IRIS AperTO

UNIVERSITÀ DEGLI STUDI DI TORINO

AperTO - Archivio Istituzionale Open Access dell'Università di Torino

Optimal solutions for the Balanced Minimum Evolution Problem

This is the author's manuscript

Original Citation:

Availability:

This version is available http://hdl.handle.net/2318/86826 since 2015-12-11T00:46:38Z

Published version:

DOI:10.1016/j.cor.2011.02.020

Terms of use:

Open Access

Anyone can freely access the full text of works made available as "Open Access". Works made available under a Creative Commons license can be used according to the terms and conditions of said license. Use of all other works requires consent of the right holder (author or publisher) if not exempted from copyright protection by the applicable law.

(Article begins on next page)

14 November 2019

UNIVERSITÀ DEGLI STUDI DI TORINO

This is an author version of the contribution published on:

R. Aringhieri, D. Catanzaro, and M. Di Summa.

Optimal solutions for the Balanced Minimum Evolution Problem.

Computers & Operations Research, 38(12):1845-1854, 2011.

DOI: 10.1016/j.cor.2011.02.020

The definitive version is available at:

http://www.sciencedirect.com/science/article/pii/S0305054811000621

Optimal solutions for the Balanced Minimum Evolution Problem

Roberto Aringhieri* Daniele Catanzaro† Marco Di Summa‡

February 24, 2011

Abstract

Phylogenies are trees representing the evolutionary relationships of a set of species (called taxa). Phylogenies find application in several scientific areas ranging from medical research to drug discovery, epidemiology, systematics and population dynamics. In these applications the available information is usually restricted to the leaves of a phylogeny and is represented by molecular data extracted from the species analysed. On the contrary, the information about the phylogeny itself is generally missing and must be determined by solving an optimisation problem, called the Phylogeny Estimation Problem (PEP), whose versions depend on the criterion used to select a phylogeny among plausible alternatives.

In this paper, we investigate one of the most significant versions of the PEP, called the *Balanced Minimum Evolution Problem*. We propose an exact algorithm based on the enumeration of non-isomorphic trees and the subsequent solution of quadratic assignment problems. Furthermore, by exploiting the underlying parallelism of the algorithm, we present a parallel version of the algorithm which shows a linear speed-up with respect to the sequential version. Extensive computational results prove the effectiveness of the proposed algorithms.

Keywords: combinatorial optimisation, quadratic assignment, computational biology, balanced minimum evolution.

1 Introduction

Molecular phylogenetics studies the hierarchical evolutionary relationships among organisms (also called *taxa*). Starting from the evaluation of dissimilarity of molecular data extracted from taxa, molecular phylogenetics aims at reconstructing the evolutionary history (phylogeny) that, from a hypothetical

*Dipartimento di Informatica, Università degli Studi di Torino. Corso Svizzera 135, I-10149 Torino, Italy.

†Graphes et Optimisation Mathématique (G.O.M.), Département d'Informatique, Université Libre de Bruxelles (U.L.B.). Boulevard du Triomphe, CP 210/01, B-1050, Brussels, Belgium.

‡Dipartimento di Informatica, Università degli Studi di Torino. Corso Svizzera 135, I-10149 Torino, Italy. Currently moving to EPFL SB IMA DISOPT, Station 8, CH-1015 Lausanne, Switzerland.

common ancestor, have given rise to the various taxa that can be observed at present.

Phylogenies find application in several scientific areas ranging from medical research to drug discovery, epidemiology, systematics and population dynamics [16, 19]. In these applications the available information is usually restricted to taxa, which are represented by means of molecular data extracted from the species analysed, such as DNA, RNA, amino acid or codon fragments. On the contrary, the information about the phylogeny itself is generally missing and can be determined by solving an optimisation problem, called the Phylogeny Estimation Problem (PEP), whose versions depend on the criterion used to select a phylogeny among plausible alternatives [8]. In this article we investigate one of the most significant versions of the PEP, first introduced by Pauplin [21] and known as the *Balanced Minimum Evolution Problem* (BMEP).

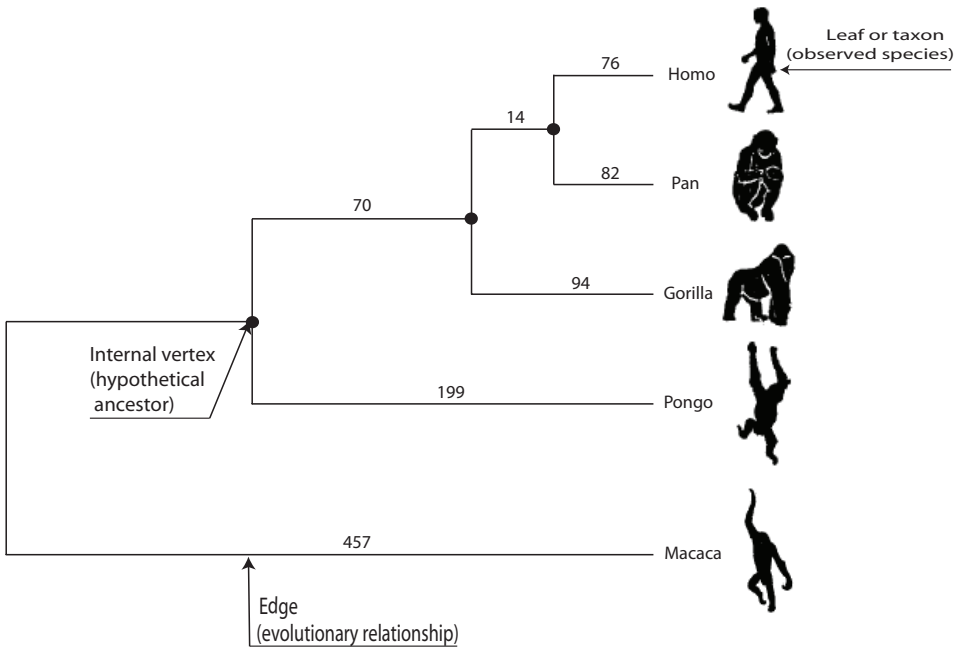

Figure 1: An example of a phylogeny with five taxa (Homo, Pan, Gorilla, Pongo and Macaca) and three internal vertices (•). The picture is from [8].

A phylogeny is usually represented by means of a weighted tree (see Figure 1). The leaves of the tree correspond to taxa, whereas the internal vertices represent the intermediate ancestors. Edges indicate evolutionary relationships between organisms. Every edge has a weight, which is a measure of the dissimilarity between the species associated with its endpoints.[1] Without loss of generality, every internal vertex is assumed to have degree equal to 3: if this is not the case, it is easy to add dummy nodes and edges with weight zero so that an equivalent phylogeny is obtained, where the degree of each internal vertex is 3 [8].

[1]This dissimilarity measure is based on molecular data extracted from the species (see, e.g., [10] for a discussion of this topic).

Given a set Γ of n taxa, consider an $n \times n$ symmetric distance matrix $\mathbf{D}$, whose generic entry d_{ij}, $i, j \in \Gamma$, represents a measure of dissimilarity between the corresponding pair of molecular data. Then the BMEP consists in finding a phylogeny T, having Γ as leaf set, that minimises the following *length function*:

$$\ell(T) = \sum_{e \in T} w_e = \sum_{i,j \in \Gamma} \frac{d_{ij}}{2^{\tau_{ij}^T}}, \tag{1}$$

where w_e is the weight of the edge e in the phylogeny T, and for $i, j \in \Gamma$, τ_{ij}^T is the number of edges belonging to the path between i and j in T. In the following, for the sake of simplicity, we write τ_{ij} instead of τ_{ij}^T. In other words, the set of feasible solutions of the BMEP is the set of all phylogenies with n leaves, and one is required to select a phylogeny minimising function (1).

The meaning and the properties of length function (1) have been discussed in depth in [12, 21]. Specifically, the peculiar expression of the objective mainly derives from a modification of the Ordinary Least-Squares edge weight estimation model [8] in which each edge weight, inducing a bipartition on the set of leaves of T, is independent of the cardinality of such subsets. Such a modification makes the length function independent of the edge weights w_e and dependent only on the topological distances τ_{ij}. We refer the interested reader to the seminal works of [12, 21] for an introduction and to [14, 9] for a systematic discussion of the combinatorial nature of the BMEP.

The optimal solution T^* of the BMEP is known to be *statistically consistent*, i.e., as the amount of molecular data analysed from taxa increases, T^* approaches the phylogeny that one would obtain if all the molecular data from taxa were available [12]. For this reason at least, solving exactly the BMEP is highly desirable. To the best of our knowledge, the only attempts to exactly solve instances of the BMEP are restricted to the use of implicit enumeration algorithms, such as those recently proposed by Pardi [20]. Specifically, from the combinatorial interpretation of the length function (1) given in [24], Pardi derived a number of lower bounds for the problem that led to an exact branch-and-bound algorithm.

Though it is not known whether the BMEP is $\mathcal{NP}$-hard, this problem seems to be very difficult to solve. This has justified the development of heuristic solution approaches, such as that proposed by Desper and Gascuel [12]. Specifically, the authors proposed a $O(n^3)$ constructive greedy heuristic that, starting from a partial phylogeny T_m of Γ, i.e., an m-leaf phylogeny whose leaves are taxa of a subset $\Gamma' \subset \Gamma$ with $|\Gamma'| = m$, iteratively constructs partial phylogenies with an increasing number of leaves until all taxa in Γ are included.

In this paper we propose an innovative method to find exact solutions of the BMEP based on an explicit enumeration approach. More specifically, we exploit a tree coding model to provide an efficient representation of BMEP solutions, and the isomorphism between trees to reduce the solution space, i.e., the number of solutions enumerated. (By "tree isomorphism" we mean a bijection between the vertex sets of two trees that preserves node adjacency; see Section 2 for a more formal definition.) The resulting algorithm can be seen as an interaction of a first phase in which unlabeled non-isomorphic spanning trees

are generated, and a second phase in which a Quadratic Assignment Problem (QAP) [11] is solved. Furthermore, by exploiting the underlying parallelism of the algorithm, we present a parallel version of the algorithm which shows a linear speed-up with respect to the sequential version.

The rest of the paper is organised as follows. In Section 2 we investigate the relationship between the BMEP and the QAP. In Section 3 we present our algorithms for the BMEP and we point out that our approach can be easily extended to deal with the problem of finding the optimal phylogeny with respect to other evaluation criteria. In Section 4 we discuss the computational experiments carried out on some biological datasets and compare our results with those obtained by our implementation of Pardi's algorithm [20]. Section 5 closes the paper.

2 The BMEP and the QAP

Given a set Γ of n taxa, we formally define a *phylogeny* for Γ as a tree T such that the leaf set of T is Γ and every non-leaf node of T has degree 3. Note that any phylogeny for Γ has exactly $(n-2)$ non-leaf nodes (also called internal nodes). We denote by $\mathcal{T}$ the set of all phylogenies for Γ.

We remark that we are not interested in labeling the internal nodes of a phylogeny. In fact, when reconstructing the evolutionary history of the taxa in Γ, the ancestors (i.e., the internal nodes) are not known; once a phylogeny T for Γ is given, the biological role of the internal nodes is implicitly determined by the structure of T itself.

Two phylogenies $T_1, T_2 \in \mathcal{T}$ are said to be *isomorphic* if there exists a graph isomorphism between T_1 and T_2, i.e., if there exists a bijection φ from the vertex set of T_1 to that of T_2 such that any two nodes u, v are adjacent in T_1 if and only if $\varphi(u), \varphi(v)$ are adjacent in T_2. Thus, if we ignore the labels of the leaves (i.e., the correspondence between leaves and taxa), it is impossible to distinguish between two isomorphic phylogenies. Hence, given a phylogeny T, we consider all phylogenies that are isomorphic to T as a single *unlabeled phylogeny*. More formally, an unlabeled phylogeny can be seen as an equivalence class with respect to the relation of isomorphism on the set of phylogenies. To stress the contrast with unlabeled phylogenies, we will sometimes use terminology *labeled phylogenies* to indicate phylogenies.

Note that the operation of selecting a phylogeny for Γ can be thought as a sequence of two sub-operations: first choosing an unlabeled phylogeny and then assigning taxa to leaves (as observed above, we are not required to consider the assignment of intermediate ancestors to internal nodes.) We now show that if we ignore the former sub-operation and restrict ourselves to a given class of isomorphic phylogenies, then the BMEP reduces to an instance of the well known Quadratic Assignment Problem (QAP) [11].

Let U be an unlabeled phylogeny for Γ and L be the set of leaves of U. Then the BMEP, restricted to those labeled phylogenies T whose corresponding unlabeled phylogeny is U, consists in assigning taxa to the leaves of U so that the length function (1) is minimised. This problem can be formally stated as

follows:

Formulation. *Leaf-Assignment Problem:*

$$\text{Find a bijection } \sigma : \Gamma \mapsto L \text{ minimising } \sum_{i,j \in \Gamma} \frac{d_{ij}}{2^{\tau_{\sigma(i)\sigma(j)}}}. \tag{2}$$

The formulation given by (2) is an instance of the QAP. In the next section we shall exploit the relation between the BMEP and the QAP to develop an exact approach to determine an optimal solution of the problem.

3 An exact algorithm for the BMEP

Given a set Γ of n taxa, the number of possible phylogenies for Γ is $|\mathcal{T}| = (2n-5)!! = 1 \cdot 3 \cdot 5 \cdots (2n-5)$ (see [13]). A possible natural approach to exactly solve instances of the BMEP consists in enumerating all phylogenies $T \in \mathcal{T}$, computing $\ell(T)$ for each $T \in \mathcal{T}$, and returning a phylogeny for which the value $\ell(T)$ is minimum. However, as the number of phylogenies grows up exponentially in function of n, such an enumerative approach becomes quickly intractable even for small values of n (e.g., $n > 10$).

Taxa	Labeled phylogenies	Unlabeled phylogenies	Taxa	Labeled phylogenies	Unlabeled phylogenies
4	3	1	13	$\sim 1.4 \cdot 10^{10}$	66
5	15	1	14	$\sim 3.2 \cdot 10^{11}$	135
6	105	2	15	$\sim 7.9 \cdot 10^{12}$	265
7	945	2	16	$\sim 2.1 \cdot 10^{14}$	552
8	10,395	3	17	$\sim 6.2 \cdot 10^{15}$	1132
9	135,135	4	18	$\sim 1.9 \cdot 10^{17}$	2410
10	2,027,025	11	19	$\sim 6.3 \cdot 10^{18}$	5098
11	34,459,425	18	20	$\sim 2.2 \cdot 10^{20}$	11,020
12	654,729,075	37	25	$\sim 2.5 \cdot 10^{28}$	565,734

Table 1: Number of labeled and unlabeled phylogenies for a given number of taxa.

However, as shown in Table 1, the number of unlabeled phylogenies, though yet characterised by an exponential growth, is in general much smaller than the number of labeled phylogenies. Hence, an alternative and possibly better approach to solution of the BMEP could consists of the following phases:

1. Enumerate all unlabeled phylogenies with n leaves;
2. For each unlabeled phylogeny, solve the Leaf-Assignment Problem stated in (2): this gives an optimal labeled phylogeny within a single class of isomorphic phylogenies;
3. Among the optimal labeled phylogenies found in Step 2, return one minimising (1).

Implementing this approach is however nontrivial, as it requires an efficient algorithm for enumerating the unlabeled phylogenies for Γ and a fast optimisation algorithm for solving the QAP (2). In the next subsections we discuss how to implement this strategy.

3.1 Enumerating unlabeled phylogenies

The problem of generating the set of unlabeled phylogenies with n taxa can be seen as a special case of the enumeration of degree-constrained trees. Aringhieri, Hansen and Malucelli [1] proposed several new algorithms for generating this kind of trees. In particular, they tested their algorithms generating the alkane molecular family, which consists of trees having maximum degree equal to four. Among the algorithms presented in [1], we adopt the one-to-one enumeration method, as it can be easily extended to face the problem of generating unlabeled phylogenies.

The one-to-one enumeration method simply generates a tree with k vertices by adding t vertices to a tree with $k-t$ vertices in such a way that the degree constraint is still satisfied, where t is as small as possible. In the case of the alkane family, t is equal to 1, whereas in the case of phylogenies, where the degree of every internal node must be exactly equal to 3, t is equal to 2: a phylogeny with k vertices can be obtained by connecting two new vertices to a leaf belonging to a phylogeny with $k-2$ vertices. Figure 2 depicts this type of enumeration.

Figure 2: An example of one-to-one enumeration: tree with 8 vertices (5 taxa) generated from a tree with 6 vertices (4 taxa).

To develop an efficient algorithm for degree-constrained trees enumeration, trees should be suitably encoded, as a code can be easily handled by computer programs. Several tree codes are proposed in the literature, see e.g., [17, 22, 26]. In our implementation, we use the CN-tuple code introduced in [15].

In order to define the CN-tuple code of a tree, we need to recall the concept of center of a tree. Given a tree, iteratively perform the following operation: if the tree has more than two nodes, remove all the leaves. When only one or two nodes are left, the remaining node (or each of the remaining nodes) is called the center of the tree.

Now, the CN-tuple code of a tree can be defined as follows. If the tree consists of a single node, its code is 0. Otherwise, we proceed recursively as follows. Let r be the center of the tree and let g be the number of vertices

$(v_1, v_2, \ldots, v_g)$ adjacent to r; we remove node r from the tree; we compute the code of the subtrees rooted at $v_1, v_2, \ldots, v_g$; then we concatenate these codes in such a way as to obtain a lexicographically maximum sequence S; finally, the CN-tuple of the tree is given by the concatenation of g and S. In the case of two center nodes r_1 and r_2, the CN-tuple is the lexicographically maximum code between the CN-tuple for r_1 and for r_2. For instance, the CN-tuple code for the two trees in Figure 2 are 320000 (tree (a)) and 32002000 (tree (b)).

As pointed out in [1], tree enumeration algorithms need methods to guarantee the unique enumeration of a tree. Let us consider the tree in Figure 2(a), from which four trees can be generated by adding two vertices to each of the four leaves. Since we are dealing with unlabeled trees, these four trees are in fact the same tree 32002000, i.e., the one in Figure 2(b). This illustrates multiple generation of a tree from the same initial tree. Another possibility is multiple generation of a tree from different initial trees.

The former case can be managed by inspection detecting symmetries in the starting tree: the use of coded trees make the implementation of this approach easier, as described in [1]. The detection of multiple generation of a tree from different trees can require more sophisticated techniques: Aringhieri, Hansen and Malucelli [1] proposed an extension of the reverse search technique introduced by Avis and Fukuda for the enumeration of vertices of polyhedra [2, 3]. We implement a detection algorithm based on hashing data structure: in our computational test, hashing technique proves to be more efficient than the reverse search even though it requires a larger use of main memory.

3.2 Solving the QAP

For the solution of the QAP (2) we make use of the branch-and-bound by Burkard and Derigs [6] that solves QAPs to optimality by means of a perturbation method.

A Fortran code [7] is available for this algorithm: the code `qapbb.f` is a modified version of that presented in [6] – a linear term can be included – and is quite efficient on problems of size $n \leq 15$. This routine accepts as input an arbitrary QAP instance of size at most $n \leq 33$ with integer data and returns an optimal assignment. Since the coefficients in objective function (2) are not integer, before calling Burkard's routine we need to convert the coefficients in (2) into integer values: this can be done by multiplying them by $2^M \cdot 10^K$, where M is the maximum number of edges between two leaves in the unlabeled phylogeny and K is the maximum number of decimals in d_{ij} for $i, j \in \Gamma$.

3.3 The algorithm

We are now ready to present our exact algorithm for the BMEP, which given the number of taxa n and the matrix $\mathbf{D} = \{d_{ij}\}$ of evolutionary distances between pairs of taxa, computes a phylogeny T^* minimising the length function (1). The algorithm, named BMEPsolver, is described in pseudo-code.

The conversion of $\mathbf{D}$ into a matrix with integer entries (line 2 of the algorithm) is performed as explained in Section 3.2. The unlabeled phylogenies

Algorithm 1: Algorithm BMEPsolver

Input : n (number of taxa); $\mathbf{D}$ ($n \times n$ distance matrix).
Output: T^* (optimal solution of (1)).

```
1  begin
2      transform D into an integral matrix;
3      z* ← +∞;
4      enumerate all unlabeled phylogenies with n leaves;
5      for each unlabeled phylogeny U do
6          for each pair of taxa i, j do compute the edge distance τ_ij;
7          (σ, z) ← optimal solution and optimal value of (2);
8          T ← corresponding labeled phylogeny;
9          if z < z* then (T*, z*) ← (T, z);
10 end
```

(line 4) are enumerated as described in Section 3.1. The calculation of values τ_{ij} (line 6) can be easily implemented by an iterated depth-first search on U. Finally, problem (2) (line 7) is solved by means of Burkard's routine (see Section 3.2).

We remark that if one wishes to solve several different instances of the BMEP with the same number of taxa n, the routine for enumerating the unlabeled phylogenies with n taxa needs to be run just once and the list of tree codes can be stored in a text file. Line 4 will then consist in just reading a new string from the text file.

We also point out that our approach can be seen as a general purpose algorithm for determining the optimal phylogeny with respect to different evaluation criteria, such as those reported in [8]: given an evaluation criterion defined on unlabeled phylogenies, our approach can be adapted by simply including in line 7 of the algorithm a procedure for determining the exact solution for the corresponding optimisation problem.

3.4 Parallel implementation

Algorithm BMEPsolver is particularly fit for parallelization. Specifically, the following straightforward parallelization is proposed: given processes $p_0, \ldots, p_k$, process p_0 enumerates all unlabeled phylogenies; for each unlabeled phylogeny, p_0 sends the corresponding encoding to one of processes $p_1, \ldots, p_k$, which executes steps 6–8 of Algorithm BMEPsolver and then sends back the optimal assignment to process p_0. Figure 3 depicts our parallel implementation.

4 Computational experiments

In this section we discuss some computational experiments testing BMEPsolver on a set of biological instances. We compare the performance of our algorithm with that of our implementation of Pardi's algorithm. Finally, we discuss and test the parallel implementation of the algorithm.

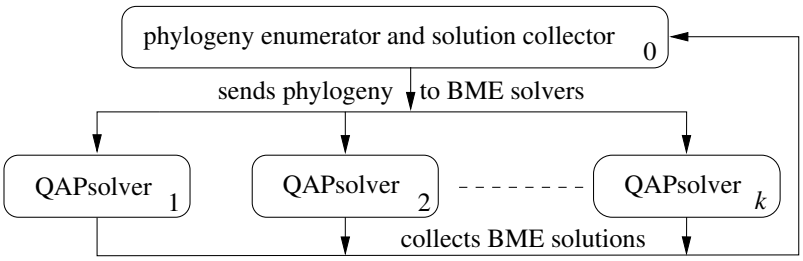

Figure 3: Description of the parallel algorithm.

4.1 Computational environments

All the algorithms proposed are coded in C standard and compiled using the GNU compiler `gcc`. MPI [25] is used for the parallel version of the algorithm. The tests comparing our sequential algorithm with that of Pardi were run on a machine with 2.4 GHz CPU and 512 MB of RAM, while the experiments comparing the sequential and the parallel version of BMEPsolver were run on a cluster with 18 4-ways nodes AMD Opteron 2.2 GHz CPU single core and 2 GB of RAM, and 37 2-ways nodes with AMD Opteron 2.2 GHz CPU dual core with 4 GB of RAM, running under Linux operating system and maintained at Cilea, Italy.

An instance of BME is usually constituted by a set of molecular sequences (typically DNA or protein sequences) that are appropriately preprocessed, e.g., by means of estimation procedures such as those described in [10], and transformed in pairwise distances $\{d_{ij}\}$. Here we consider a number of real aligned DNA datasets, as described in Table 2: for each dataset, we give the number of molecular sequences (i.e., the number of taxa) contained in the dataset, the number of characters that form each sequence, and the biological entity from which the molecular sequences are taken. From each dataset we have extracted the first 20 taxa (or all taxa if the dataset contains less than 20 taxa) and built the associated distance matrices by using the General Time Reversible (GTR) model of DNA sequence evolution [23, 18, 27]. The estimation method used to obtained GTR distances is described in [10]. Moreover, from each distance matrix we have extracted the corresponding k-th leading principal submatrices, $k \in \{10, \ldots, \max\}$, where max is 12 for Primates12, 17 for M17, 18 for M18, and 20 for the remaining datasets, generating therefore an overall number of 84 real instances of the BMEP. The distance matrices are available at the following url: `http://homepages.ulb.ac.be/~dacatanz/COR_ACD.zip`.

4.2 Comparison with Pardi's algorithm

To the best of our knowledge, the only attempts to exactly solve instances of the BMEP are restricted to the use of implicit enumeration algorithms. One of the most efficient algorithms for the BMEP was recently proposed by Pardi [20]. Specifically, from the combinatorial interpretation of the length function (1) given in [24], Pardi derived a number of lower bounds for the problem that led to an exact branch-and-bound algorithm. We report on the comparison

Dataset name	Number of sequences	Characters per sequence	Extracted from
M18/8128	18	8128	cetacea
M43/2086	43	2086	mammals
M62/3768	62	3768	hyracoidae
M82/2062	82	2062	fungi
M17/2550	17	2550	insects
RbcL55/1314	55	1314	rbcL gene
Rana64/1976	64	1976	ranoid frogs
Plant25/19784	25	19784	pinoles
Primates12/898	12	898	primates

Table 2: Description of the instances.

between the performance of BMEPsolver and our implementation of Pardi's algorithm.

First, we remark that, because of the observation made in Section 3.3, the computing time for BMEPsolver does not include the time needed to generate the list of encodings of all unlabeled phylogenies (anyway, this amount of time is negligible – as reported in Table 3 – since in our case the largest instances have 20 taxa).

Taxa	Unlabeled phylogenies	Time (s)	Taxa	Unlabeled phylogenies	Time (s)
20	11020	0.070	24	254371	2.796
21	23846	0.161	25	565734	8.400
22	52233	0.387	26	1265579	24.537
23	114796	1.008	27	2841632	71.496

Table 3: Computing time for enumerating unlabeled phylogenies.

We tried to solve the instances described in Section 4.1 both with Pardi's algorithm and with BMEPsolver. In both cases, the time limit was set to one hour. For each instance, the computation time (in seconds) and the best solution found are given in detail in the appendix (Table 8). When the algorithm terminates within the time limit, the best solution is indeed the optimal solution, whilst "N/A" indicates that the algorithm could not find any feasible solution within the time limit.

For the sake of readability, a compact summary of the results is given in Tables 4 and 5: in the former table the instance are grouped by family, while in the latter table the instances are grouped by number of taxa. In both Tables 4 and 5 we report the number of instances for which BMEPsolver performed better than Pardi's algorithm, separating the cases in which at least one or neither of the algorithms could terminate within the time limit. The reason for this distinction is the following: when at least one of the algorithms terminated within the time limit, the performance of an algorithm is considered better than that of the other if it could find the optimal solution faster, while in the other

case, an algorithm is considered better than the other if it could find a better feasible solution.

	Algorithms within time limit?		
Instance family	At least one	Neither	Total
M18	6/6	0/3	6/9
M43	3/8	1/3	4/11
M62	1/8	1/3	2/11
M82	6/6	4/5	10/11
M17	3/8	0/0	5/8
RbcL55	4/6	5/5	9/11
Rana64	0/7	2/2	2/9
Plant25	5/7	3/4	8/11
Primates12	1/3	0/0	1/3
Total	29/59	16/25	45/84

Table 4: Performance of BMEPsolver compared with Pardi's algorithm, with the results grouped by instance family. An entry of the form a/b indicates that BMEPsolver performed better than Pardi's algorithm on a instances over a total of b instances.

The results show that 50 of the 84 instances were solved to optimality by both algorithms. For 25 of these 50 instances, BMEPsolver was faster than Pardi's algorithm, while for the other 25 instances Pardi's routine was faster. For 9 instances, exactly one of the two algorithms could find the optimal solution within the time limit: 4 of these 9 instances were solved to optimality by BMEPsolver and the other 5 by Pardi's algorithm. Thus, if we consider these $50 + 9 = 59$ instances, we can see that neither of the two algorithms dominates the other. Finally, for the remaining 25 instances neither algorithm terminated within the time limit. It is worth noting that for 16 of these instances the best feasible solution found by Algorithm BMEPsolver was better than that found by Pardi's routine.

Table 4 highlights that for some specific families of instances, one algorithm seems to be much more effective than the other. It is natural to believe that the reason for this phenomenon lies in the properties of the dissimilarity matrices. We have observed that in some cases, e.g., when the dissimilarity matrix approaches an additive matrix (i.e., a matrix such that $d_{ij} + d_{pq} \leq \max\{d_{ip} + d_{jq}, d_{iq} + d_{jp}\}$ for all $i, j, p, q \in \Gamma$), the problem is usually more prone to be solved by Pardi's algorithm (see [4, 5, 8] for a more detailed discussion on additive matrices). However, this is not a general trend. At present, despite our efforts, it seems very hard to formalize the properties that a distance matrix must satisfy in order to be better tackled with Pardi's algorithm rather than with another approach.

We can conclude that there is no clear supremacy of one of the two algorithms, though on the instances tried BMEPsolver performs slightly better than Pardi's. Moreover, if we consider all the instances having a given number of taxa and restrict ourselves to those instances for which both algorithms could

Taxa	Algorithms within time limit? At least one	Neither	Total
10	6/9	0/0	6/9
11	5/9	0/0	5/9
12	6/9	0/0	6/9
13	4/8	0/0	4/8
14	4/8	0/0	4/8
15	3/8	0/0	3/8
16	1/5	2/3	3/8
17	0/3	4/5	4/8
18	0/0	3/7	3/7
19	0/0	5/5	5/5
20	0/0	2/5	2/5
Total	29/59	16/25	45/84

Table 5: Performance of BMEPsolver compared with Pardi's algorithm, with the results grouped by number of taxa. The meaning of the entries is as in Table 4.

terminate within the time limit, we can observe that the average running time of BMEPsolver was almost always much smaller than the average running time of Pardi's algorithm: this is shown in Table 6. There is a single exception, namely for the case of 16 taxa. However, we remark that the number of instances considered in that case is only 2, thus the average is not really relevant from a statistical point of view.

Taxa	Number of instances	Average running time (s) Pardi's algorithm	BMEPsolver
10	9	0.63	0.16
11	9	3.93	0.88
12	9	23.67	4.36
13	8	94.96	20.46
14	7	198.43	123.93
15	6	755.74	568.24
16	2	543.48	1808.32

Table 6: Average running times on instances with the same number of taxa. Only instances for which both algorithms could terminate within the time limit are considered (this never happened with more than 16 taxa).

The fact that the performance of our algorithm is at least as good as that of Pardi's is particularly remarkable if we keep in mind that in line 7 of Algorithm BMEPsolver we make use of a general routine for solving an *arbitrary* QAP problem, and that our approach can be generalized to other evaluation criteria.

4.3 Results for the parallel version of the algorithm

In Tables 9–11 in the appendix, we report on the comparison between the sequential and the parallel version of Algorithm BMEPsolver, with 8, 16 and 32 processes respectively. For both versions of the algorithm, we give the computing time (time limit set to one hour) and the number of unlabeled phylogenies analysed. The total number of unlabeled phylogenies for a given number of taxa has been provided in Table 1. The last column of Tables 9–11 gives the speed-up of the parallel implementation with respect to the sequential one. This value is computed as follows. Let t_s and t_p be the computing time for the sequential and parallel version of the algorithm respectively; also, let n_s and n_p be the number of unlabeled phylogenies analysed by the sequential and parallel version of the algorithm respectively. Then

$$\text{speed-up} = \frac{n_p/t_p}{n_s/t_s}.$$

This expression is the ratio between the number of unlabeled phylogenies analysed by the parallel algorithm in one second and the number of unlabeled phylogenies analysed by the sequential algorithm in one second. Note that when both algorithms terminate within the time limit, this ratio is just t_s/t_p, while when neither algorithm terminates within the time limit the ratio is simply n_p/n_s.

In Table 7 we give a summary of the results reported in Tables 9–11 with the instances grouped by number of taxa.

In a "perfect" parallel implementation, the speed-up should ideally be equal to the number of processes used, that is 7 (resp., 15, 31) for the parallelization with 8 (resp., 16, 32) processes. In fact, for the instances of medium size the speed-up is really close to the ideal value. This is not the case when the instances are small, as in this case the time required to set up the parallel environment is not negligible with respect to the time needed for solving the QAPs. On the other hand, when the instances are large, the number of trees analysed within the time limit by the sequential algorithm is too small to give an accurate estimation of the speed-up.

5 Conclusions and future work

In this paper, we present an innovative method for computing exact solutions for the Balanced Minimum Evolution Problem based on an explicit enumeration approach. The main idea is to exploit an efficient tree coding model and tree-isomorphism to reduce the solution space. The resulting algorithm can be seen as an interaction of a first phase in which unlabeled phylogenies are generated, and a second phase in which the Leaf-Assignment Problem (a particular instance of Quadratic Assignment Problem) is solved.

We report an extensive computational analysis on instances obtained by real datasets, providing also a comparison with our implementation of Pardi's algorithm, which is the only other exact algorithm for solving the Balanced Minimum Evolution Problem. Computational results show the effectiveness

	Average speed-up		
Taxa	8 processes	16 processes	32 processes
10	0.36	0.17	0.18
11	1.37	0.76	0.96
12	2.45	3.29	2.17
13	4.99	6.91	6.76
14	5.70	10.06	13.86
15	6.28	11.40	18.03
16	6.66	13.23	23.66
17	6.70	13.54	25.43
18	6.20	13.65	27.56
19	7.97	17.33	34.94
20	8.10	21.73	41.93

Table 7: Performance of the parallel implementation of BMEPsolver, with the instances grouped by number of taxa. The average speed-up is calculated as a geometric mean.

of the proposed method. Moreover, exploiting the underlying parallelism, we present a parallel version of the algorithm showing a linear speed-up.

As a direction for future research, we observe that since we make use of a routine that solves a general QAP, it would be interesting to refine our algorithm by developing a procedure that better tackles the Leaf-Assignment Problem (2). In other words, it is possible that better results can be achieved by exploiting the particular structure of the QAPs that we have to solve.

Furthermore, as already discussed in Section 4.2, it would be useful to understand which properties of the dissimilarity matrix make an instance particularly prone to be solved by a specific algorithm.

Finally, an interesting open question concerns the complexity of the BMEP: to date, it is not known whether this is an $\mathcal{NP}$-hard problem.

Acknowledgement

Daniele Catanzaro acknowledges support from the Belgian National Fund for Scientific Research (F.N.R.S.), of which he is "Chargé de Recherches", and the Communauté Française de Belgique – Actions de Recherche Concertées (ARC). Marco Di Summa acknowledges support from the Italian Ministry of Education, University and Research (MIUR), of which he was a post-doc under the project PROG100582 at the University of Torino (Italy). The authors also acknowledge Gheorghi Pentchev for his support to numerical experiments for the unlabeled non-isomorphic phylogeny generation.

References

[1] R. Aringhieri, P. Hansen, and F. Malucelli. Chemical trees enumeration algorithms. *4OR*, 1:67–83, 2003.

[2] D. Avis and K. Fukuda. A pivoting algorithm for convex hulls and vertex enumeration of arrangements and polyhedra. *Discrete Comput. Geometry*, 8:295–313, 1992.

[3] D. Avis and K. Fukuda. Reverse search for enumeration. *Discrete Applied Mathematics*, 65:21–46, 1996.

[4] P. Buneman. The recovery of trees from measure of dissimilarities. In F. R. Hodson, D. G. Kendall, and P. Tautu, editors, *Archaeological and Historical Science*, pages 387–395. Edinburgh University Press, Edinburgh, UK, 1971.

[5] P. Buneman. A note on the metric properties of trees. *Journal of combinatorial theory*, 17:48–50, 1974.

[6] R. E. Burkard and U. Derigs. *Assignment and matching problems: Solution methods with fortran programs*, volume 184 of *Lecture Notes in Economics and Mathematical Systems*. Springer, Berlin, 1980.

[7] R. E. Burkard, S. E. Karisch, and F. Rendl. QAPLIB – A quadratic assignment problem library. *Journal of Global Optimization*, 10(4):391–403, 1997. http://www.opt.math.tu-graz.ac.at/qaplib/.

[8] D. Catanzaro. The minimum evolution problem: Overview and classification. *Networks*, 53(2):112–125, 2009.

[9] D. Catanzaro. Estimating phylogenies from molecular data. In R. Bruni, editor, *Mathematical Approaches to Polymer Sequence Analysis and Related Problems*. Springer, New York, 2010.

[10] D. Catanzaro, R. Pesenti, and M. Milinkovitch. A non-linear optimization procedure to estimate distances and instantaneous substitution rate matrices under the GTR model. *Bioinformatics*, 22(6):708–715, 2006.

[11] E. Çela. *The Quadratic Assignment Problem*. Kluwer Academic Publishers, Boston, MA, 1997.

[12] R. Desper and O. Gascuel. Theoretical foundations of the balanced minimum evolution method of phylogenetic inference and its relationship to the weighted least-squares tree fitting. *Molecular Biology and Evolution*, 21(3):587–598, 2004.

[13] J. Felsenstein. *Inferring Phylogenies*. Sinauer Associates, Sunderland, MA, 2004.

[14] O. Gascuel. *Mathematics of Evolution and Phylogeny*. Oxford University Press, New York, 2005.

[15] P. Hansen, B. Jaumard, C. Labatteux, and M. Zeng. Coding chemical trees with the centered n-tuple code. *Journal of Chemical Information and Computer Science*, 34:782–790, 1994.

[16] P. H. Harvey, A. J. L. Brown, J. M. Smith, and S. Nee. *New uses for new phylogenies*. Oxford University Press, Oxford, UK, 1996.

[17] V. Kvasnička and J. Pospichal. Canonical indexing and constructive enumeration of molecular graphs. *Journal of Chemical Information and Computer Science*, 56:1777–1802, 1991.

[18] C. Lanave, G. Preparata, C. Saccone, and G. Serio. A new method for calculating evolutionary substitution rates. *Journal of Molecular Evolution*, 20:86–93, 1984.

[19] L. Pachter and B. Sturmfels. The mathematics of phylogenomics. *SIAM Review*, 49(1):3–31, 2007.

[20] F. Pardi. *Algorithms on Phylogenetic Trees*. PhD thesis, University of Cambridge, UK, 2009.

[21] Y. Pauplin. Direct calculation of a tree length using a distance matrix. *Journal of Molecular Evolution*, 51:41–47, 2000.

[22] R. C. Read. The coding of various kind of unlabelled trees. In R. C. Read and C. Berge, editors, *Graph Theory and Computing*, pages 153–182. Academic Press: New York, 1972.

[23] F. Rodriguez, J. L. Oliver, A. Marin, and J. R. Medina. The general stochastic model of nucleotide substitution. *Journal of Theoretical Biology*, 142:485–501, 1990.

[24] C. Semple and M. Steel. Cyclic permutations and evolutionary trees. *Advances in Applied Mathematics*, 32(4):669–680, 2004.

[25] M. Snir, S. Otto, S. Huss-Lederman, D. Walker, and J. Dongarra. *MPI-The Complete Reference, Volume 1: The MPI Core*. MIT Press, Cambridge, MA, USA, 1998.

[26] N. Trinajstić, S. Nicolić, J. V. Knop, W. R. Müller, and K. Szymanski. *Computational Chemical Graph Theory*. Ellis Horwood: New York, 1991.

[27] P. J. Waddell and M. A. Steel. General time-reversible distances with unequal rates across sites: Mixing gamma and inverse gaussian distributions with invariant sites. *Molecular Phylogenetics and Evolution*, 8:398–414, 1997.

Appendix. Detailed computational results

We conclude the paper with the detailed computational results for the experiments described in Section 4. Tables summarizing the results presented here have been given in Section 4.

		Pardi's algorithm		BMEPsolver	
Instance	Taxa	Best solution	Time (s)	Best solution	Time (s)
M18	10	0.185870	0.53	0.185870	0.15
	11	0.193819	8.42	0.193819	0.58
	12	0.196762	40.43	0.196762	4.81
	13	0.214613	126.45	0.214613	28.68
	14	0.219707	558.03	0.219707	258.38
	15	0.236396	3600.00	0.236365	1866.58
	16	0.244174	3600.00	0.246283	3600.00
	17	0.247570	3600.00	0.251217	3600.00
	18	0.253569	3600.00	0.260519	3600.00
M43	10	0.102559	0.27	0.102559	0.08
	11	0.102189	0.81	0.102189	0.47
	12	0.106037	2.33	0.106037	1.86
	13	0.109305	7.21	0.109305	7.76
	14	0.110411	23.88	0.110411	43.40
	15	0.121009	195.98	0.121009	205.03
	16	0.125425	681.44	0.125425	1276.08
	17	0.128164	1678.14	0.127963	3600.00
	18	0.134976	3600.00	0.136425	3600.00
	19	0.137392	3600.00	0.137212	3600.00
	20	0.150177	3600.00	0.150634	3600.00
M62	10	0.160035	0.06	0.160035	0.04
	11	0.171106	0.18	0.171106	0.45
	12	0.184038	2.49	0.184038	3.80
	13	0.192964	2.05	0.192964	15.15
	14	0.204197	59.46	0.204197	104.69
	15	0.218539	259.43	0.218539	892.44
	16	0.228587	440.48	0.228262	3600.00
	17	0.238747	1637.56	0.239767	3600.00
	18	0.253518	3600.00	0.254847	3600.00
	19	0.277959	3600.00	0.276513	3600.00
	20	0.284730	3600.00	N/A	3600.00
M82	10	0.051924	2.43	0.051924	0.26
	11	0.051974	19.64	0.051974	2.30
	12	0.055136	120.40	0.055136	7.89
	13	0.066300	504.59	0.066300	39.62
	14	0.075609	3600.00	0.075548	424.19
	15	0.078151	3600.00	0.076740	2273.08
	16	0.078149	3600.00	0.077404	3600.00
	17	0.081023	3600.00	0.079842	3600.00
	18	0.082674	3600.00	0.081771	3600.00
	19	0.084128	3600.00	0.081500	3600.00
	20	0.085289	3600.00	N/A	3600.00

		Pardi's algorithm		BMEPsolver	
Instance	Taxa	Best solution	Time (s)	Best solution	Time (s)
M17	10	0.102670	0.08	0.102670	0.26
	11	0.127555	1.68	0.127555	1.82
	12	0.132234	25.15	0.132234	5.79
	13	0.144238	59.14	0.144238	27.47
	14	0.146977	146.15	0.146977	141.45
	15	0.152536	580.93	0.152536	1089.49
	16	0.156030	215.78	0.156140	3600.00
	17	0.158601	411.28	0.161000	3600.00
RbcL55	10	0.148909	2.18	0.148909	0.47
	11	0.160471	3.00	0.160471	1.46
	12	0.167474	14.14	0.167474	8.43
	13	0.183456	54.68	0.183456	29.43
	14	0.188897	213.84	0.188897	252.10
	15	0.196785	662.78	0.196785	800.87
	16	0.206987	3600.00	0.204931	3600.00
	17	0.213336	3600.00	0.212163	3600.00
	18	0.215366	3600.00	0.215190	3600.00
	19	0.221125	3600.00	0.215053	3600.00
	20	0.225472	3600.00	0.217774	3600.00
Rana64	10	0.040257	0.02	0.040257	0.07
	11	0.042560	0.20	0.042560	0.22
	12	0.044688	0.17	0.044688	2.07
	13	0.046716	4.36	0.046716	7.72
	14	0.047230	9.37	0.047230	39.51
	15	0.048923	37.96	0.048923	188.05
	16	0.050213	405.52	0.050213	2340.55
	17	0.051464	3600.00	0.051066	3600.00
	18	0.058981	3600.00	0.058455	3600.00
Plant25	10	0.089314	0.03	0.089314	0.04
	11	0.100708	1.32	0.100708	0.22
	12	0.104447	7.63	0.104447	1.14
	13	0.115207	1.17	0.115207	7.87
	14	0.117750	378.26	0.117750	27.96
	15	0.124688	2797.38	0.124688	233.53
	16	0.129360	3600.00	0.127068	1406.77
	17	0.134997	3600.00	0.132244	3600.00
	18	0.137926	3600.00	0.139882	3600.00
	19	0.145964	3600.00	0.141571	3600.00
	20	0.156886	3600.00	0.153756	3600.00
Primates12	10	0.122039	0.05	0.122039	0.05
	11	0.162517	0.13	0.162517	0.40
	12	0.195945	0.29	0.195945	3.48

Table 8: Performance of BMEPsolver compared with Pardi's algorithm.[a]

[a]Experiments run on a machine with 2.4 GHz CPU and 512 MB of RAM.

Instance	Taxa	Sequential Time (s)	Sequential Trees	Parallel Time (s)	Parallel Trees	Speed-up
M18	10	0.08	11	0.51	11	0.16
	11	0.34	18	0.16	18	2.14
	12	2.81	37	0.93	37	3.03
	13	16.94	66	3.07	66	5.52
	14	160.69	135	24.79	135	6.48
	15	1119.65	265	163.80	265	6.84
	16	3600.00	360	698.09	552	7.91
	17	3600.00	187	2451.18	1132	8.89
	18	3600.00	43	3600.00	360	8.37
M43	10	0.04	11	0.11	11	0.35
	11	0.27	18	0.16	18	1.66
	12	1.15	37	0.77	37	1.50
	13	4.82	66	1.07	66	4.50
	14	25.68	135	4.51	135	5.69
	15	129.43	265	19.52	265	6.63
	16	764.86	552	112.45	552	6.80
	17	3600.00	1117	567.73	1132	6.43
	18	3600.00	550	1995.69	2410	7.90
	19	3600.00	162	3600.00	1359	8.39
	20	3600.00	70	3600.00	660	9.43
M62	10	0.02	11	0.15	11	0.13
	11	0.27	18	0.30	18	0.91
	12	2.32	37	1.16	37	2.00
	13	9.43	66	2.59	66	3.65
	14	62.20	135	14.34	135	4.34
	15	534.15	265	103.09	265	5.18
	16	3041.64	552	513.36	552	5.92
	17	3600.00	251	2447.98	1132	6.63
	18	3600.00	100	3600.00	483	4.83
	19	3600.00	34	3600.00	246	7.24
	20	3600.00	1	3600.00	21	21.00
M82	10	0.15	11	0.14	11	1.10
	11	1.34	18	0.29	18	4.68
	12	4.63	37	1.27	37	3.64
	13	23.51	66	3.70	66	6.35
	14	267.33	135	40.20	135	6.65
	15	1376.73	265	214.31	265	6.42
	16	3600.00	447	674.36	552	6.59
	17	3600.00	162	3600.00	938	5.79
	18	3600.00	62	3600.00	372	6.00
	19	3600.00	8	3600.00	93	11.63
	20	3600.00	0	3600.00	12	—

Instance	Taxa	Sequential Time (s)	Sequential Trees	Parallel Time (s)	Parallel Trees	Speed-up
M17	10	0.15	11	0.25	11	0.60
	11	1.06	18	0.51	18	2.10
	12	3.40	37	1.25	37	2.71
	13	17.21	66	2.80	66	6.14
	14	84.50	135	12.74	135	6.63
	15	653.38	265	99.67	265	6.56
	16	2848.22	552	412.59	552	6.90
	17	3600.00	179	2844.84	1132	8.00
RbcL55	10	0.27	11	0.13	11	2.13
	11	0.84	18	0.74	18	1.13
	12	5.22	37	1.45	37	3.59
	13	18.54	66	3.05	66	6.08
	14	151.61	135	24.18	135	6.27
	15	483.74	265	73.76	265	6.56
	16	2830.02	552	453.86	552	6.24
	17	3600.00	295	2529.74	1132	5.46
	18	3600.00	69	3600.00	428	6.20
	19	3600.00	13	3600.00	92	7.08
	20	3600.00	4	3600.00	17	4.25
Rana64	10	0.04	11	0.22	11	0.18
	11	0.13	18	0.50	18	0.26
	12	1.23	37	0.59	37	2.08
	13	4.58	66	0.97	66	4.73
	14	24.95	135	5.03	135	4.96
	15	119.79	265	19.45	265	6.16
	16	1486.39	552	230.06	552	6.46
	17	3600.00	581	1353.43	1132	5.18
	18	3600.00	420	3600.00	1712	4.08
Plant25	10	0.02	11	0.09	11	0.22
	11	0.13	18	0.13	18	1.01
	12	0.66	37	0.30	37	2.19
	13	4.94	66	1.30	66	3.79
	14	17.62	135	3.48	135	5.06
	15	140.10	265	23.27	265	6.02
	16	901.42	552	135.86	552	6.63
	17	3600.00	504	991.00	1132	8.16
	18	3600.00	130	3600.00	949	7.30
	19	3600.00	48	3600.00	309	6.44
	20	3600.00	9	3600.00	46	5.11
Primates12	10	0.03	11	0.11	11	0.27
	11	0.24	18	0.13	18	1.84
	12	2.10	37	0.95	37	2.20

Table 9: Comparison between sequential and parallel version ($7+1$ processes) of Algorithm BMEPsolver.[a]

[a]Experiments run on a cluster with 18 4-ways nodes AMD Opteron 2.2 GHz CPU single core and 2 GB of RAM, and 37 2-ways nodes with AMD Opteron 2.2 GHz CPU dual core with 4 GB of RAM

Instance	Taxa	Sequential Time	Sequential Trees	Parallel Time	Parallel Trees	Speed-up
M18	10	0.08	11	0.23	11	0.35
	11	0.34	18	0.32	18	1.07
	12	2.81	37	0.62	37	4.56
	13	16.94	66	2.00	66	8.45
	14	160.69	135	12.76	135	12.59
	15	1119.65	265	78.84	265	14.20
	16	3600.00	360	337.96	552	16.33
	17	3600.00	187	1155.16	1132	18.87
	18	3600.00	43	3600.00	846	19.67
M43	10	0.04	11	0.23	11	0.17
	11	0.27	18	0.31	18	0.86
	12	1.15	37	0.47	37	2.44
	13	4.82	66	0.87	66	5.56
	14	25.68	135	2.36	135	10.88
	15	129.43	265	10.33	265	12.53
	16	764.86	552	57.14	552	13.38
	17	3600.00	1117	306.20	1132	11.91
	18	3600.00	550	943.11	2410	16.73
	19	3600.00	162	3600.00	3001	18.52
	20	3600.00	70	3600.00	1441	20.59
M62	10	0.02	11	0.38	11	0.05
	11	0.27	18	0.39	18	0.70
	12	2.32	37	0.80	37	2.91
	13	9.43	66	2.53	66	3.73
	14	62.20	135	10.24	135	6.08
	15	534.15	265	63.32	265	8.44
	16	3041.64	552	279.05	552	10.90
	17	3600.00	251	1288.24	1132	12.60
	18	3600.00	100	3600.00	1252	12.52
	19	3600.00	34	3600.00	457	13.44
	20	3600.00	1	3600.00	70	70.00
M82	10	0.15	11	0.46	11	0.32
	11	1.34	18	0.32	18	4.16
	12	4.63	37	0.63	37	7.35
	13	23.51	66	2.20	66	10.67
	14	267.33	135	20.16	135	13.26
	15	1376.73	265	106.97	265	12.87
	16	3600.00	447	323.24	552	13.75
	17	3600.00	162	2153.08	1132	11.68
	18	3600.00	62	3600.00	696	11.23
	19	3600.00	8	3600.00	240	30.00
	20	3600.00	0	3600.00	26	—

Instance	Taxa	Sequential Time	Sequential Trees	Parallel Time	Parallel Trees	Speed-up
M17	10	0.15	11	0.30	11	0.50
	11	1.06	18	0.71	18	1.50
	12	3.40	37	0.75	37	4.53
	13	17.21	66	1.76	66	9.78
	14	84.50	135	7.35	135	11.50
	15	653.38	265	50.72	265	12.88
	16	2848.22	552	195.67	552	14.56
	17	3600.00	179 l	1407.57	1132	16.17
RbcL55	10	0.27	11	0.59	11	0.46
	11	0.84	18	0.74	18	1.14
	12	5.22	37	1.49	37	3.51
	13	18.54	66	2.24	66	8.28
	14	151.61	135	13.40	135	11.32
	15	483.74	265	37.45	265	12.92
	16	2830.02	552	239.53	552	11.82
	17	3600.00	295	1217.79	1132	11.34
	18	3600.00	69	3600.00	858	12.43
	19	3600.00	13	3600.00	213	16.38
	20	3600.00	4	3600.00	48	12.00
Rana64	10	0.04	11	0.18	11	0.22
	11	0.13	18	0.62	18	0.21
	12	1.23	37	0.60	37	2.04
	13	4.58	66	0.88	66	5.18
	14	24.95	135	2.96	135	8.42
	15	119.79	265	14.61	265	8.20
	16	1486.39	552	118.55	552	12.54
	17	3600.00	581	646.52	1132	10.85
	18	3600.00	420	2542.42	2410	8.13
Plant25	10	0.02	11	0.31	11	0.06
	11	0.13	18	0.63	18	0.21
	12	0.66	37	0.42	37	1.57
	13	4.94	66	0.74	66	6.66
	14	17.62	135	2.03	135	8.66
	15	140.10	265	12.97	265	10.80
	16	901.42	552	67.80	552	13.29
	17	3600.00	504	472.93	1132	17.10
	18	3600.00	130	3537.49	2410	18.87
	19	3600.00	48	3600.00	613	12.77
	20	3600.00	9	3600.00	116	12.89
Primates12	10	0.03	11	0.97	11	0.03
	11	0.24	18	0.56	18	0.43
	12	2.10	37	0.57	37	3.72

Table 10: Comparison between sequential and parallel version ($15 + 1$ processes) of Algorithm BMEPsolver.[a]

[a]Experiments run on a cluster with 18 4-ways nodes AMD Opteron 2.2 GHz CPU single core and 2 GB of RAM, and 37 2-ways nodes with AMD Opteron 2.2 GHz CPU dual core with 4 GB of RAM

Instance	Taxa	Sequential Time	Sequential Trees	Parallel Time	Parallel Trees	Speed-up
M18	10	0.08	11	0.39	11	0.20
	11	0.34	18	0.51	18	0.66
	12	2.81	37	0.82	37	3.42
	13	16.94	66	1.50	66	11.28
	14	160.69	135	7.21	135	22.28
	15	1119.65	265	42.18	265	26.54
	16	3600.00	360	174.04	552	31.72
	17	3600.00	187	568.33	1132	38.34
	18	3600.00	43	3600.00	2052	47.72
M43	10	0.04	11	0.23	11	0.17
	11	0.27	18	0.33	18	0.81
	12	1.15	37	0.81	37	1.42
	13	4.82	66	1.37	66	3.51
	14	25.68	135	1.80	135	14.24
	15	129.43	265	6.12	265	21.15
	16	764.86	552	32.27	552	23.70
	17	3600.00	1117	185.65	1132	19.65
	18	3600.00	550	466.70	2410	33.80
	19	3600.00	162	3156.28	5098	35.89
	20	3600.00	70	3599.95	3222	46.03
M62	10	0.02	11	0.53	11	0.04
	11	0.27	18	0.40	18	0.68
	12	2.32	37	1.46	37	1.58
	13	9.43	66	2.62	66	3.60
	14	62.20	135	8.94	135	6.96
	15	534.15	265	47.84	265	11.17
	16	3041.64	552	194.27	552	15.66
	17	3600.00	251	739.70	1132	21.95
	18	3600.00	100	3532.88	2410	24.56
	19	3600.00	34	3599.39	1134	33.36
	20	3600.00	1	3600.00	141	141.00
M82	10	0.15	11	0.23	11	0.67
	11	1.34	18	0.40	18	3.39
	12	4.63	37	0.81	37	5.71
	13	23.51	66	1.92	66	12.26
	14	267.33	135	11.41	135	23.43
	15	1376.73	265	61.29	265	22.46
	16	3600.00	447	163.40	552	27.21
	17	3600.00	162	1089.00	1132	23.10
	18	3600.00	62	3600.00	1252	20.19
	19	3600.00	8	3600.00	460	57.50
	20	3600.00	0	3600.00	65	—

Instance	Taxa	Sequential Time	Sequential Trees	Parallel Time	Parallel Trees	Speed-up
M17	10	0.15	11	0.25	11	0.61
	11	1.06	18	0.45	18	2.34
	12	3.40	37	1.28	37	2.66
	13	17.21	66	1.58	66	10.91
	14	84.50	135	5.34	135	15.82
	15	653.38	265	31.11	265	21.00
	16	2848.22	552	100.43	552	28.36
	17	3600.00	179	759.94	1132	29.96
RbcL55	10	0.27	11	0.71	11	0.38
	11	0.84	18	0.40	18	2.08
	12	5.22	37	1.87	37	2.79
	13	18.54	66	1.67	66	11.10
	14	151.61	135	8.59	135	17.64
	15	483.74	265	20.50	265	23.60
	16	2830.02	552	141.43	552	20.01
	17	3600.00	295	624.38	1132	22.12
	18	3600.00	69	3600.00	1760	25.51
	19	3600.00	13	3600.00	371	28.54
	20	3600.00	4	3600.00	82	20.50
Rana64	10	0.04	11	0.24	11	0.16
	11	0.13	18	0.33	18	0.40
	12	1.23	37	0.89	37	1.37
	13	4.58	66	1.20	66	3.82
	14	24.95	135	2.50	135	9.98
	15	119.79	265	12.43	265	9.64
	16	1486.39	552	66.53	552	22.34
	17	3600.00	581	344.06	1132	20.39
	18	3600.00	420	1295.60	2410	15.94
Plant25	10	0.02	11	0.21	11	0.10
	11	0.13	18	0.34	18	0.38
	12	0.66	37	0.74	37	0.90
	13	4.94	66	0.91	66	5.42
	14	17.62	135	1.86	135	9.48
	15	140.10	265	8.44	265	16.60
	16	901.42	552	37.21	552	24.23
	17	3600.00	504	238.36	1132	33.92
	18	3600.00	130	1797.32	2410	37.13
	19	3600.00	48	3600.00	1273	26.52
	20	3600.00	9	3600.00	209	23.22
Primates12	10	0.03	11	0.57	11	0.05
	11	0.24	18	0.33	18	0.72
	12	2.10	37	0.81	37	2.60

Table 11: Comparison between sequential and parallel version ($31 + 1$ processes) of Algorithm BMEPsolver.[a]

[a]Experiments run on a cluster with 18 4-ways nodes AMD Opteron 2.2 GHz CPU single core and 2 GB of RAM, and 37 2-ways nodes with AMD Opteron 2.2 GHz CPU dual core with 4 GB of RAM