaro et al. (2012), Semple and Steel (2004), Catanzaro et al. (2020b, 2015), Aringhieri et al. (2011), Catanzaro and Pesenti (2019), Catanzaro et al. (2020a), Gascuel and Steel (2006) and Gascuel and Steel (2016), specific tree-coding schemes, and the multicore (shared-memory) features of modern CPUs. The resulting massively parallel exact solution algorithm proves to be up to one order of magnitude faster than the current state-of-the-art under the same computing settings and environment and be able to solve up to 25% bigger BMEP instances within a prefixed time limit.

Data availability

Data will be made available on request

Acknowledgments

The first author acknowledges support from the Belgian National Fund for Scientific Research, Belgium (FNRS) via the grant FNRS PDR 40007831, and the Foundation Louvain, Belgium via the grant "COALESCENS". The second author acknowledges support from the European Union's Horizon 2020 research and innovation program under the Marie Skłodowska-Curie grant agreement no. 101034253, and by the NWO Gravitation project NETWORKS under grant no. 024.002.003. Computational resources have been provided by the supercomputing facilities of the Université catholique de Louvain (CISM/UCL) and the Consortium des Équipements de Calcul Intensif en Fédération Wallonie Bruxelles (CÉCI) funded by the Fond de la Recherche Scientifique de Belgique (F.R.S.-FNRS) under convention 2.5020.11 and by the Walloon Region.

References

Allen, B.L., Steel, M., 2001. Subtree transfer operations and their induced metrics on evolutionary trees. Ann. Comb. 5, 1–15.
Aringhieri, R., Catanzaro, D., Di Summa, M., 2011. Optimal solutions for the balanced minimum evolution problem. Comput. Oper. Res. 38, 1845–1854.
Auch, A.F., R., H.S., Holland, B.R., M., G., 2006. Genome BLAST distance phylogenies inferred from whole plastid and whole mitochondrion genome sequences. BMC Bioinformatics 7 (350), 1–5.
Batagelj, V., Pisanski, T., Simoes-Pereira, J.M., 1990. An algorithm for tree-realizability of distance matrices. Int. J. Comput. Math. 34 (171–176).
Bordewich, M., Gascuel, O., Huber, K.T., Moulton, V., 2009. Consistency of topological moves based on the balanced minimum evolution principle of phylogenetic inference. IEEE/ACM Trans. Comput. Biol. Bioinform. 6 (1), 110–117.
Buneman, P., 1971. The recovery of trees from measure of dissimilarities. In: Hodson, F.R., Kendall, D.G., Tautu, P. (Eds.), Archaeological and Historical Science. Edinburgh University Press, Edinburgh, UK, pp. 387–395.
Buneman, P., 1974. A note on the metric properties of trees. J. Combin. Theory 17 (B), 48–50.
Caminiti, S., Finocchi, I., Petreschi, R., 2007. On coding labeled trees. Theoret. Comput. Sci. 382, 97–108.
Catanzaro, D., Aringhieri, R., di Summa, M., Pesenti, R., 2015. A branch-price-and-cut algorithm for the minimum evolution problem. European J. Oper. Res. 244 (3), 753–765.
Catanzaro, D., Frohn, M., Gascuel, O., Pesenti, R., 2022. A tutorial on the balanced minimum evolution. European J. Oper. Res. 300 (1), 1–19.

Catanzaro, D., Frohn, M., Pesenti, R., 2020a. An information theory perspective on the balanced minimum evolution problem. Oper. Res. Lett. 48 (3), 362–367.
Catanzaro, D., Labbé, M., Pesenti, R., Salazar-Gonzáles, J.J., 2012. The balanced minimum evolution problem. INFORMS J. Comput. 24 (2), 276–294.
Catanzaro, D., Pesenti, R., 2019. Enumerating vertices of the balanced minimum evolution polytope. Comput. Oper. Res. 109, 209–217.
Catanzaro, D., Pesenti, R., Milinkovitch, M., 2006. A non-linear optimization procedure to estimate distances and instantaneous substitution rate matrices under the GTR model. Bioinformatics 22 (6), 708–715.
Catanzaro, D., Pesenti, R., Wolsey, L.A., 2020b. On the balanced minimum evolution polytope. Discrete Optim. 36, 1–33.
Çela, E., 1997. The Quadratic Assignment Problem. Kluwer Academic Publishers, Boston, MA.
Chandra, R., Dagum, L., Kohr, D., Menon, R., Maydan, D., McDonald, J., 2001. Parallel Programming in Open MP. Morgan Kaufmann.
Cheng, X., Du, D.Z., 2001. Steiner Trees in Industry. Kluwer Academic Publishers, Boston, MA.
Cieslik, D., 1998. Steiner Minimal Trees. Springer, Boston, MA, USA.
Criscuolo, A., Michel, C.J., 2009. Phylogenetic inference with weighted codon evolutionary distances. J. Mol. Evol. 68, 377–392.
Cueto, M.A., Matsen, F.A., 2011. Polyhedral geometry of phylogenetic rogue taxa. Bull. Math. Biol. 73 (6), 1202–1226.
Desper, R., Gascuel, O., 2002. Fast and accurate phylogeny reconstruction algorithms based on the minimum evolution principle. J. Comput. Biol. 9 (5), 687–705.
Desper, R., Gascuel, O., 2004. Theoretical foundations of the balanced minimum evolution method of phylogenetic inference and its relationship to the weighted least-squares tree fitting. Mol. Biol. Evol. 21 (3), 587–598.
Desper, R., Gascuel, O., 2008. Chapter Distance-based phylogeny reconstruction (optimal radius). In: Kao, Ming-Yang (Ed.), Encyclopedia of Algorithms. Springer, pp. 1–99.
Du, D.Z., Hu, X., 2008. Steiner Tree Problems in Computer Communication Networks. World Scientific Publishing Company, Singapore.
Du, D.Z., Smith, J.M., Rubinstein, J.H., 2000. Advances in Steiner Trees. Kluwer Academic Publishers, Boston, MA, USA.
Felsenstein, J., 1984. Distance methods for inferring phylogenies: A justification. Evolution 38, 16–24.
Felsenstein, J., 2004. Inferring Phylogenies. Sinauer Associates, Sunderland, MA.
Fiorini, S., Joret, G., 2012. Approximating the balanced minimum evolution problem. Oper. Res. Lett. 40 (1), 31–35.
Forcey, S., Keefe, L., Sands, W., 2015. Facets of the balanced minimal evolution polytope. J. Math. Biol. 73 (2), 447–468.
Forcey, S., Keefe, L., Sands, W., 2017. Split-facets for balanced minimal evolution polytopes and the permutoassociahedron. Bull. Math. Biol. 79, 975–994 (in press).
Frohn, M., 2020. On the approximability of the fixed-tree balanced minimum evolution problem. Optim. Lett. (in press).
Gascuel, O., 1994. A note on Sattath and Tversky's, Saitou and Nei's and Studier and Keppler's algorithms for inferring phylogenies from evolutionary distances. Mol. Biol. Evol. 11, 961.
Gascuel, O., 1997. BIONJ: An improved version of the NJ algorithm based on a simple model of sequence data. Mol. Biol. Evol. 14 (7), 685–695.
Gascuel, O., 2005. Mathematics of Evolution and Phylogeny. Oxford University Press, New York, NY.
Gascuel, O., Steel, M.A., 2006. Neighbor-joining revealed. Mol. Biol. Evol. 23 (11), 1997–2000.
Gascuel, O., Steel, M., 2016. A 'stochastic safety radius' for distance-based tree reconstruction. Algorithmica 74, 1386–1403.
Gusfield, D., 1984. The Steiner Tree Problem in Phylogeny. Technical Report 334, Dep. Computer Science, Yale University, New Haven, CT.
Haws, D.C., Hodge, T.L., Yoshida, R., 2011. Optimality of the neighbor joining algorithm and faces of the balanced minimum evolution polytope. Bull. Math. Biol. 73, 2627–2648.
Hendy, M.D., Penny, D., 1982. Branch and bound algorithms to determine minimal evolutionary trees. Math. Biosci..
Hwang, F.K., Richards, D.S., Winter, P., 1992. The Steiner Tree Problem. North-Holland, Amsterdam, The Netherlands.
Jung, M., 2012. Évolution Du VIH: MÉthodes, ModÈles et Algorithmes (Ph.D. thesis). Université Montpellier II - Sciences et Techniques du Languedoc, France.
Le, S.Q., Gascuel, O., 2008. An improved general amino acid replacement matrix. Mol. Biol. Evol. 25 (7), 1307–1320.
Lefort, V., Desper, R., Gascuel, O., 2015. FastME 2.0: A comprehensive, accurate, and fast distance-based phylogeny inference program. Mol. Biol. Evol. 32 (10), 2798–2800.
Li, X., Mao, Y., 2016. Generalized Connectivity of Graphs. Springer, Boston, MA, USA.
Lu, C.L., Tang, C.Y., Lee, R.C.T., 2003. The full Steiner tree problem. Theoret. Comput. Sci. 306, 55–67.
Page, R.D.M., Holmes, E.C., 1998. Molecular Evolution: A Phylogenetic Approach. Blackwell Science, Oxford, UK.
Pardi, F., 2009. Algorithms on Phylogenetic Trees (Ph.D. thesis). University of Cambridge, UK.
Pardi, F., Gascuel, O., 2012. Combinatorics of distance-based tree inference. Proc. Natl. Acad. Sci. USA 109 (41), 16443–16448.
Pardi, F., Gascuel, O., 2016. Encyclopedia of evolutionary biology. In: Kliman, Richard M. (Ed.), Distance-Based Methods in Phylogenetics. Elsevier, pp. 458–465.
Pardi, F., Guillemot, S., Gascuel, O., 2010. Robustness of phylogenetic inference based on minimum evolution. Bull. Math. Biol. 72, 1820–1839.
Parker, D.S., Ram, P., 1996. The construction of Huffman codes is a submodular ("convex") optimization problem over a lattice of binary trees. SIAM J. Comput. 28 (5), 1875–1905.
Pop, P.C., 2012. Generalized Network Design Problems: Modeling and Optimization. De Gruyter, Berlin, Germany.
Prömel, H.J., Steger, A., 2002. The Steiner Tree Problem: A Tour Through Graphs, Algorithms, and Complexity. Vieweg+Teubner Verlag, Berlin.
Saitou, N., Nei, M., 1987. The neighbor-joining method: A new method for reconstructing phylogenetic trees. Mol. Biol. Evol. 4, 406–425.
Semple, C., Steel, M.A., 2004. Cyclic permutations and evolutionary trees. Adv. Appl. Math. 32 (4), 669–680.
Steel, M., 2005. Phylogenetic diversity and the greedy algorithm. Syst. Biol. 54 (4), 527–529.
Studier, J.A., Keppler, K.J., 1988. A note on the neighbor-joining algorithm of Saitou and Nei. Mol. Biol. Evol. 5, 729–731.
Wu, B.Y., Chao, K.M., 2004. Spanning Trees and Optimization Problems. Chapman and Hall/CRC, Boca Raton, FL.
Yang, Z., 2014. Molecular Evolution: A Statistical Approach. Oxford University Press, Oxford, UK.