Comp. Appl. Math. (2018) 37:5681–5701
https://doi.org/10.1007/s40314-018-0638-y

PULLPRU: a practical approach to estimate phylogenies from single nucleotide polymorphism haplotypes under the maximum parsimony criterion

R. Feizabadi[1] · **M. Bagherian**[1] · **H. R. Vaziri**[2] · **M. Salahi**[1]

Received: 5 April 2017 / Revised: 29 April 2018 / Accepted: 2 May 2018 / Published online: 4 June 2018

Abstract Phylogeny estimation has been the subject of several researches due to its significant importance and numerous applications. The aim of this research is to study the phylogeny estimation from Single Nucleotide Polymorphism (SNP) haplotypes under the maximum parsimony criterion (MPPEP-SNP). Previous exact methods have modeled the mentioned problem as a Mixed Integer Programming (MIP) problem. Since the problem, in general, proved to be NP-hard which causes MIP models to take long runtime for solving large-scale instances, the need for non-exact methods is obvious. In this paper, the authors propose a heuristic algorithm that attempts to find the MPPEP-SNP solution in several stages by solving a specific MIP model in each stage. Created based on network flows formulation, MIP models appearing in each stage are very small; therefore, their exact solutions could be found practically very fast. In order to evaluate the performance of the proposed algorithm, it has been tested on both simulated and real instances and compared with Pars and Flow-RM as two of the best known methods. Our numerical experiments show that the proposed method can compete with the previous methods in terms of accuracy, runtime, and specially the largeness of the solved instances.

Keywords Phylogeny estimation · Haplotype · Maximum parsimony · Computational biology · Greedy algorithm · Graph · Network flows · Mixed integer programming

Mathematics Subject Classification 90C11 · 90C35 · 90C90

Communicated by Joerg Fliege.

✉ M. Bagherian
mbagherian@guilan.ac.ir

1 Department of Applied Mathematics, Faculty of Mathematical Science, University of Guilan, Rasht, Iran

2 Department of Biology, Faculty of Science, University of Guilan, Rasht, Iran

1 Introduction

According to the theory of evolution, all organisms are inter-related through a common ancestry. The genetic relationship among species could be indicated by a large evolutionary tree called the Tree of life. Living organisms, called taxon, are the leaves of the tree, and if we look back to the root of the tree, we can find their ancestors which lived hundreds of million years ago. The edges of the tree represent the estimated evolutionary relationships, and the weight of each edge represents the similarity between the two species or transformation intervals (Felsenstein 2004).

Phylogeny estimation has emerged at the time of Darwin (Gascuel 2005). In the past, biologists used to hire morphological data (phenotypes), such as beak shape, the existence or non-existence of wings, etc. to produce evolutionary trees (Semple and Steel 2003). Then, molecular data such as DNA sequences, or RNA or protein sequences, were used to do so. Recently, haplotype of a common area of a gene is used for such a purpose (Catanzaro et al. 2013; Sridhar et al. 2008).

Construction of phylogeny has many applications in various research fields such as biosystematics, medical research, drug discovery, and population dynamics (Bruni 2010). Some applications are predicting the background of human influenza A, understanding the relationships between the virulence and the genetic evolution of HIV, identifying emerging viruses such as SARS, recreating and investigating ancestral proteins, designing neuropeptides causing smooth muscle contraction, relating geographical patterns to macro evolutionary processes; in addition, in disease association studies, reconstructing tumor phylogenies, etc. (Bruni 2010).

Previous researchers suggested several criteria in order to select one phylogeny out of the possible phylogenies (Felsenstein 2004). Parsimony, likelihood, and distance-based are three important criteria that have been considered in several previous studies (Gascuel 2005). Most of these studies modeled the problem as an optimization problem (Gascuel 2005). Among the earliest studies carried out on reconstruction of the phylogenetic tree are Hein (1990, 1993). The first introduced criterion was maximum parsimony (Sforza and Edwards 1964) which is a simple non-parametric criterion, unlike maximum likelihood models (Felsenstein 2004). Maximum parsimony plays a significant role and has a great influence among the algorithms of phylogeny reconstruction (Camin and Sokal 1965; Zheng and Zheng 2015) and has been used in other methods of phylogenetic network studies (Brooks et al. 2007; Jin et al. 2007). Based on this criterion, a species is extracted from other species by minimum mutations (Semple and Steel 2003). The best phylogeny is the one which includes the least number of changes; in other words, in optimal phylogeny, weighted summation of edges in each path from one taxon to another is minimum (Graham and Foulds 1982). More specifically, phylogeny H is optimal if it has the following two conditions:

1. having the shortest length,
2. for every pair of distinct haplotypes (h_i, h_j), the length of the path from h_i to h_j is not less than the number of differences between them (Catanzaro 2011).

Finding phylogeny that meets the above-mentioned two conditions is known as MPPEP.[1] Some of these problems can be solved in polynomial time (Ding et al. 2006; Bonizzoni 2007); the most important of them is the perfect phylogeny in which each site mutates only once in the entire tree (Agarwala and Fernández-Baca 1994; Gusfield 1991; Kannan and Warnow, 1997). However, in general, it is proved that this problem is NP-hard (Foulds and Graham

[1] Most Parsimonious Phylogeny Estimation Problem.

1982; Garey and Johnson 1979). This motivates the need for the development of the exact and non-exact methods for perfect phylogeny problem (Felsenstein 2004; Catanzaro 2009). Some exact methods model the problem as an Integer Programming (IP) problem (Gusfield 2003).

Several previous studies, namely Pollock et al. (2002), Zwickl and Hillis (2002), Hillis et al. (2003), Rosenberg and Kumar (2003), Hedtke et al. (2006) and Gatesy et al. (2007) examined the effect of increasing the number of characters as well as the increase of the number of taxa on the accuracy of phylogeny. The results obtained by Pollock et al. (2002) and Zwickl and Hillis (2002) indicated that increasing the number of taxa is more effective than increasing the characters (Heath et al. 2008). On the other hand, in some applications, it happens that the number of taxa is much bigger than the number of characters in real instances. In this paper, we construct the phylogeny on instances with up to more than 1000 taxa and 14 characters in less than 2 min.

A recent version of MPPEP using SNPs to draw phylogenies is known as MPPEP-SNP. Sridhar et al. (2007) modeled this problem in the form of an IP. They have introduced a polynomial model as well as an exponential model for this problem in Sridhar et al. (2008) and reported that the exponential model is more efficient than the polynomial model in practice. Misra et al. introduced another IP model for MPPEP-SNP (Misra et al. 2011). Afterward, Catanzaro et al. introduced a new formulation as well as a series of valid inequalities which led to the gap reduction between the optimal solution and its non-integral linear programing bound (Catanzaro et al. 2013).

In this paper, we introduce a heuristic algorithm for the MPPEP-SNP which solves the problem in several stages by solving a small MIP at each stage. The proposed algorithm is compared with the MIP model of Catanzaro et al. (2013) as well as to Pars heuristic method on both simulated and real data. In Sect. 2 of this paper, the MPPEP-SNP is formulated. Then, the models presented by Sridhar et al. and Catanzaro et al. are briefly described. In Sect. 3, we introduce a new algorithm for solving MPPEP-SNP, and the fourth section deals with some numerical experiments in order to investigate the efficiency of the proposed method. Finally, the fifth section ends the paper with some concluding remarks and future research directions.

2 Problem modeling

Consider a set H consisting of n haplotypes. Each haplotype comprises m SNPs and is corresponding to one of the recent species. The purpose is to build a phylogeny which has the following two conditions:

1. it has the shortest length;
2. for every pair of distinct haplotypes (h_i, h_j), the length of the path from h_i to h_j should not be less than the number of differences between their SNPs (Catanzaro 2011).

In other words, to construct a phylogeny, two species are adjacent if they have only one difference in their corresponding haplotypes. According to the definition of adjacency, probably, there will be a forest after the constitution of phylogeny. Differently stated, the resulting graph may be disconnected because some intermediate species were extinct or not included in the species under our investigation. Parsimony criterion searches the minimum number of haplotypes that lead to the connectivity of the corresponding phylogeny after adding them to H. This is known as MPPEP-SNP. We call the existing haplotypes real haplotypes and the absent haplotypes that need to be added to H in order to meet the connectivity of final phylogeny, virtual haplotypes.

Since each haplotype is shown by an m-vector including merely zero and one, there are 2^m haplotypes with m columns. As a result, feasible region for the virtual haplotypes includes $2^m \backslash |H|$ members. This exponential number makes it hard to solve the problem. In other words, according to the above definition of adjacency, all real and virtual haplotypes form an m-dimensional hypercube such that each node corresponds to one haplotype.

Sridhar et al. (2008) have introduced a polynomial MIP and an exponential MIP for this problem. Despite a lot of variables and constraints, the exponential model gives better results in practice (Sridhar et al. 2008). The model is formulated in the following form:

$$
\begin{aligned}
&\min \sum_{u,v} w_{u,v} s_{u,v} \\
&s.t. \sum_{v} f^{t}_{u,v} = \sum_{v} f^{t}_{v,u} \quad \text{for all } t \in H\backslash\{r\}, u \notin H \qquad (1.1)\\
&\quad \sum_{v} f^{t}_{r,v} = 1 \quad \text{for all } t \in H\backslash\{r\} \qquad (1.2)\\
&\quad \sum_{v} f^{t}_{v,t} = 1\,, \sum_{v} f^{t}_{t,v} = 0 \quad \text{for all } t \in H\backslash\{r\} \qquad (1.3)\\
&\quad \sum_{v} f^{t}_{u,v} = \sum_{v} f^{t}_{v,u} \quad \text{for all } u, t \in H\backslash\{r\} : u \neq t \qquad (1.4)\\
&\quad 0 \leq f^{t}_{u,v} \leq s_{u,v} \quad \text{for all } (u,v) \in E\,,\ t \in H\backslash\{r\} \qquad (1.5)\\
&\quad s_{u,v} \in \{0,1\} \quad \text{for all } (u,v) \in E.
\end{aligned}
\tag{1}
$$

In model (1), E is the set of edges and r is the root node which can be any of the real nodes. This model looks at the problem as a multi-commodity flows problem that sends one unit of flow from root node to each destination node. All real nodes, except the root node, $(H \big/ \{r\})$, are sink nodes. There are two types of variables $f^t_{u,v}$ and $s_{u,v}$ corresponding to each edge. Real variable $f^t_{u,v}$ indicates the amount of the flow which are sent from edge (u, v) whose destination is the real node $t \in H \big/ \{r\}$; and the binary variable $s_{u,v}$ indicates the presence or the absence of edge (u, v) at optimality. Constraints set (1.1) and (1.2) are balanced constraints corresponding to virtual nodes and root node, respectively. The constraints sets (1.3) and (1.4) play the same role for the rest of the real nodes $(H \big/ \{r\})$. According to the constraints set (1.5), every edge through which a flow is sent must exist in the optimal solution. The objective function is defined in such a way that the phylogenetic tree has the minimum length. The weight of edge (u, v) is $w_{u,v}$ which is equal to 1 for each edge in MPPEP-SNP.

Catanzaro et al. (2013) presented the Binary Integer Programming (BIP) model (2) for MPPEP-SNP.

They considered an upper bound UB and a lower bound LB on the number of virtual nodes (virtual haplotypes) in their model. Based on UB and LB, the following sets appear:

$$
\begin{aligned}
V_H &= \{1, 2, \ldots, n\}\\
V_{H'} &= \{n+1, n+2, \ldots, n+UB\}\\
Q &= \{1, 2, \ldots, n+LB\}\\
R &= \{n+LB+1, n+LB+2, \ldots, n+UB\}\\
C_H &= \{i \in C : i \in V_H\} \text{ for all } C \subset V.
\end{aligned}
$$

They have considered virtual haplotype's SNPs as binary variables whose number and amount would be determined after running the model. If the ith haplotype exists at optimality, u_i equals 1, and equals zero otherwise. The number of differences in sth SNP of the two adjacent haplotypes i and j is characterized by z_{ij}^s. The set of all variables z_{ij}^s is denoted by Z. The amount of sth SNP of ith haplotype equals to $h_i(s)$. The set of all columns after applying preprocessing techniques described in Sridhar et al. (2008) is $\hat{S}$. Weights w^s, in objective function, are the number of identical columns in input data before processing in which $s \in \hat{S}$. The interested reader is referred to Catanzaro et al. (2013) for more details.

$$
\begin{aligned}
\min \; & \sum_{\substack{i,j \in V \\ i,j \in Z}} \sum_{s \in \hat{S}} w^s z_{ij}^s \\
s.t. \; & x_i^s \le u_i \quad \forall \mathrm{s} \in \hat{S}, i \in \mathrm{R} \\
& \sum_{\substack{s' \in \hat{S} \\ s' \neq s}} z_{ij}^{s'} + \mathrm{h}_i(\mathrm{s}) + x_j^s \le \quad \forall \mathrm{s} \in \hat{S}, i \in \mathrm{V}_H, \mathrm{j} \in \mathrm{V}_{H'} \\
& \sum_{\substack{s' \in \hat{S} \\ s' \neq s}} z_{ij}^{s'} - \mathrm{h}_i(\mathrm{s}) + x_j^s \le 1 \quad \forall \mathrm{s} \in \hat{S}, i \in \mathrm{V}_H, \mathrm{j} \in \mathrm{V}_{H'} \\
& \sum_{\substack{s' \in \hat{S} \\ s' \neq s}} z_{ij}^{s'} + x_i^s - x_j^s \le 1 \quad \forall \mathrm{s} \in \hat{\mathrm{S}}, i, j \in \mathrm{V}_{H'} : \mathrm{i,j} \in Z \\
& \sum_{\substack{s' \in \hat{S} \\ s' \neq s}} z_{ij}^{s'} - x_i^s + x_j^s \le 1 \quad \forall \mathrm{s} \in \hat{\mathrm{S}}, i, j \in \mathrm{V}_{H'} : \mathrm{i,j} \in Z \\
& \sum_{s \in \hat{S}} z_{ij}^s \le 1 \quad \forall \mathrm{i,j} \in \mathrm{V} \backslash \mathrm{R} : \mathrm{i,j} \in Z \\
& \sum_{s \in \hat{S}} z_{ij}^s \le \mathrm{u}_i \quad \forall \mathrm{i} \backslash \mathrm{R}, \mathrm{j} \in \mathrm{V} : \mathrm{i,j} \in Z \\
& \sum_{s \in \hat{S}} z_{ij}^s \le \mathrm{u}_j \quad \forall \mathrm{j} \backslash \mathrm{R}, \mathrm{i} \in \mathrm{V} : \mathrm{i,j} \in Z \\
& \sum_{\substack{j \in V: \\ j \in Z}} \sum_{s \in \hat{S}} z_{ij}^s \ge 1 \quad \forall \mathrm{i} \in \mathrm{Q} \\
& \sum_{\substack{j \in V: \\ j \in Z}} \sum_{s \in \hat{S}} z_{ij}^s \ge u_i \quad \forall \mathrm{i} \in \mathrm{R} \\
& \sum_{\substack{i,j \in C: \\ i,j \in Z}} \sum_{s \in \hat{S}} z_{ij}^s \le \sum_{i \in C : i \in R} u_i + |\mathrm{C}_H| - 1 \quad \forall \mathrm{C} \subset \mathrm{V} : \mathrm{C} \cap V_H \neq \phi \\
& \sum_{\substack{i,j \in V: \\ i,j \in Z}} \sum_{s \in \hat{S}} z_{ij}^s = n + LB + \sum_{i \in R} u_i - 1 \\
& u_i, x_i^s, z_{ij}^s, y_{ij} \in \{0, 1\}
\end{aligned}
\qquad (2)
$$

Catanzaro et al. also introduced a large number of valid inequalities for model (2) to reduce the difference between the optimal solution and the solution obtained from the LP relaxation by cutting the feasible region.

Although these models obtain exact solutions in reasonable time for small instances, they are not usually effective and do not produce good results for large-scale instances. For example, the model of Catanzaro et al. (2013) failed to solve some large-scale real instances.

In the next section, we will present a heuristic algorithm for the MPPEP-SNP. Our algorithm obtains the solution of the main problem in several stages by a greedy selection in each stage. Each stage has two steps, in which the second one solves a small MIP model. Our experiments on simulated and real data, especially large instances, demonstrate that the new approach can compete with the previous methods in terms of accuracy and running time.

3 New algorithm

Our algorithm, PULLPRU, begins with a graph of two nodes in the first stage and extends it to the final phylogeny. In the middle stages, it is possible for a node to be consisted of a set of haplotypes; in other words, it is not the case that necessarily each node should correspond to just one haplotype. At every stage, we partition the sets until the last stage in which one haplotype remains associated with each node. The partitioning is done in m stages. First, we index haplotype columns from *1* to m. At each stage, a column of haplotypes is considered, respectively. In the first stage, we skip $m-1$ last columns and only examine the first column. This stage partitions the set of haplotypes into two subsets depending on the value of 0 or 1 in their first column. In other words, we put all haplotypes which have 0 in their first component in the first subset. So the second subset contains all the haplotypes having 1 in their first position. We plot a graph with two nodes such that each node is corresponding to one of these subsets. Since in the first stage we consider only one column, the number of different columns between the two subsets is equal to 1. Therefore, according to the above adjacency definition, we connect two nodes by one edge. Each stage contains two steps. At the first step of each stage, a new column is investigated, each node of the graph is divided into two nodes, and its corresponding set is partitioned into two subsets depending on the value of the new column in each haplotype. In other words, in the first step of ith stage, we consider ith SNP, and regarding the value of 0 or 1 of the haplotypes in this SNP, each of the sets of the corresponding nodes in the graph is partitioned into two subsets. This is equivalent to providing two copies of the graph. In the first copy, each node corresponds to a set of haplotypes which has 0 in ith component, and the second copy corresponds to sets of haplotypes with 1 in ith component.

To each copy of the graph in the previous stage, a column has been added. The number of different columns between two nodes in one copy has not changed because the added column has the same value in all nodes of the copy. Thus, it is clear that the adjacency of the nodes in each of the graph copies is exactly the same as the adjacency of those in previous stage. So if two nodes are adjacent in the previous stage graph, i.e., have one difference in their columns, they are also adjacent in each copy. For this reason, the edges' position and number do not change in each copy compared to the previous graph.

To determine the adjacency between corresponding nodes of two copies, note that every node of the graph of $i-1$th stage contains a set of haplotypes which are exactly the same in the first $i-1$ SNP. Adding column i probably partitions this set into two subsets; one containing the haplotypes with 0 at their ith component, and the other containing all haplotypes with 1

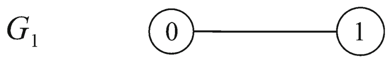


Fig. 1 Initial graph

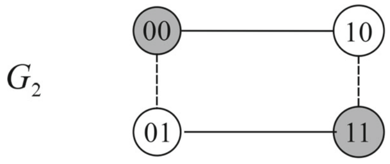


Fig. 2 The graph is constructed by duplication of G_1 and connecting their corresponding nodes in two copies

at their ith component. Therefore, some of them are placed in the first copy and the others in the second one. Now the number of distinct components between members of one subset and another is changed from zero to one which shows the adjacency between them. Therefore, we connect two corresponding nodes in two copies by an edge. We name this step as pullulation that includes duplicating the graph of the previous stage, partitioning the sets of haplotypes, and connecting two copies.

In transition from stage $i - 1$ to i, we partition the haplotypes set corresponding to each node into two subsets with different ith component. Especially in the final stages, it is very likely that one of these subsets be empty. In other words, between real haplotypes, there exists no haplotype whose first i nucleotide site corresponds to that node. In this case, it is possible to remove the empty nodes and edges emanating from them, as long as the connectivity of the under construction graph is not violated. Hence, if the number and the location of the nodes of this kind are in such a way that removing them leads to losing connectivity, we have to ignore the removal of some of these nodes and add their corresponding virtual haplotypes. According to the parsimony criteria, the number of nodes of this type should be minimized. We call this step pruning which involves removing maximum number of empty nodes, provided that the connectivity is maintained.

For more details, the first stage partitions haplotypes set into two subsets. Figure 1 depicts the corresponding graph. Node 0 contains all of the haplotypes with 0 at their first nucleotide site, and node 1 includes all the haplotypes with 1 at their first nucleotide site. We assume that none of the nodes are empty.

The pullulation step of the second stage provides two copies of the graph G_1, in one of which all haplotypes whose second nucleotide site equals to 0 are placed, and the other one contains the haplotypes whose second nucleotide site is equal to 1. Corresponding nodes in two copies are connected by an edge. These edges are plotted with dotted line in Fig. 2.

Now suppose that the two first nucleotide sites of all given haplotypes are 01 or 10, i.e., there is no haplotype with the first nucleotide site 00 or 11 among the real haplotypes. So, the corresponding sets to nodes 00 and 11 are empty. These virtual nodes are shown with gray circles in Fig. 2. If we remove both virtual nodes in pruning step, the graph which is made from merely real haplotypes has the two nodes 10 and 01 which are disconnected. To have a connected graph, it is necessary to preserve one of the nodes 00 or 11. This also occurs in the next stages. However, in the next stages, the number of virtual nodes may be so high that in order to preserve the connectivity of the graph, we have to solve an optimization problem to be able to choose the minimum number of virtual nodes according to parsimony criterion. So the pruning stage needs to solve an optimization problem. We present two different models for pruning stage as follows:

(a) We model the problem as a multi-commodity flows problem. Consider a network N corresponding to the graph G so that each edge is replaced with two arcs. Suppose V is the set of all nodes, and set A contains all arcs of N. We partition node set V into three subsets: one of the real nodes that can be selected arbitrarily is distinguished from other nodes as the source node s, other real nodes as sink nodes are placed in set C, and all of the virtual nodes in set VV. We consider one commodity corresponding to each of the nodes in C with the same index, and one unit of flow of each commodity must be sent from source node s to its corresponding sink node in C. Continuous variable f_{ij}^{k} exhibits the value of the kth commodity sent from node i to node j. Each binary variable x_j is corresponding to one of the graph's nodes and also to a set of haplotypes. If node j is present at optimality, x_j is equal to 1; otherwise it is equal to 0. According to above definitions the model is as follows:

$$
\begin{aligned}
\min \quad & \sum_{j \in V} x_j \\
s.t. \quad & f_{ij}^{k} \le x_j \quad \forall (i,j) \in A,\ \forall k \in C \qquad (3.1)\\
& \sum_{\{i:(s,i)\in A\}} f_{si}^{k} - \sum_{\{i:(i,s)\in A\}} f_{is}^{k} = 1 \quad \forall k \in C \qquad (3.2)\\
& \sum_{\{j:(j,k)\in A\}} f_{jk}^{k} - \sum_{\{j:(k,j)\in A\}} f_{kj}^{k} = 1 \quad \forall k \in C \qquad (3.3)\\
& \sum_{\{j:(j,k)\in A\}} f_{jk}^{t} - \sum_{\{j:(k,j)\in A\}} f_{kj}^{t} = 0 \quad \forall k,t \in C,\ k \ne t \quad (3.4)\\
& \sum_{\{i:(i,j)\in A\}} f_{ij}^{k} - \sum_{\{i:(j,i)\in A\}} f_{ij}^{k} = 0 \quad \forall k \in C,\ \forall j \in VV \quad (3.5)\\
& x_j \in \{0,1\} \quad \forall j \in V \\
& f_{ij}^{k} \ge 0 \quad \forall k \in C, \quad \forall (i,j) \in A.
\end{aligned} \qquad (3)
$$

The constraints set (3.1) states that if some of the commodities pass through an arc, its terminal nodes must be present in the optimal solution. Source node s supplies $|C|$ kinds of commodity to the other real nodes in set C. This is guaranteed by the constraints set (3.2). The ith real node in C demands one unit of commodity i. This demand is satisfied by the constraints set (3.3). Mass balance constraints in (3.4) manage the supply and demand for other commodities in real nodes C; similarly, constraints set (3.5) plays the same role in virtual nodes.

Sending flows from real node s to all other real nodes guarantees the existence of a route between all real nodes at optimality that means the connectivity of the optimal graph. The objective function minimizes the number of nodes in the optimal graph.

(b) Again, we model the problem as a network flows problem, but this time as a single-commodity flows problem. The definitions of node s and sets C and VV are the same as (a). In this case, source node s supplies $|C|$ units of flows, and each of the real nodes in set C demands one unit of flows. The binary variable corresponding to node j is x_j, and the continuous variable f_{ij} exhibits the amount of flows passing through arc (i, j). The model is as follows:

$$
\begin{aligned}
\min \quad & \sum_{j \in V} x_j \\
s.t. \quad & f_{ij} \le M x_j \quad \forall (i,j) \in A \qquad (4.1)\\
& \sum_{\{i:(s,i)\in A\}} f_{si} - \sum_{\{i:(i,s)\in A\}} f_{is} = |c| \qquad (4.2)
\end{aligned}
$$

$$\sum_{\{j:(k,j)\in A\}} f_{kj} - \sum_{\{j:(j,k)\in A\}} f_{jk} = -1 \quad \forall k \in C \quad (4.3)$$
$$\sum_{\{i:(i,j)\in A\}} f_{ij} - \sum_{\{i:(j,i)\in A\}} f_{ji} = 0 \quad \forall j \in VV \quad (4.4)$$
$$x_j \in \{0,1\} \quad \forall j \in V$$
$$f_{ij} \geq 0 \quad \forall (i,j) \in A. \tag{4}$$

The constraints set (4.1) guarantees that if some flows pass through an arc, its terminal nodes must be present in the optimal solution. Since x_j is binary but f_{ij} is continuous and may be greater than one, we multiply x_j by M that is sufficient to be $|C|$ which is the maximum flows in the network. The constraint (4.2) expresses that source node s supplies $|C|$ units of the flow. Constraints set (4.3) meets the demand of each real node in C. Mass balance constraints for virtual nodes are guaranteed by (4.4). Like model (3), the objective function counts the number of nodes in each solution that must be minimum at optimal solution and the length of the final phylogeny is obtained by subtracting one from the optimal value.

It is clear that the nodes containing real haplotypes must exist at optimality, and the value of their corresponding variable should be 1. Therefore, to have simpler models, we can assign 1 to these variables before implementing either of the models (3) or (4).

By eliminating the binary variables corresponding to the real nodes, the number of binary variables in models (3) and (4) is equal to the number of the virtual nodes in the current graph, and this number would not be high because the graph is pruned in each stage. Since the number of the discrete variables determines the hardness of the MIP models, the models we face in each stage have fewer discrete variables and thus run faster.

Model (3) assigns $|C|$ continuous variables to each arc; therefore, it has $|C| \times |A|$ continuous variables in general, but Model (4) assigns just one continuous variable to each arc, and it has $|A|$ continuous variables. This difference is significant because $|C|$ is approximately equal to the number of haplotypes, and some instances have thousands of haplotypes.

The number of constraints in model (3) is equal to $|C|(|V|+|A|)$ and in model (4) is equal to $|V|+|A|$.

It should be noted that the output of the algorithm may not be a tree, but this is not so important. If drawing a phylogenetic tree is desired, it can be obtained by one of the minimum spanning tree algorithms such as Kruskal or Prim. Moreover, the edges of the produced graph are weightless that provide the possibility of using more efficient algorithms to extract a minimum spanning tree. However, the bottleneck operation of the algorithm is not to extract a minimum spanning tree.

In the following, we introduce a few techniques that affect the runtime and accuracy of PULLPRU.

3.1 Prune skipping

We attempt to remove the maximum number of virtual nodes in the pruning step of each stage based on the parsimony criterion. But Pruning is done on a small graph instead of an m-dimensional hypercube that may cause the wrong removal of some of the nodes in the final phylogeny. This probability can be reduced by ignoring pruning in some stages. Therefore, the final solution would probably be more accurate if we prune the graph once in several stages instead of once in each stage. Although ignoring some of the pruning steps causes to reduce runtime, it also enlarges the graph in the next stages, and this will increase the size of the MIP model in the first stage that we have to prune which requires more time. Sometimes, especially in the final stages, these small changes in size are such that it is impossible to solve the problem in reasonable time, but using this technique in the early stages seems reasonable and can improve accuracy and runtime simultaneously. The number of virtual nodes in the graph can be a good criterion for pruning decision.

3.2 Different permutations of SNPs

As explained above, we indexed SNPs from left to right and from 1 to m, and the ith SNP is examined in the ith stage. Changing SNPs permutation before implementation of the algorithm strongly affects the accuracy and the runtime. Unfortunately, a particular change does not have the same effect on all instances, i.e., whereas one index changing improves the accuracy and runtime of one instance, it is likely to produce the opposite result on another instance. However, some permutations generally have positive effects on the accuracy and runtime. One type of permutation for columns is to have an assessment on the unexamined columns in the beginning of each stage in order to make sure which column(s) would produce a better result in the final phylogeny. This process is done by assigning scores to unexamined columns.

To score the tth column, we consider the corresponding sets of the nodes of the graphs obtained from the previous stage. For every set S_k, one unit will be added to the score of tth column if the two following condition satisfy:

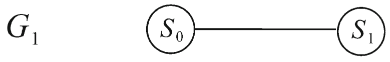
1. S_k has more than one member,
2. all the members in S_k have the same value in their tth SNPs.

For the first stage, we study the column which has the least difference between the number of its 0s and 1s, and if there are some columns of this kind having the same least difference, we choose one of them randomly. Examining columns with higher score in each stage was experimentally proved to give proper results. The reason behind such scoring system is to postpone examining the columns that lead to further expansion of the graph to the final stages. Because during the expansion of the graph, there may be some more virtual nodes to be added to the graph. Adding these nodes in the early stages will allow them to be pullulation more quickly (regardless of pruning stages, the replication rate of a virtual node added in the ith would be 2^{m-i}). Clearly, pruning stages will not prune them all. However, by adding virtual nodes in the final stages, they would be less duplicated. Therefore, it would be logical to try to impose less virtual nodes to the network in the early stages, and this would occur if we prevent the expansion of real nodes as far as possible. With regard to the scoring system defined above, if in a stage the tth column is examined and this column receives one score from S_k set, the S_k will not be partitioned. Therefore, partitioning the set S_k will be postponed, which is desirable. Therefore, in each stage, we should examine the columns with higher score.

4 Numerical experiments

To give a better description of PULLPRU, in this section, first, we solve a small example; then we discuss the results for both simulated and real large-scale instances.

Fig. 3 Graph of the first stage



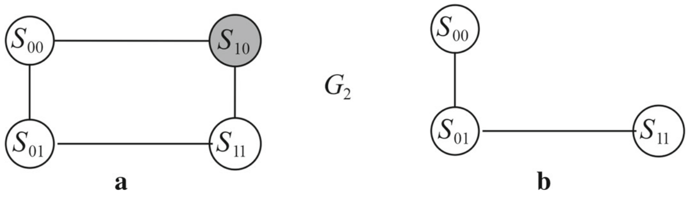


Fig. 4 Graphs of the second stage. **a** The graph obtained from pullulation step. **b** The graph obtained from pruning step

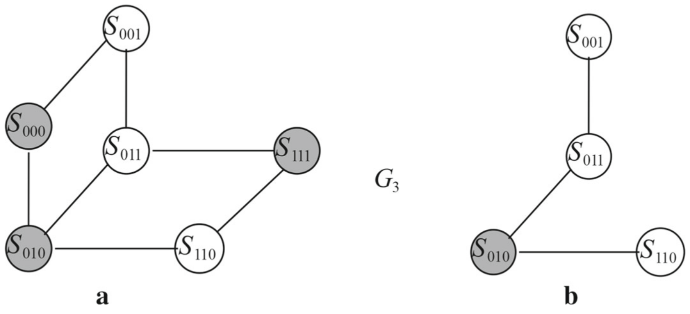


Fig. 5 Graphs of the third stage. **a** The graph obtained from pullulation step. **b** The graph obtained from pruning step

4.1 Small example

Suppose $H = \{00111, 01101, 01110, 11010\}$ is a given set of haplotypes, and estimating a phylogenetic network on H is desired. Corresponding phylogenetic networks would be built in five stages. We call the final graph of stage i, G_i and indicate the virtual nodes with gray circles in all figures.

Stage 1. The graph of the first stage is depicted in Fig. 3 where $S_0 = \{00111, 01101, 01110\}$ and $S_1 = \{11010\}$. Since none of S_0 and S_1 is empty, this stage does not need the pruning step. *Stage 2*. In the second stage, we provide two copies of G_1 and connect the corresponding nodes in the two copies to each other. Then, we partition haplotypes into four nodes according to their first and second columns properly. The resulting graph is depicted in Fig. 4.a where $S_{00} = \{00111\}$, $S_{01} = \{01101, 01110\}$, $S_{11} = \{11010\}$ and $S_{10} = \{\}$. Since S_{10} is empty, it can be removed from the graph without losing the connectivity. Therefore, we have the graph G_2 in Fig. 4b at the end of the second stage. *Stage 3*. Figure 5a shows the initial graph in the third stage obtained by duplicating G_2 and connecting the corresponding nodes in two copies where $S_{000} = \{\}$, $S_{010} = \{\}$, $S_{110} = \{11010\}$, $S_{001} = \{00111\}$, $S_{111} = \{\}$ and $S_{011} = \{01101, 01110\}$. Because sets S_{000}, S_{010} and S_{111} are all empty, we can remove their corresponding nodes from the graph. But this will cause losing the connectivity of G_3. According to parsimony criterion, we have to keep the least number of virtual haplotype(s) in the graph. From Fig. 5a, it is clear that there is no need to solve an optimization problem, and the addition of only virtual haplotype 010 meets this need. As a result, the final graph of the third stage is demonstrated in Fig. 5b. *Stage 4*. The graph obtained from the pullulation step of the fourth stage is depicted in Fig. 6a where $S_{0010} = \{\}$, $S_{0110} = \{01101\}$, $S_{0100} = \{\}$,

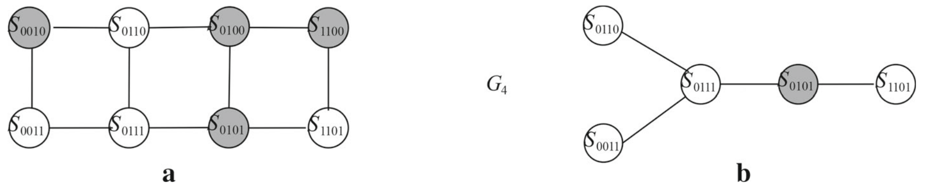


Fig. 6 Graphs of the forth stage. **a** The graph obtained from pullulation step. **b** The graph obtained from pruning step

$S_{1100} = \{\}$, $S_{0011} = \{00111\}$, $S_{0111} = \{01110\}$, $S_{0101} = \{\}$ and $S_{1101} = \{11010\}$. Since the sets S_{0010}, S_{0100}, S_{1100} and S_{0101} are empty, real haplotypes alone do not constitute a connected graph. But at the end of the fourth stage, adding only the virtual haplotype 0101 results in the connected graph demonstrated in Fig. 6b. *Stage 5*. The graph resulted from the pullulation step of the fifth stage is depicted in Fig. 7a. Except S_{00111}, S_{01101}, S_{01110}, S_{11010} that each one contains haplotypes equal to their indices, other sets are empty and their corresponding nodes have the potential to be removed from the graph. We use model (4) for pruning step in order to identify virtual haplotypes needed to establish the connectivity. Each node is labeled by a number in Fig. 7a which specifies binary variable index in the model. Real node 1, as the source node, supplies three units of flows, and every real node 2, 3 and 4 demands one unit of flow. Input flows must be equal to output flows in virtual nodes 5–10. Note that there are only 3 units of flows on the network. This number 3 specifies coefficients of x_j in the first set of constraints. Model (5) formulates the pruning step of this stage as follows:

$$\begin{aligned}
&\min \sum_{j=1}^{10} x_j \\
&s.t. \sum_{\{i:(i,j)\in A\}} f_{ij} \le 3x_j \quad j = 1, \ldots, 10 \\
&f_{15} + f_{17} - f_{51} - f_{71} = 3 \\
&f_{26} + f_{27} - f_{62} - f_{72} = -1 \\
&f_{35} + f_{36} + f_{37} + f_{39} - f_{53} - f_{63} - f_{73} - f_{93} = -1 \\
&f_{49} + f_{4,10} - f_{94} - f_{10,4} = -1 \\
&f_{51} + f_{53} - f_{15} - f_{35} = 0 \\
&f_{62} + f_{63} - f_{26} - f_{36} = 0 \\
&f_{71} + f_{72} + f_{73} + f_{78} - f_{17} - f_{27} - f_{37} - f_{87} = 0 \\
&f_{87} + f_{89} + f_{8,10} - f_{78} - f_{98} - f_{10,8} = 0 \\
&f_{93} + f_{94} + f_{98} - f_{39} - f_{49} - f_{89} = 0 \\
&f_{10,4} + f_{10,8} - f_{4,10} - f_{8,10} = 0 \\
&f_{ij} \ge 0 \quad \forall (i, j) \in A \\
&x_j \in \{0, 1\} \qquad j = 1, \ldots, 10
\end{aligned} \tag{5}$$

The optimal solution of model (5) is $x_1 = 1, x_2 = 1, x_3 = 1, x_4 = 1, x_5 = 0, x_6 = 0, x_7 = 1, x_8 = 0, x_9 = 1$ and $x_{10} = 0$.

Thus, Fig. 7.b shows the final phylogeny, where 01111 and 01010 are virtual haplotypes which have been added in order to meet connectivity.

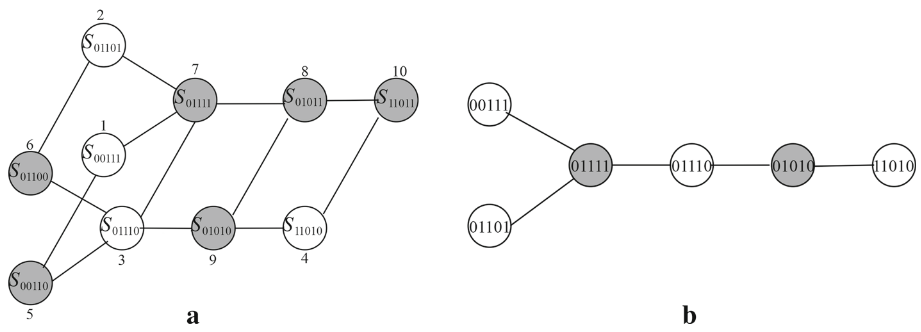


Fig. 7 Graphs of the fifth stage. **a** The graph obtained from pullulation step. **b** The final phylogeny

4.2 Experimental results

Here we compare PULLPRU with well-known Pars program of Phylip version 3.695 (Felsenstein 2005) and model (2) as one of the best exact models on several simulated and real instances. Comparison is done in terms of accuracy, runtime, and largeness of datasets.

We implemented PULLPRU in Matlab 32-bit 7.14.0.739 linked to GAMS version 24.1.2 using the MIP solver Cplex. Both PULLPRU and Pars program were run on a Core i5-2430 M, 2.40 GHz, equipped with 2.69 GByte RAM and operating system win7 32-bit. The authors of Catanzaro et al. (2013) implemented their model by means of Mosel 64-bit 3.2.0 of Xpress-MP, Optimizer version 22, running on a Pentium 4, 3.2 GHz, equipped with 2 GByte RAM and operating system Gentoo release 7 (kernel linux 2.6.17).

In Sridhar et al. (2008) the authors offered some preprocessing techniques for the MPPEP-SNP. These techniques are applied on input data before solving the problem which reduces the size of the data without losing the optimality. Thus, before running PULLPRU, we used two preprocessing techniques that will be explained in the following:

1. If there are duplicate haplotypes, we can eliminate all except one, since all of them are in one node in phylogeny which is constructed;
2. moreover, if there exists a nucleotide site with the same amount in all haplotypes which are under investigation, this nucleotide site can be eliminated. Needless to say, the corresponding nucleotide site value in the virtual haplotypes and the real haplotypes will be equal.

Interestingly, it happens sometimes that the number of haplotypes is much bigger than the number of SNPs in real instances (Catanzaro et al. 2013). They attempted to solve model (2), called Flow-RM, on such datasets. So, they simulated instances of the MPPEP-SNP having up to 300 haplotypes and 10 SNPs, and succeeded to solve Flow-RM within 3 h. However, it took less than 3 s for PULLPRU to solve on simulated instances having up to more than 1000 haplotypes and 10 SNPs.

We simulated the instances similar to Catanzaro et al. (2013). They, In order to create a haplotype which is a vector containing randomly generated 0 and 1, first determined the number of its 1's sites by generating a random number; then, randomly specified the sites of these SNPs. This process is repeated until n haplotypes of one instance are generated. We simulated 120 random instances.

Table 1 contains ten groups of datasets: each group contains ten instances, and each instance has 10 SNPs. The instances of the groups 1–10 have 100, 200, 300, 400, 500, 600, 700, 800, 900, and 1000 haplotypes, respectively (before preprocessing). We compared

Table 1 Comparison of the performance of PULLPRU with Pars on simulated data with ten SNPs

$\|H\|$	Instance	$\|H\|$ post reduction	Pars		PULLPRU		$\|H\|$	Instance	|H| post reduction	Pars		PULLPRU	
			L	Time	L	Time				L	Time	L	Time
100	1	69	78	0.29	78	1.30	200	1	123	130	1.00	128	1.46
	2	74	83	0.21	83	1.32		2	131	135	0.86	132	1.52
	3	73	83	0.15	79	1.40		3	134	139	0.73	139	1.51
	4	78	84	0.28	84	1.37		4	139	143	1.37	143	1.47
	5	76	83	0.19	83	1.35		5	122	130	1.03	127	1.43
	6	74	86	0.27	84	1.43		6	123	129	1.21	129	1.58
	7	76	87	0.25	87	1.32		7	123	129	1.13	128	1.49
	8	83	92	0.33	90	1.32		8	125	131	1.23	131	1.45
	9	78	88	0.26	86	1.31		9	125	134	1.09	132	1.56
	10	74	83	0.31	83	1.32		10	118	124	1.17	125	1.48
300	1	166	170	1.93	169	1.66	400	1	196	197	4.79	197	1.76
	2	171	173	2.86	171	1.66		2	203	205	4.80	205	1.79
	3	178	181	3.00	181	1.70		3	190	194	3.30	192	1.69
	4	160	164	2.26	164	1.61		4	193	197	3.90	195	1.73
	5	164	168	2.72	167	1.63		5	201	202	3.47	201	1.82
	6	163	167	3.32	166	1.52		6	197	197	3.22	197	1.75
	7	150	152	2.34	152	1.62		7	202	203	4.28	203	1.82
	8	163	166	2.75	166	1.70		8	201	202	3.83	202	1.90
	9	169	173	3.38	173	1.74		9	195	198	3.26	196	1.86
	10	168	167	2.77	167	1.52		10	206	208	4.37	207	1.95
500	1	231	231	5.75	231	2.00	600	1	244	244	7.23	243	2.03
	2	224	226	5.30	225	2.12		2	253	253	10.22	253	2.02

Table 1 continued

$\|H\|$	Instance	$\|H\|$ post reduction	Pars		PULLPRU		$\|H\|$	Instance	|H| post reduction	Pars		PULLPRU	
			L	Time	*L*	Time				*L*	Time	*L*	Time
	3	230	231	5.16	231	1.98		3	254	255	6.74	253	2.24
	4	231	231	6.15	231	1.96		4	246	246	8.22	245	2.00
	5	216	217	5.37	216	1.87		5	242	243	8.52	242	2.11
	6	218	220	7.39	219	1.87		6	249	251	5.71	248	1.94
	7	216	217	4.51	216	2.04		7	236	237	8.21	235	1.96
	8	214	213	6.79	213	1.84		8	248	249	6.95	248	2.08
	9	213	213	5.29	212	1.84		9	249	249	8.80	248	2.13
	10	217	218	5.97	217	1.87		10	244	244	9.41	244	2.04
700	1	263	264	13.47	263	2.21	800	1	297	296	19.32	296	2.44
	2	266	265	13.09	265	2.21		2	286	285	16.32	285	2.35
	3	277	276	11.73	276	2.24		3	289	289	13.54	289	2.31
	4	256	256	14.99	256	2.06		4	275	275	21.08	274	2.25
	5	257	256	9.89	256	2.02		5	287	287	13.58	286	2.30
	6	261	260	11.46	260	2.03		6	284	283	15.31	283	2.33
	7	263	262	16.54	262	2.09		7	278	278	15.62	277	2.37
	8	269	269	11.95	269	2.27		8	281	281	18.99	280	2.24
	9	260	260	11.80	259	2.14		9	277	280	13.67	277	2.09
	10	258	258	11.74	258	2.12		10	293	293	13.26	292	2.34
900	1	308	307	21.48	307	2.59	1000	1	313	313	29.00	312	2.46
	2	308	307	21.19	307	2.44		2	297	296	23.31	296	2.27

Table 1 continued

$\|H\|$	Instance	$\|H\|$ post reduction	Pars		PULLPRU		$\|H\|$	Instance	|H| post reduction	Pars		PULLPRU	
			L	Time	L	Time				L	Time	L	Time
	3	288	288	21.24	287	2.37		3	292	293	28.53	291	2.48
	4	285	285	28.52	284	2.13		4	289	288	37.44	288	2.34
	5	302	301	28.72	301	2.38		5	294	293	22.72	293	2.23
	6	292	293	18.15	291	2.35		6	318	317	32.26	317	2.47
	7	301	301	14.35	300	2.37		7	303	302	27.51	302	2.48
	8	298	297	24.88	297	2.46		8	306	305	22.68	305	2.33
	9	295	294	21.91	294	2.38		9	309	308	28.05	308	2.44
	10	295	295	18.12	294	2.33		10	311	310	33.59	310	2.40

Table 2 Comparison of the performance of PULLPRU with Pars on simulated data with 14 SNPs

|H|	Ins	|H| post …	Pars		PULLPRU	
			L	Time	*L*	Time
2000	1	1069	1101	477.76	1104	34.56
	2	1036	1065	347.31	1063	30.33
	3	1012	1043	494.62	1044	30.11
	4	1028	1063	516.57	1063	30.20
	5	1050	1079	428.39	1081	32.65
3000	1	1377	1393	1801.52	1391	56.80
	2	1361	1380	1598.14	1380	55.10
	3	1329	1355	1719.23	1355	52.31
	4	1349	1371	2028.80	1371	52.53
	5	1364	1380	1716.05	1381	55.72
4000	1	1637	1645	3180.52	1645	79.57
	2	1620	1626	4777.36	1626	81.97
	3	1620	1626	4703.13	1626	77.98
	4	1640	1650	3650.10	1648	80.24
	5	1620	1626	4609.73	1626	77.99
5000	1	1844	1846	6240.15	1845	102.87
	2	1867	1869	10,001.56	1867	107.19
	3	1856	1857	10,535.78	1857	105.57
	4	1844	1846	6395.93	1845	102.79
	5	1844	1846	6353.11	1845	104.36

PULLPRU with Pars program of Phylip on this dataset. In using Pars, we did not change its default settings except 'number of trees to save' which was decreased from 100 to 1 in order to have a better runtime. In implementing PULLPRU, we performed the pruning step only in stages 4, 8 and 10, using model (4).

The column '$|H|$ post reduction' shows the number of haplotypes in each instance after preprocessing. After applying the preprocessing techniques, the number of columns did not change. The column 'L' refers to the length of the obtained phylogeny, and column 'Time' reports the runtime in second for the corresponding method. In 47 instances out of 100 (i.e., in almost half of the instances) PULLPRU obtained better solution compared to Pars, while Pars obtained better result only in one instance which is instance 10 in the group of 200 haplotypes. In 20 instances of the groups of 100 and 200 haplotypes, Pars ran faster than PULLPRU, but in 80 other instances PULLPRU was superior.

Obviously, the runtime increases by increasing the instance size. The difference of the average runtime of group 1 instances and that of group 10 instances in Pars is 28.26 s while it is just 1.05 s for PULLPRU. This means that increasing the instance size has little impact on PULLPRU runtime.

Table 2 contains four groups of simulated data, each group consists of five instances, each instance have 14 SNPs, and each instance in each group consists of more than 1000 haplotypes. The initial settings of Pars and PULLPRU are the same as the initial settings of Table 1, except in implementing PULLPRU, we skipped pruning the graphs with less than 110 virtual nodes. 20 instances were studied in Table 2. In 7 instances, PULLPRU obtained

Table 3 Comparison between Pars and PULLPRU in terms of accuracy and runtime

Dataset	Input (before)	Input (after)	Pars		PULLPRU		Opt
			L	Time	*L*	Time	
Human chromosome Y	150×49	14×15	16	0.51	16	1.20	16
Bacterial mtDNA	17×1510	12×89	96	0.05	96	159.70	96
Chimpanzee mtDNA	24×1041	19×61	63	0.33	63	106.86	63
Chimpanzee chromosome Y	15×98	15×98	99	0.14	99	240.82	99
Human mtDNA	40×52	32×52	73	1.98	74	20.53	73
Human mtDNA	395×830	34×39	53	14.61	53	9.45	53
Human mtDNA	13×390	13×42	48	0.29	48	23.02	48
Human mtDNA	33×405	27×39	43	0.30	43	5.87	43

Table 4 Comparison between flow-RM, Pars and PULLPRU

Dataset	Input (before)	Input (after)	Flow-RM		Pars		PULLPRU	
			Block size	Time	*L*	Time	*L*	Time
f1	$63 \times 16{,}569$	50×234	15	10,286.1	289	9.78	289	998.11
i2	40×977	31×113	10	781.85	198	0.29	227	205.15
k3	100×757	55×110	13	588.38	228	19.22	289	426
m4	26×48	23×36	10	5	43	0.15	43	9.08
p5	$21 \times 16{,}548$	20×158	10	22,283.4	181	0.16	193	551

better solutions compared to Pars, however, the results obtained by Pars were better than PULLPRU's in 4 other instances; and in 9 instances the lengths of the obtained tree are equal in both methods. PULLPRU solved the problem in less than 2 min for all the instances, while the runtime of Pars was between 5 min and 3 h. The average runtime of each group is consistent with the results of Table 1 about the impact of size of instances on the runtime.

While implementing PULLPRU on instances of Tables 3 and 4, we used the scoring system which has been defined in Sect. 3.1 in order to determine the permutation of the columns.

Table 3 contains eight real instances of MPPEP-SNP. These instances include human chromosome Y constituted of 150 haplotypes with 49 SNPs in each; bacterial DNA constituted of 17 haplotypes with 1510 SNPs in each; chimpanzee mitochondrial DNA constituted of 24 haplotypes with 1041 SNPs in each; chimpanzee chromosome Y constituted of 15 haplotypes with 98 SNPs in each; and a set of four human mitochondrial DNA from HapMap constituted f 40 haplotypes with 52 SNPs each, 395 haplotypes with 830 SNPs in each, 13 haplotypes with 390 SNPs in each, and 33 haplotypes with 405 SNPs in each, respectively.

In Table 3, we have reported the output of PULLPRU compared with Pars program of Phylip. The second column shows the number of haplotypes and SNPs in each instance before preprocessing, and the third column shows them after preprocessing. The last column is the optimal solution. The columns ‘L’ and ‘Time’ report the length and runtime of phylogenetic tree obtained from each method. As we see, PULLPRU has gained the exact solution in all instances, except in the fifth instance, in which it has one difference between the optimal solution and the length of obtained phylogeny. Pars gained the exact solution in all instances. In the sixth instance, PULLPRU ran faster than Pars, but in other instances Pars had better runtime.

In Table 4, we have investigated five real instances, namely f1 constituted of 63 haplotypes having 16,569 SNPs in each haplotype, i2 constituted of 40 haplotypes having 977 SNPs in each haplotype, k3 constituted of 100 haplotypes having 757 SNPs in each haplotype, m4 constituted of 26 haplotypes having 48 SNPs in each haplotype, and p5 constituted of 21 haplotypes having 16,548 SNPs. Flow-RM fail to solve these instances due to the large number of SNPs. So instead of solving the instances in general, the authors in Catanzaro et al. (2013) partitioned each instance to some blocks with the same number of haplotypes and the number of SNPs equal to 10, 13, and 15, and solved only one of these blocks as the most difficult block. The maximum number of SNPs for each instance that they were able to solve and their corresponding runtimes are given in the fourth and fifth columns of Table 4. Similar to Table 3, the second and third columns refer to the number of haplotypes and SNPs in each instance before and after preprocessing, respectively. The columns ‘L’ and ‘Time’, respectively, report the length of the phylogenetic tree and runtime of estimating process for each method.

The runtime of Pars is meaningfully less than those of Flow-RM and PULLPRU in all instances, and except in instances f1 and m4, Pars has obtained more accurate solutions than PULLPRU. The results of Tables 3 and 4 are corresponding to instances with small number of taxa for real instances we found in the literature. The simulation results approved that if there exist real instances with a large number of taxa (up to thousands), then PULLPRU would be more efficient.

5 Conclusion

In this paper, we have proposed a heuristic algorithm for most parsimonious phylogeny estimation problems from single nucleotide polymorphism (SNP) haplotypes (MPPEP-SNP) which is called PULLPRU. It solves the problem in several stages and each stage contains two steps of pullulation and pruning. Although pruning step requires finding the solution of a mixed integer programming (MIP), usually those MIPs are of small sizes, and their optimal solution can be found in a short time. We have evaluated the performance of PULLPRU on several simulated and real instances and compared the results with those of Pars heuristic program of Phylip and Flow-RM as one of the best exact models in terms of accuracy, runtime, and largeness of datasets. The numerical results show that PULLPRU can solve the large-scale simulated datasets with up to more than 1000 taxa, more efficiently than Pars. This would be beneficial for improving the accuracy of phylogeny estimation by increasing the number of taxa. The adjustment of PULLPRU on multi-state character data and its performance evaluation could be the subjects of further studies.

Acknowledgements This work is partly supported by a grant from University of Guilan, Rasht, IRAN.

References

Agarwala R, Fernández-Baca D (1994) A polynomial-time algorithm for the perfect phylogeny problem when the number of character states is fixed. SIAM J Comput 23:1216–1224

Bonizzoni P (2007) A linear-time algorithm for the perfect phylogeny haplotype problem. Algorithmica 48:267–285

Brooks DR, Bilewitch J, Condy C, Evans DC, Folinsbee KE, Fröbisch J (2007) Quantitative phylogenetic analysis in the 21st century. Rev Mex Biodivers 78:225–252

Bruni R (2010) Mathematical approaches to polymer sequence analysis and related problems. Springer, Berlin

Camin JH, Sokal RR (1965) A method for deducing branching sequences in phylogeny. Evolution 19:311–326

Catanzaro D (2009) The minimum evolution problem: overview and classification. Networks 53:112–125

Catanzaro D (2011) Estimating phylogenies from molecular data. In: Mathematical approaches to polymer sequence analysis and related problems. Springer, New York, pp 149–176

Catanzaro D, Ravi R, Schwartz R (2013) A mixed integer linear programming model to reconstruct phylogenies from single nucleotide polymorphism haplotypes under the maximum parsimony criterion. Algorithms Mol Biol 8:1

Ding Z, Filkov V, Gusfield D (2006) A linear-time algorithm for the perfect phylogeny haplotyping (PPH) problem. J Comput Biol 13:522–553

Felsenstein J (2004) Inferring phylogenies, vol 2. Sinauer Associates, Sunderland

Felsenstein J (2005) PHYLIP (phylogeny inference package) version 3.6. Department of Genome Sciences, University of Washington, Seattle

Foulds LR, Graham RL (1982) The Steiner problem in phylogeny is NP-complete. Adv Appl Math 3:43–49

Garey MR, Johnson DS (1979) A guide to the theory of NP-completeness. WH Freemann, New York

Gascuel O (2005) Mathematics of evolution and phylogeny. Oxford University Press, Oxford

Gatesy J, DeSalle R, Wahlberg N (2007) How many genes should a systematist sample? Conflicting insights from a phylogenomic matrix characterized by replicated incongruence. Syst Biol 56:355–363

Graham R, Foulds L (1982) Unlikelihood that minimal phylogenies for a realistic biological study can be constructed in reasonable computational time. Math Biosci 60:133–142

Gusfield D (1991) Efficient algorithms for inferring evolutionary trees. Networks 21:19–28

Gusfield D (2003) Haplotype inference by pure parsimony. In: Baeza-Yates R, Chávez E, Crochemore M (eds) Combinatorial Pattern Matching. CPM 2003. Lecture Notes in Computer Science, vol 2676. Springer. Berlin, Heidelberg, pp 144–155

Heath TA, Hedtke SM, Hillis DM (2008) Taxon sampling and the accuracy of phylogenetic analyses. J Syst Evol 46:239–257

Hedtke SM, Townsend TM, Hillis DM (2006) Resolution of phylogenetic conflict in large data sets by increased taxon sampling. Syst Biol 55:522–529

Hein J (1990) Reconstructing evolution of sequences subject to recombination using parsimony. Math Biosci 98:185–200

Hein J (1993) A heuristic method to reconstruct the history of sequences subject to recombination. J Mol Evol 36:396–405

Hillis DM, Pollock DD, McGuire JA, Zwickl DJ (2003) Is sparse taxon sampling a problem for phylogenetic inference? Syst Biol 52:124

Jin G, Nakhleh L, Snir S, Tuller T (2007) Efficient parsimony-based methods for phylogenetic network reconstruction. Bioinformatics 23:e123–e128

Kannan S, Warnow T (1997) A fast algorithm for the computation and enumeration of perfect phylogenies. SIAM J Comput 26:1749–1763

Misra N, Blelloch G, Ravi R, Schwartz R (2011) Generalized Buneman pruning for inferring the most parsimonious multi-state phylogeny. J Comput Biol 18:445–457

Pollock DD, Zwickl DJ, McGuire JA, Hillis DM (2002) Increased taxon sampling is advantageous for phylogenetic inference. Syst Biol 51:664

Rosenberg MS, Kumar S (2003) Taxon sampling, bioinformatics, and phylogenomics. Syst Biol 52:119

Semple C, Steel MA (2003) Phylogenetics, vol 24. Oxford University Press, Oxford

Sforza CL, Edwards AWF (1964) Analysis of human evolution. Genet. Today 3:923–933

Sridhar S, Lam F, Blelloch GE, Ravi R, Schwartz R (2007) Efficiently finding the most parsimonious phylogenetic tree via linear programming. In: Măndoiu I, Zelikovsky A (eds) Bioinformatics Research and Applications. ISBRA 2007. Lecture Notes in Computer Science, vol 4463. Springer. Berlin, Heidelberg, pp 37–48

Sridhar S, Lam F, Blelloch GE, Ravi R, Schwartz R (2008) Mixed integer linear programming for maximum-parsimony phylogeny inference. IEEE/ACM Trans Comput Biol Bioinf 5:323–331

Zheng W, Zheng WM(2015) A brief review to phylogenetic reconstruction by maximum parsimony. In: 2015 Fifth International Conference on Instrumentation and Measurement, Computer, Communication and Control (IMCCC) Qinhuangdao, China (pp 830–835). IEEE. https://doi.org/10.1109/IMCCC.2015.181

Zwickl DJ, Hillis DM (2002) Increased taxon sampling greatly reduces phylogenetic error. Syst Biol 51:588–598