Parameterized Algorithms for the Steiner Arborescence Problem on a Hypercube

Sugyani Mahapatra\textsuperscript{1,2,3}\textsuperscript{*}, Manikandan Narayanan\textsuperscript{1,2,3}\textsuperscript{*} and N. S. Narayanaswamy\textsuperscript{1}\textsuperscript{*}

\textsuperscript{1}Department of Computer Science and Engineering, Indian Institute of Technology (IIT) Madras, Chennai, Tamil Nadu, India.
\textsuperscript{2}Center for Integrative Biology and Systems Medicine, IIT Madras, Chennai, Tamil Nadu, India.
\textsuperscript{3}Robert Bosch Centre for Data Science and Artificial Intelligence, IIT Madras, Chennai, Tamil Nadu, India.

\textsuperscript{*}Corresponding author(s). E-mail(s): msugyani@cse.iitm.ac.in; nmanik@cse.iitm.ac.in; swamy@cse.iitm.ac.in;

Abstract
Motivated by a phylogeny reconstruction problem in evolutionary biology, we study the minimum Steiner arborescence problem on directed hypercubes (MSA-DH). Given \( m \), representing the directed hypercube \( \vec{Q}_m \), and a set of terminals \( R \), the problem asks to find a Steiner arborescence that spans \( R \) with minimum cost. As \( m \) implicitly represents \( \vec{Q}_m \) comprising \( 2^m \) vertices, the running time analyses of traditional Steiner tree algorithms on general graphs does not give a clear understanding of the actual complexity of this problem. We present algorithms that exploit the structure of the hypercube and run in time polynomial in \( |R| \) and \( m \).

We explore the MSA-DH problem on three natural parameters - \( |R| \), and two above-guarantee parameters, number of Steiner nodes \( p \) and penalty \( q \). For above-guarantee parameters, the parameterized MSA-DH problem take \( p \geq 0 \) or \( q \geq 0 \) as input, and outputs a Steiner arborescence with at most \( |R| + p - 1 \) or \( m + q \) edges respectively. We present the following results (\( \tilde{O} \) hides the polynomial factors):

1. An exact algorithm that runs in \( \tilde{O}(3^{|R|}) \) time.
2. A randomized algorithm that runs in \( \tilde{O}(9^q) \) time with success probability \( \geq 4^{-q} \).
3. An exact algorithm that runs in \( \tilde{O}(36^q) \) time.
4. A \((1 + q)\)-approximation algorithm that runs in \( \tilde{O}(1.25284^q) \) time.
5. An \( \mathcal{O}(p\ell_{\text{max}}) \)-additive approximation algorithm that runs in \( \tilde{\mathcal{O}}(\ell_{\text{max}}^{p+2}) \) time, where \( \ell_{\text{max}} \) is the maximum distance of any terminal from the root.

**Keywords:** Steiner arborescence, Directed hypercube, Parameterized algorithms, FPT (Fixed-Parameter Tractable) algorithms, Parameterized approximation algorithms

1 Introduction

1.1 Problem and Motivation

A long-standing problem inspired by Darwin’s theory of evolution is to reconstruct the “tree of life” or phylogeny relating all current-day species from their DNA/protein sequences. The problem is to specifically infer a phylogenetic tree [11] that relates the observed sequences of current-day species via the unobserved sequences of their common ancestral species. We can model this problem as the hypercube variant of the Steiner tree problem, by considering the binary representation of the observed sequences as input terminals and unobserved ancestral sequences as Steiner nodes. Finding a Steiner tree in an undirected hypercube solves the phylogeny problem when mutations are reversible (i.e., both \( 0 \rightarrow 1 \) and \( 1 \rightarrow 0 \) bit flips/changes are allowed), and a directed hypercube solves the problem when mutations are irreversible (i.e., only \( 0 \rightarrow 1 \) bit flips are possible). The former is better studied, and the focus of this work is on open questions related to the latter problem of finding a Steiner Arborescence in a directed hypercube. Given that the Steiner tree problem is one of Karp’s 21 NP-complete problems [3], there is rich work on the parameterized complexity of the Steiner tree problem on general as well as special-case graphs. For instance in general graphs, the Steiner tree problem is FPT on the number of input terminals \( |R| \) as parameter [9, 21], but \( W[2] \)-hard on the number of Steiner nodes \( p \) as parameter [10]. Despite this rich literature, the directed hypercube variant of the Steiner tree focused in this study has not been addressed before.

We now formally state our problem. For an \( m \geq 1 \), the \( m \)-hypercube \( Q_m \) has all the \( 2^m \) binary strings of length \( m \) as the vertex set, i.e., \( V(Q_m) = \{0,1\}^m \), and there exists an edge between two vertices \( t_1 \) and \( t_2 \) if they differ exactly at a single bit position. Every bit position is also known as a ‘character’ [2, 23]. \( \bar{Q}_m \) denotes the directed hypercube rooted at \( 0^m \) such that the edges are oriented away from the root towards vertices with a higher number of 1s. For an arborescence \( T \), \( \text{COST}(T) \) denotes \( |E(T)| \). Then MSA-DH problem is stated as follows.

**Minimum Steiner Arborescence on Directed Hypercube (MSA-DH) Problem**

**Instance:** An integer \( m \), and a set of terminals \( R \subseteq V(\bar{Q}_m) \).

**Question:** Find a Steiner arborescence \( T \) in \( \bar{Q}_m \), rooted at \( 0^m \), that spans \( R \) with the minimum cost.

As \( 0^m \) must be in any output Steiner arborescence, we assume without loss of generality that \( 0^m \in R \). Note that in a solution arborescence \( T \), the path from the root to a terminal encodes a sequence of bit flips/changes, one at a time, to transform or
mutate the root to become the terminal. Thus, each edge in $T$ is associated with the bit flip/change of one of the $m$ characters. This problem models a variant of the phylogeny problem as discussed above, called the Camin-Sokal Phylogeny [4], and is also motivated by a related model in network science known as the Expander Hierarchy Model [14], which uses a hierarchical tree representation of graph objects to efficiently compute properties of a dynamic graph.

We cannot hope for an algorithm that runs in time polynomial in $|R|$ and $m$, since the MSA-DH problem on hypercubes encode a generalized version of the minimum vertex cover problem and thus, is NP-complete for both undirected [13] and directed [6] cases. Owing to the exponential size of the underlying graph, existing algorithms for Steiner trees on general graphs will run in time exponential in the input size. There are some parameterized complexity [2, 20, 23] and approximability [12] results on the penalty parameter $q$ (defined as the extra cost incurred by the solution on the characters) for the undirected version of the problem. However, in the directed case, i.e., the MSA-DH problem, there are no such results to the best of our knowledge. The case when the optimal penalty $q_{opt} = 0$ is called the Perfect Phylogeny problem and can be solved in polynomial time [7, 8, 15, 19]. A summary of the literature is in Table 1 and provides the context for our work.

### Table 1: Brief review of literature on Steiner trees in different types of graphs

The black and purple colors show existing literature and our contributions respectively.

| Parameter | General Graphs | DAG | Hypercube |
|-----------|----------------|-----|-----------|
| $|R|$ | $O(2^{\log^{+\epsilon} |R|})$ unless NP $\subseteq$ ZTIME | $O(2^{\log^{+\epsilon} |R|})$ | OPEN $O(1)^\text{opt}$ cf. [Theorem 1] |
| $p$ | $O(2^{\log^{+\epsilon} |R|})$ unless NP $\subseteq$ ZTIME | $O(2^{\log^{+\epsilon} |R|})$ | OPEN $O(1)^\text{opt}$ with $\epsilon > 4^{-4}$ cf. [Theorem 2] $O(1)^{\text{opt}}$ cf. [Lemma 4] |
| $q$ | NA | NA | NA $O(2^{\log^{+\epsilon} |R|})$ cf. [Theorem 1] |

| Approximation Algorithms | General Graphs | DAG | Hypercube |
|--------------------------|----------------|-----|-----------|
| $\epsilon$-approx [24] | No $O(2^{\log \epsilon^{-\epsilon} |R|})$ unless NP $\subseteq$ ZTIME | No $O(2^{\log \epsilon^{-\epsilon} |R|})$ unless NP $\subseteq$ ZTIME | OPEN $O(1)^{\text{opt}}$ with $\epsilon > 4^{-4}$ cf. [Theorem 2] $O(1)^{\text{opt}}$ with $\epsilon > 4^{-4}$ |

| Parameterized Approximation Algorithms | General Graphs | DAG | Hypercube |
|---------------------------------------|----------------|-----|-----------|
| $|R|$ | PAS and PSAKS, no polynomial kernel unless NP $\subseteq$ ZTIME | OPEN | OPEN $O(1)^{\text{opt}}$ |
| $p$ | $O(p^{\epsilon})$-approximation in $O(|R|^{1+\epsilon})$ cf. [Theorem 4]; $O(p^{\epsilon})$-approximation in $O(|R|^{1+\epsilon})$ cf. [Theorem 4] | EPAS [10] | OPEN $O(1)^{\text{opt}}$ |
| $q$ | NA | NA | $(1+q)$-approximation in $O(2^{\log^{+\epsilon} |R|})$ cf. [Theorem 3]; $(1+q)$-approximation in $O(2^{\log^{+\epsilon} |R|})$ cf. [Theorem 3] |

1. **Our Results**

We consider three parameters - number of terminals $|R|$, number of Steiner nodes $p$ and penalty $q$. To consider the parameterizations with respect to $p$ and $q$ as above guarantee...
parameters, we observe the following lower bounds on the cost of any solution. Since $|R|$ is the number of terminals, any solution has at least $|R| - 1$ edges. Thus the cost of a Steiner arborescence $T$ is given by $\text{COST}(T) = |R| - 1 + p_T$ where $p_T$ is the number of Steiner nodes. For the parameter $q$, note that a character with the same bit value across all the terminals does not play a significant role in the construction of the Steiner arborescence (cf. Lemma 1); thus we assume that for each of the $m$ characters, the terminal set can be partitioned into two non-empty sets – those in which the character is 0 and those in which it is 1. Thus in any solution, the value of each character must change at least once along some edge in $T$. Thus, $\text{COST}(T) \geq m$, and the penalty $q_T$ of the arborescence $T$ is given as $q_T = \text{COST}(T) - m$.[2, 23]. The optimum number of Steiner nodes is denoted by $p_{opt} = \min_T p_T$; and the optimal penalty is denoted by $q_{opt} = \min_T q_T$.

The parameterized (decision) version of the problem is now defined as:

**Parameterized MSA-DH problem**

**Instance:** An integer $m$, and a set of terminals $R \subseteq V(\tilde{Q}_m)$.

**Parameters:** $|R|$, penalty $q \geq 0$, number of Steiner nodes $p \geq 0$.

**Question:** Is there a Steiner arborescence $T$ in $\tilde{Q}_m$, rooted at $0^m$, that spans $R$ and has cost at most $m + q$ or at most $|R| + p - 1$?

In the decision algorithms we attempt to compute the optimal Steiner arborescence, thus giving the value of $p_{opt}$ and $q_{opt}$. Thus, if $q \geq q_{opt}$ ($p \geq p_{opt}$), we output the Steiner arborescence $T$; otherwise when $q < q_{opt}$ ($p < p_{opt}$), the algorithms output “No”. We use the notation $\text{FPT}[k]$ to denote an $\text{FPT}$ algorithm parameterized by $k$.

On the contrary, with parameterized approximation algorithms, we attempt to compute an approximate solution with better running times. We adopt the definition used by Dvorák et al.[10], i.e., if $q \geq q_{opt}$ ($p \geq p_{opt}$), we output an approximate solution with performance ratio $\alpha$ on the cost; otherwise the output is arbitrary.

We first present a dynamic programming formulation similar to the $\text{FPT}[|R|]$-algorithm for the Steiner tree problem in undirected graphs by Dreyfus and Wagner[9].

**Theorem 1.** The MSA-DH problem can be solved in $O\left(3^{|R|} |R|m + |R|^2 m^2\right)$ time and $O\left(2^{|R|} \log |R| m\right)$ space.

For the parameter $q$, the undirected version of the problem admits a randomized $\text{FPT}[q]$-algorithm (see Table 1). Inspired by this algorithm, we present a randomized $\text{FPT}[q]$-algorithm for the parameterized MSA-DH problem as well.

**Theorem 2.** The parameterized MSA-DH problem can be solved in $O\left(9^q q^3 + q |R| m^2\right)$ time with success probability $\geq 4^{-q}$.

The algorithm can be derandomized using a depth-bounded search tree in $\tilde{O}(36^q)$ time (cf. Lemma 4).

We next present a novel parameterized approximation algorithm with improved running time but at the expense of exactness by using the exact algorithm for the minimum vertex cover problem [5].

**Theorem 3.** There is a $(1+q)$-approximation algorithm for the parameterized MSA-DH problem that runs in $\tilde{O}(1.25284^q)$ time, where $\tilde{O}$ hides the polynomial factors in the input size.
The Steiner tree problem is infamously \( \mathcal{W}[2] \)-hard on the parameter \( p \), the number of Steiner nodes, for general graphs and has not been explored for DAGs and undirected hypercubes. To the best of our knowledge, the parameterized complexity of the problem on directed hypercubes is yet to be known. We present a novel parameterized approximation algorithm on the parameters \( p \) and \( \ell_{\max} \) for the same, where \( \ell_{\max} \) is the maximum distance of any terminal from the root.

**Theorem 4.** There is a \( O(p \ell_{\max}) \)-additive approximation algorithm for the parameterized MSA-DH problem that runs in \( O(p^{\ell_{\max}+2} + \ell_{\max}^2 |R|^2 m) \) time, where \( \ell_{\max} \) is the maximum distance of any terminal from the root.

In our algorithms, the key point is to note the polynomial dependence on the \( m \) and \(|R|\) for the parameters \( p \) and \( q \), and the exponential dependence on the parameter. The algorithms have all exploited the nature of the directed hypercube, and we believe that the dependence on the parameters can be further improved. In particular, we have not been able to improve the running time of the algorithm parameterized by the number of terminals from \( \tilde{O}(3^{|R|}) \) to \( \tilde{O}(2^{|R|}) \). We also note that while these algorithms can be used to solve the Steiner problem in an undirected hypercube, the resulting tree may not be optimal (e.g., Remark 1) or adhere to the approximation ratios. Conversely, results on the Steiner problem in an undirected hypercube cannot of course be assumed to hold “as is” for its directed version (where \( 1 \rightarrow 0 \) character flips/changes are disallowed). These observations motivate our results above on the complexity of the Steiner problem in a directed hypercube.

## 2 Preliminaries

Given the dimension \( m \), let \( \bar{Q}_m \) denote the directed hypercube rooted at \( 0^m \). Let \( R \subseteq V(\bar{Q}_m) = \{0,1\}^m \) be the input set of terminals. We denote an optimal Steiner arborescence as \( T_{\text{opt}} \) and the output Steiner arborescence as \( T \) (\( T_{\text{opt}}^R \) and \( T^R \) denote the optimal and output Steiner arborescence on the terminal set \( R \) respectively). \( V(T) \) and \( E(T) \) denote the node and edge set of the arborescence \( T \) respectively. We use \( OPT_R \) and \( SOL_R \) to denote cost of the optimal and output arborescence respectively for \( R \). When \( R \) is clear from the context, we just write \( OPT \) and \( SOL \).

Throughout the paper, \( I = (m, R) \) denotes a MSA-DH problem instance. We assume \(|R| \geq 2\) without loss of generality (otherwise the solution is trivial). Let \([m]\) denote the set of all \( m \) characters for \( I \). For \( t \in V(\bar{Q}_m) \) and \( u \in [m] \), \( t[u] \) denotes the value of the \( u^{th} \) character in \( t \). Also, we use the notation \([i,j]\) to represent the set \{\( i,i+1,\ldots,j \)\}.

For \( t_1, t_2 \in V(\bar{Q}_m) \), let \( d(t_1,t_2) \) represent the Hamming distance between \( t_1 \) and \( t_2 \). The vertices of \( \bar{Q}_m \) are partitioned into \( m + 1 \) levels based on the Hamming distance of the vertices from \( 0^m \). For any node \( t \in V(\bar{Q}_m) \), \( \text{level}(t) = d(0^m,t) \) represents the level of \( t \) in \( \bar{Q}_m \). We use \( \ell_{\max} \) to represent the maximum level of the terminals i.e. \( \ell_{\max} = \max_{t \in R} \text{level}(t) \). \( \bar{Q}_m^{\ell} \) is the set of nodes in \( \bar{Q}_m \) at level \( \ell \). Given any node \( t \in V(\bar{Q}_m) \), we use \( \text{Ancestor}(t) \), \( \text{Descendant}(t) \), \( \text{parent}(t) \) and \( \text{child}(t) \) to represent the sets of ancestors, descendants, parents and children of \( t \) respectively.

For a subset \( S \subseteq V(\bar{Q}_m) \), the least common ancestor of \( S \) in the hypercube, denoted by \( \text{LCA}(S) \), is that node in the hypercube whose descendants contain \( S \), but no child
Dynamic Programming on Number of Input LCA problems are rooted at contain invariant characters. Thus, the optimal arborescences considered for the subproblems the subsets of the input terminal set are not preprocessed and could mon ancestors and is given in Lemma 2. It must be noted here that in the recursive algorithm \[9\]. The optimal substructure of the DP formulation is based on least com-

Our Dynamic Programming (DP) formulation has similarities to the Dreyfus-Wagner Lemma 2. For an instance \[m\] such that for all \(t_1, t_2 \in R \setminus \{0^m\}\), \(t_1[u] = t_2[u]\). Let \(I' = (m', R')\), where \([m'] = [m] \setminus \{u\}\) and \(R' = \{t' | \exists t \in R \text{ such that } \forall v \in [m'] \ t'[v] = t[v]\}. Then

\[
OPT_R = \begin{cases} 
OPT_{R'} & \text{if } \forall t \in R \setminus \{0^m\} \ t[u] = 0 \\
OPT_{R'} + 1 & \text{if } \forall t \in R \setminus \{0^m\} \ t[u] = 1 
\end{cases}
\]

Proof. Observe that an optimum solution for \(R\) exists with the following canonical property – if \(t[u] = 0\) for all terminals, then the character \(u\) does not change along any edge in the solution. On the other hand, if \(t[u] = 1\) for all terminals excluding \(0^m\), then the character \(u\) changes once on the only edge leaving \(0^m\), and does not change along any other edge. This lemma immediately follows from this observation. \(\square\)

We thus assume every input instance is preprocessed by Lemma 1, unless mentioned explicitly. As a consequence of this lemma, \(LCA(R) = 0^m\). Also note that if there is no requirement to root the Steiner arborescence at \(0^m\), every optimal Steiner arborescence for \(R\) will be rooted at \(LCA(R)\).

### 3 Dynamic Programming on Number of Input Terminals

Our Dynamic Programming (DP) formulation has similarities to the Dreyfus-Wagner algorithm \[9\]. The optimal substructure of the DP formulation is based on least common ancestors and is given in Lemma 2. It must be noted here that in the recursive subproblems the subsets of the input terminal set are not preprocessed and could contain invariant characters. Thus, the optimal arborescences considered for the subproblems are rooted at \(LCA\) of the corresponding subset of the terminal set instead of \(0^m\).

Lemma 2. For an instance \(I\) of the MSA-DH problem, there exists \(\phi \neq R' \subset R\) and \(R'' = R \setminus R'\) such that \(T_{opt}^R\) can be decomposed into two subtrees that are each optimal Steiner arborescences for terminal sets \(R'\) and \(R''\) respectively. As a result,

\[
OPT_R = OPT_{R'} + OPT_{R''} + d(LCA(R), LCA(R')) + d(LCA(R), LCA(R'')) \tag{1}
\]

Proof. We know that \(LCA(R)\) will be the root of \(T_{opt}^R\). If \(LCA(R)\) has at least two children in \(T_{opt}^R\), then let \(x\) be any one of its child. Let \(R'\) be the set of all terminals spanned by the subtree of \(T_{opt}^R\) rooted at \(x\), denoted by \(T_x\), and let \(R'' = R \setminus R'\). Since \(T_{opt}^R\) is a minimum cost Steiner arborescence, note that \(R' \neq \phi\). Note also that \(R'' \neq \phi\) (by a similar argument where we let \(x\) to be any other child of \(LCA(R)\)). So we have a non-trivial partition of \(R\) (see Figure 1a for an illustration).

Clearly, \(T_x\) is an optimal Steiner arborescence spanning \(R' \cup \{x\}\); otherwise we can replace \(T_x\) by another arborescence of smaller cost which (when combined with the rest of \(T_{opt}^R\)) would result in a valid arborescence with smaller cost, contradicting the optimality of \(T_{opt}^R\). Now, \(T_x\) can be split into a path from \(x\) to \(LCA(R')\) and \(T_{opt}^{R''}\).
(a) When $\text{LCA}(R)$ has at least two subtrees. (b) When $\text{LCA}(R)$ has one subtree.

Fig. 1: The structure of the output Steiner arborescence. $R'$ and $R''$ represent any non-empty subset of the input terminal set $R$. Triangles represent subtrees while circles represent nodes in the arborescence. The dotted and solid arrows represent paths and edges respectively.

(it could be that $x = \text{LCA}(R')$ in which case the path is trivial/empty). Therefore, we have decomposed the edge $(\text{LCA}(R), x)$ and $T_x$ into a path from $\text{LCA}(R)$ to $\text{LCA}(R')$ and $T_{\text{opt}}^{R'}$. Let $T_{\text{rem}}$ be the subtree of $T_{\text{opt}}^{R'}$ obtained by removing the edge $(\text{LCA}(R), x)$ and $T_x$ (see Figure 1a). A similar proof as above can be used to show that $T_{\text{rem}}$ is an optimal Steiner arborescence spanning $R' \cup \{\text{LCA}(R)\}$ and can be decomposed into a path from $\text{LCA}(R)$ to $\text{LCA}(R')$ and $T_{\text{opt}}^{R'}$.

We now consider the case when $\text{LCA}(R)$ has only one child $x$ (see Figure 1b). $\text{LCA}(R)$ must be a terminal, otherwise $x = \text{LCA}(R)$. Let $R' = \{\text{LCA}(R)\}$ and $R'' = R \setminus R'$. Then $T_{\text{opt}}^{R'}$ can be split into the four subtrees/paths that the lemma requires.

Lemma 2 leads to the following corollary:

Corollary 1. The cost of an optimal Steiner arborescence on the terminal set $R$ is:

$$
\text{OPT}_R = \min_{\phi \neq R' \subset R} \left\{ \text{OPT}_{R'} + \text{OPT}_{R \setminus R'} + d(\text{LCA}(R), \text{LCA}(R')) + d(\text{LCA}(R), \text{LCA}(R \setminus R')) \right\}
$$

Algorithm 1 uses Corollary 1 to compute the optimal cost on every possible subset of the terminal set, and stores the DP backpointers to the subsets that result in the optimal cost. Finally, the Steiner arborescence is constructed by following these backpointers. The pseudocode is given in Algorithm 1 and the algorithm is analyzed in Theorem 1. We comment on the usage of this algorithm for the undirected hypercube case in Remark 1.

Theorem 1. The MSA-DH problem can be solved in $O(3^{|R|} |R| m + |R|^2 m^2)$ time and $O(2^{|R|} \log |R| m)$ space.
Algorithm 1  A Dynamic Programming Algorithm for the MSA-DH problem.

Input: Dimension of the hypercube $m$, terminal set $R$.

Parameter: The number of input terminals $|R|$.

Output: An optimal Steiner arborescence $T_{opt}$.

Data Structure: A vector $M$ such that $M[S]$ stores the optimal cost of the Steiner arborescence spanning $S \subseteq R$. There are two backpointers from $S$ to $S' \subset S$ and $S'' = S \setminus S'$. Initially, the values in the vector is set to zero and the backpointers to NULL.

1: for $R' \subseteq R$ (in increasing order of $|R'|$) do
2: if $|R'| > 1$ then
3: $M[R']$ is computed using Corollary 1 as:
4: $M[R'] \leftarrow \min_{\phi \neq S \subseteq R'} \{M[S] + M[R' \setminus S] + d(LCA(R'), LCA(S)) + d(LCA(R'), LCA(R' \setminus S))\}$
5: Let $S_{min} \subseteq R'$ be the subset that results in the minimum value. Add the following backpointers – $R'$ to $S_{min}$ and $R'$ to $R' \setminus S_{min}$.
6: end if
7: end for
8: Add $0^m$ to the arborescence $T_{opt}$.
9: Traverse the backpointers from $R$ until NULL (e.g., using a DFS) and add $LCA$ of the subsets to $T_{opt}$ as roots of the corresponding subtrees.

Proof. The optimality of the output Steiner arborescence follows directly from Lemma 2 and Corollary 1. Since the least common ancestor can be computed in $O(|R|m)$ time and there are at most $|R|m$ edges in the optimal Steiner arborescence, the backtracking can be performed in $O((|R|m)^2)$ time. Thus, the time complexity of the algorithm is dominated by the time required by the iterations. The loop runs for every possible subset of the terminal set $R$ and for every subset it explores all its proper subsets. We assume the two table lookups take unit time each. Thus the total time required by the loop is:

$$
\sum_{|R'|=2}^{\lfloor |R|/2 \rfloor} \sum_{|S|=1}^{\lfloor |R'|/2 \rfloor} \binom{|R'|}{|S|} \binom{|R'| - 1}{|R'| - |S|} (2 + |R'|m + |S|m + (|R'| - |S|)m + m \log(|R'|m))
\leq \sum_{|R'|=0}^{\lfloor |R|/2 \rfloor} \sum_{|S|=0}^{\lfloor |R'|/2 \rfloor} \binom{|R'|}{|S|} 2(1 + |R'|m + m) \leq 2(1 + |R|m + m) \sum_{|R'|=0}^{\lfloor |R|/2 \rfloor} \left(\frac{|R|}{|R'|}\right)^2 |R'|
= 2(1 + |R|m + m)2^{|R|}
$$

The optimal cost for every possible subset of the terminal set is stored in a $O(2^{|R|})$ length vector. The maximum cost possible is $|R|m$ which requires $\log |R|m$ bits. We assume the two pointers use constant space, thus proving the space complexity. $\square$
Remark 1. It may be noted here that our algorithm can be used for the undirected hypercube case but the resulting arborescence, even though would be a correct solution, may not be optimal.

4 FPT Algorithms Parameterized by Penalty

In this section, we present two parameterized algorithms on the penalty parameter. We first present a randomized algorithm for the parameterized MSA-DH problem that has some similarities to the ‘buildNPP’ algorithm by Blelloch et al. [2] and Sridhar et al. [23] for undirected hypercubes. Following this, we present a novel parameterized approximation algorithm for the MSA-DH problem using a FPT algorithm for minimum vertex cover problem.

4.1 Preliminaries for Algorithms on Penalty $q$

We first study certain properties of the input characters and structure of the output arborescence that have been used to design our algorithms. Based on these properties, we also present an additional preprocessing that plays a key role in the exact FPT algorithm (and can be optionally used by the other algorithms).

4.1.1 Background on Classifications of Characters

Given an arborescence $T$, a character $u \in [m]$ is a ‘good character’ of $T$ if $u$ changes along exactly one edge in $T$; otherwise $u$ is a ‘bad character’ of $T$ [2, 23]. Let $\text{GOOD}(T)$ and $\text{BAD}(T)$ denote the set of all good and bad characters of $T$ respectively. Clearly, the penalty of $T$, $q_T \geq |\text{BAD}(T)|$, as also shown in Blelloch et al. [2] and Sridhar et al. [23].

Given a set of terminals $R$, a pair of characters $u, v \in [m]$ is said to conflict with each other if there exists $t_1, t_2, t_3, t_4 \in R$ such that $(t_1[u] = 0 \land t_1[v] = 0)$ and $(t_2[u] = 0 \land t_2[v] = 1)$ and $(t_3[u] = 1 \land t_3[v] = 0)$ and $(t_4[u] = 1 \land t_4[v] = 1)$ [11]. This condition is known as the four-gamete condition. We say a character $u \in [m]$ is conflicting if there is another character $v \in [m]$ such that $u$ and $v$ conflict with each other in $R$. Similarly, a character $u \in [m]$ is isolated if it does not conflict with any other character.

Blelloch et al. [2] and Sridhar et al. [23] define the conflict graph as an undirected graph $CG(R) = (V, E)$, where $V = [m]$ and $E = \{(u, v) \mid u$ and $v$ conflict with each other}. As it takes $O(|R|)$ time to determine if two bits conflict with each other, the conflict graph can be constructed in $O(|R|m^2)$ time.

When the conflict graph does not have any edges, i.e., when all the characters are isolated, the instance is known to satisfy the perfect phylogeny condition [11]. When this condition is satisfied, it is well-known that the MSA-DH problem can be solved in polynomial time [7, 8, 15, 19], and all the characters are good in the resulting optimal arborescence. Thus, the following corollary holds true.

Corollary 2. If there are no edges in the conflict graph, then $q_{opt} = 0$ and the MSA-DH problem can be solved in polynomial time.

For a character $u \in [m]$, we define $R_u = \{t \in R \mid t[u] = 1\}$. In an arborescence $T$, we use the notation $e_u = (x_u, x_u)$ to denote an edge along which $u$ changes in $T$ (i.e.,
Let $x_u$ and $x_i$ have the same bit value for all characters except the character $u$ at which $x_u[u] = 0$ and $x_i[u] = 1$. Note that if $u ∈ \text{GOOD}(T)$, then there is only one such edge $e_u$.

### 4.1.2 Structure of the Output Arborescence

We further study the relationship between the properties of characters and structure of output arborescences, as it forms the basis of our algorithms. We first show the following bounds on $q_{opt}$.

**Proposition 1.** For an instance $I$ of the MSA-DH problem, the following statements hold true.

1. If $q_{opt} < m$, then every $T_{opt}$ has at least one good character.
2. If $q_{opt} ≥ m$, then $|R| ≤ 2q_{opt} + 1$.

**Proof.** If $q_{opt} < m$, $OPT = m + q_{opt} < 2m$. Then all the characters cannot flip at least twice, which implies that there is at least one character which flips once. In fact, $T_{opt}$ has at least $m - q_{opt}$ good characters, since $m = |\text{GOOD}(T_{opt})| + |\text{BAD}(T_{opt})|$, and $|\text{BAD}(T_{opt})| ≤ q_{opt}$. If $q_{opt} ≥ m$, $OPT ≤ 2q_{opt}$. As $|R| − 1 ≤ OPT$, $|R| ≤ 2q_{opt} + 1$. □

So, if $q_{opt} ≥ m$, then the input size is bounded by $q_{opt}$ and any (trivial) exponential algorithm will construct a Steiner arborescence in FPT time. Instances for which $q_{opt} < m$ are more interesting since the hardness of the parameterized version of the problem for this scenario cannot be inferred directly. Henceforth, we assume $q_{opt} < m$ and study this case in more details.

We use **Proposition 2** on the structure of an optimal arborescence multiple times in the design of the exact algorithms.

**Proposition 2.** Consider a path $P = (x_1, x_2, \ldots, x_k, x_{k+1})$ (for $k ≥ 2$) such that $∀ i ∈ [2, k] x_i$ is a Steiner node and has exactly one child (i.e., $x_{i+1}$) in an optimal Steiner arborescence $T_{opt}$. Let the characters $u_1, \ldots, u_k$ flip along the corresponding edges in $P$ in the same order, and $u'_1, \ldots, u'_k$ be any permutation of these characters. Then there exists an optimal Steiner arborescence $T'_{opt}$ such that $u'_1, \ldots, u'_k$ flip along corresponding edges in the path $P' = (x_1, x'_2, \ldots, x'_k, x_{k+1})$ in $T'_{opt}$ in the same order and the rest of $T'_{opt}$ is identical to $T_{opt}$.

**Proof.** Observe that $∀ j ∈ [1, k] x_j[u_j] = 0$ and $x_{k+1}[u_j] = 1$. So there is a path $P' = (x_1, x'_2, \ldots, x'_k, x_{k+1})$ in $\bar{Q}_m$ such that $u'_1, \ldots, u'_k$ change along the corresponding edges in the same order. Also notice that $∀ i ∈ [2, k] x'_i ∉ V(T_{opt})$; otherwise if for some $i ∈ [2, k]$ $x'_i$ is in $T_{opt}$, then we can replace the path $P$ with a path from $x'_i$ to $x_{k+1}$ and obtain a valid Steiner arborescence with cost reduced at least by 1. As a consequence of these two observations, we can replace the path $P = (x_1, x_2, \ldots, x_k, x_{k+1})$ in $T_{opt}$ with $P' = (x_1, x'_2, \ldots, x'_k, x_{k+1})$ and obtain a valid Steiner arborescence $T'_{opt}$. As the length of the new replaced path is same as the length of the original path, $\text{COST}(T'_{opt}) = \text{COST}(T_{opt})$ and $T'_{opt}$ is also optimal. □

**Remark 2.** The above proposition leads to an interesting observation. Let $u$ be a good character of $T_{opt}$, and $T_u$ be the subtree of $T_{opt}$ rooted at $x_u$. Now, every terminal $t ∈ R_u$ is in $T_u$. Also $R_u$ is rooted at $y = \text{LCA}(R_u)$ which must be a descendent of

10
Notice that, there cannot be any subtrees along the path from $x_u$ to $y$ in $T_{opt}$. Let $u_1, u_2, \ldots, u_k$ be the set of characters that change along some edge in $P_1$ and $P_2$. If $B_1 = B_2$, then $x_1 = x_2$, which contradicts the claim that $e_1$ and $e_2$ are distinct. Let $T_3$ be rooted at some node $z$ in $P_1$. If there is a character $v \in B_1 \setminus B_2$ that flips along some edge in the path from $x$ to $z$, then $(t_1[v] = 1 \wedge t_1[z] = 1)$ and $(t_2[u] = 1 \wedge t_2[v] = 0)$ and $(t_3[u] = 0 \wedge t_3[v] = 1)$. Now the root $r = 0^m$ has $(r[u] = 0 \wedge r[v] = 0)$, implying that $u$ and $v$ are swapped, which is a contradiction.

We now consider the case when there are no such characters along both the paths. Let $x_1 \in B_1$ and $x_2 \in B_2$ be the first characters that flip along some edge in $P_1$ and $P_2$ respectively such that $v_1 \notin B_2$ and $v_2 \notin B_1$. Let $v_1$ and $v_2$ change along the edges $(x_{v_1}, x_{v_1})$ and $(x_{v_2}, x_{v_2})$. Observe that there are no subtrees along the paths $P'_1$ (from $x_{v_1}$ to $x_1$) and $P'_2$ (from $x_{v_2}$ to $x_2$). Again by Proposition 2, the characters changing
along the paths from \( x_{v_1} \) to \( x_1 \) and from \( x_{v_2} \) to \( x_2 \) can be permuted to obtain an optimal Steiner arborescence. Let \( B'_1, B'_2 \subseteq B_1 \cap B_2 \) be the subset of characters that change along \( P'_1 \) and \( P'_2 \) respectively. Then, from \( x_{v_1} \) we first change all characters in \( B'_1 \) and then we change \( v_1 \). Similarly, from \( x_{v_2} \) we first change all characters in \( B'_2 \) and then we change \( v_2 \). In the new paths, let \( v_1 \) and \( v_2 \) change along the edges \((x'_{v_1}, x_{v_1})\) and \((x'_{v_2}, x_{v_2})\) respectively. Then, \( x'_{v_1} = x'_{v_2} \), contradicting the optimality of \( T_{\text{opt}} \). \( \square \)

We use Proposition 3 to prove the following lemma.

**Lemma 3.** Let \( u \in [m] \) be an isolated character in a MSA-DH problem instance \( I = (m, R) \). Let \( I' = (m', R') \) be defined as \([m'] = [m] \setminus \{u\}\) and \( R' = \{t' \mid \exists t \in R \text{ such that } \forall v \in [m'] \ t'[v] = t[v]\}\). Then, \( \text{OPT}_R = \text{OPT}_{R'} + 1 \).

**Proof.** By Proposition 3, there is exactly one edge \( e_u = (x_u, x_u) \) along which \( u \) changes in \( T_{\text{opt}}^{R_u} \). We remove the edge \( e_u \), connect every child of \( x_u \) in \( T_{\text{opt}}^{R_u} \) to \( x_u \) and delete the character \( u \) from every node. The resulting arborescence is a valid solution of \( I' \), and hence \( \text{OPT}_{R'} \leq \text{OPT}_R - 1 \).

To prove the other bound \( \text{OPT}_{R'} \geq \text{OPT}_R - 1 \), we add the character \( u \) to every node of \( T_{\text{opt}}^{R_u} \) and set it to 0. Let the resulting arborescence be \( T \). For every terminal \( t \in R_u \), let \( \tilde{t} \) be defined as:

\[
\tilde{t}[v] = \begin{cases} 
0, & \text{if } v = u \\
t[v], & \text{otherwise}
\end{cases}
\]

Let \( \tilde{R}_u = \{\tilde{t} \mid t \in R_u\} \). Note that \( T \) is an optimal Steiner arborescence for the terminal set \( R \cup \tilde{R}_u \setminus R_u \), and the subset \( \tilde{R}_u \) is rooted at \( y_u = \text{LCA}(\tilde{R}_u) \) in \( T \).

Let the character \( w \) change along an edge \( e = (y_u, x) \) in \( T \); and let the subtree \( T_x \) rooted at \( x \) contains a terminal from \( \tilde{R}_u \). Then \( T_x \) does not contain any terminal \( t' \in \tilde{R}_u \), otherwise \( u \) will conflict with \( w \). We do the following modifications to \( T \) to obtain \( T' \):

- Add the edge \((y_u, y_u)\) that changes on \( u \).
- For every edge \( e = (y_u, x) \) in \( T \) such that \( T_x \) contains terminals from \( \tilde{R}_u \), remove the edge \( e \) and add the edge \((y_u, x)\).
- For every node \( x \) in the subtree rooted at \( y_u \), flip \( x[u] \) to 1.

Trivially, \( T' \) is a valid Steiner arborescence on \( R \). \( \square \)

Observe that every character of an instance that has been preprocessed by Lemma 3 is a conflicting character. Hence, the following corollary holds true by Proposition 1.

**Corollary 4.** In every instance preprocessed by Lemma 1 and Lemma 3, if number of conflicting characters \( \leq q_{\text{opt}} \), then \( |R| \leq 2q_{\text{opt}} + 1 \).

### 4.2 Randomized FPT Algorithm for the MSA-DH problem

For this algorithm, we exploit another property of the good characters in Proposition 4, and use Corollary 3 to prove the same.
**Proposition 4.** Let $T_{\text{opt}}^{(0)}$ be an optimal Steiner arborescence of the MSA-DH problem instance $I = \langle u_i, R \rangle$, and let $u_1, \ldots, u_k \in \text{GOOD}(T_{\text{opt}}^{(0)})$. Then, there exists an optimal Steiner arborescence $T_{\text{opt}}^{(\kappa)}$ of $I$ such that $\text{GOOD}(T_{\text{opt}}^{(\kappa)}) = \text{GOOD}(T_{\text{opt}}^{(0)})$, and for every $i \in [1, \kappa]$, $x_{u_i}^{(\kappa)} = \text{LCA}(R_{u_i}^{(\kappa,i)})$. Here $x_{u_i}^{(\kappa)}$ denotes the head of the edge $(e_{u_i}^{(\kappa)} = (x_{u_i}^{(\kappa)}, x_{u_i}^{(\kappa)}'))$ along which the good character $u_i$ changes in $T_{\text{opt}}^{(\kappa)}$, and $R_{u_i}^{(\kappa,i)} = \{ t \in R^{(\kappa,i)} | t[u_i] = 1 \}$, with $R^{(\kappa,i)} = R \cup \bigcup_{j \in [1, i-1]} \{ x_{u_j}^{(\kappa)}, x_{u_j}^{(\kappa)} \}$.

**Proof.** We prove the statement using induction on $\kappa$. For the base case, $R_{\text{opt}}^{(1,1)} = R \implies R_{u_1}^{(1,1)} = R_{u_1}$, and the statement holds true by Corollary 3.

Let $T_{\text{opt}}^{(i-1)}$ be the optimal Steiner arborescence spanning $R$ with the same set of good characters as $T_{\text{opt}}^{(0)}$ such that the above statement holds true for the characters $u_1, \ldots, u_{i-1}$. In $T_{\text{opt}}^{(i-1)}$, let the character $u_i$ change along the edge $e_{u_i}^{(i-1)} = (x_{u_i}^{(i-1)}, x_{u_i}^{(i-1)})$; also define $R_{u_i}^{(i-1,i)}$ as in the proposition statement. Notice that $R_{u_i}^{(i-1,i)}$ is $R$ augmented with possibly other nodes in $T_{\text{opt}}^{(i-1)}$, so $T_{\text{opt}}^{(i-1)}$ is also an optimal solution for spanning $R_{\text{opt}}^{(i-1,i)}$.

We construct $T_{\text{opt}}^{(i)}$ from $T_{\text{opt}}^{(i-1)}$ as follows:

**Case 1:** If $x_{u_i}^{(i-1)} = \text{LCA}(R_{u_i}^{(i-1,i)})$, then $T_{\text{opt}}^{(i)} \leftarrow T_{\text{opt}}^{(i-1)}$.

**Case 2:** Otherwise, $T_{\text{opt}}^{(i)} \leftarrow \text{modified } T_{\text{opt}}^{(i-1)}$ as per Remark 2.

Clearly in both cases, $T_{\text{opt}}^{(i)}$ satisfies the proposed statement for $u_i$ (i.e., $x_{u_i}^{(i)} = \text{LCA}(R_{u_i}^{(i,i)})$), while retaining the same set of good characters, arborescence cost and hence optimality of spanning $R$ as $T_{\text{opt}}^{(i-1)}$ (by Remark 2 and proof of Proposition 2). To complete the proof that $T_{\text{opt}}^{(i)}$ satisfies the proposed statement for every character $u_1, \ldots, u_i$, we prove the claim: $\forall j \in [1, i-1]$, the edge along which $u_j$ changes is the same between $T_{\text{opt}}^{(i)}$ and $T_{\text{opt}}^{(i-1)}$ (thereby implying $R_{\text{opt}}^{(i,j)} = R_{\text{opt}}^{(i-1,j)}$) and hence $x_{u_j}^{(i)} = \text{LCA}(R_{u_j}^{(i,j)})$.

The above claim holds trivially in Case 1. In Case 2, consider the path $P$ in $T_{\text{opt}}^{(i-1)}$ from $x_{u_j}^{(i-1)}$ to $\text{LCA}(R_{u_j}^{(i-1,i)})$, with $v$ being the last character that changes along this path. For every $j \in [1, i-1]$, $u_j$ does not change along any edge in $P$ (otherwise, it contradicts the last node in $P$ being $\text{LCA}(R_{u_j}^{(i-1,i)})$); so, although the positions of $u_i$ and $v$ along $P$ are swapped to construct $T_{\text{opt}}^{(i)}$ in Remark 2, the edge along which $u_j$ changes stays invariant, thus proving the above claim.

4.2.1 Algorithm Outline

Our proposed algorithm iteratively constructs an optimal Steiner arborescence $T_{\text{opt}}$ with one good character (and its associated edge) at a time. The algorithm maintains a partition $P$ of the set of input terminals and other augmented terminals. During an iteration, each subset in $P$ is analyzed to obtain its conflicting characters; and from the union of conflicting characters across all the subsets, the algorithm chooses one
character $u$ uniformly at random. The cardinality of this union is referred to as the “total number of conflicting characters” in this work.

Assume that the random choice $u$ during any iteration leads to a successful outcome, viz., $u \in \text{GOOD}(T_{\text{opt}})$. Then, the edge along which $u$ changes in $T_{\text{opt}}$ is constructed using Proposition 4. This edge should split $T_{\text{opt}}$ into two subtrees, each covering a non-empty subset of the terminals. The terminal set is partitioned into these two subsets (along with the endpoints of the newly added edge) and the entire process is repeated until the total number of conflicting characters is at most $q$. Exact details/pseudocode is in Algorithm 2.

Observe that any subset newly created in an iteration is not preprocessed and hence may contain invariant or isolated characters. So, after the loop terminates, every remaining subset is preprocessed by Lemma 1 and Lemma 3 to remove such characters and the optimal Steiner arborescence of the subset is constructed independently using Algorithm 1. It may be noted here that since by Proposition 1 and Corollary 4, the preprocessed subset with at most $q$ (conflicting) characters has at most $2q+1$ terminals, the optimal Steiner arborescence can be constructed in $\text{FPT}$ time. Analysis of the overall algorithm is in Theorem 2.

Algorithm 2  Randomized Algorithm for the Parameterized MSA-DH problem.

\textbf{Input:} Dimension of the hypercube $m$, terminal set $R$, an integer $q \geq 0$.

\textbf{Parameter:} Penalty on the cost $q$.

\textbf{Output:} An optimal Steiner arborescence $T_{\text{opt}}$.

\textbf{Data Structure:} A partition $\mathcal{P}$ of the set of input and newly augmented terminals.

1. Initialize $\mathcal{P}$ with the trivial partition \{R\}.
2. while total number of conflicting characters $> q$ do
3.   From the union of conflicting characters across all subsets in $\mathcal{P}$, choose a character $u$ uniformly at random. Let $S = \{t \in S | t[u] = 1\}$ and $S_u = S \setminus \{S_u\}$. Let $x_u \leftarrow \text{LCA}(S_u)$.
4.   Add edge $e_u = (x_u, x_u)$, which changes on $u$, to $T$.
5.   Update $\mathcal{P}$ as $\mathcal{P} \leftarrow \mathcal{P} \cup \{S_u \cup \{x_u\}, S_u \cup \{x_u\}\} \setminus \{S\}$.
6. if number of iterations $> q$ then
7.   return error
8. end if
9. end while
10. for all remaining subsets $S$ in $\mathcal{P}$ do
11.   Preprocess $S$ by Lemma 1 and Lemma 3.
12. Use Algorithm 1 to construct its optimal Steiner arborescence $T_{S_{\text{opt}}}$.
13. $T \leftarrow T \cup T_{S_{\text{opt}}}$.
14. end for

\textbf{Theorem 2.} The parameterized MSA-DH problem can be solved in $O\left(9^q q^3 + q|R|m^2\right)$ time with success probability $\geq 4^{-q}$.

\textbf{Proof.} Consider an optimal Steiner arborescence $T_{\text{opt}}$. By Proposition 1, there must be a conflicting character that is good in $T_{\text{opt}}$ if the loop is invoked. Assume that the
algorithm’s randomly chosen characters \( u_1, \ldots, u_\kappa \) are good characters of \( T_{\text{opt}} \) (success probability has been calculated below). Then, by Proposition 4, there is an optimal Steiner arborescence \( T_{\text{opt}}^{(\kappa)} \) with the same set of good characters as \( T_{\text{opt}} \) and in which \( \forall i \in [1, \kappa], x_{u_i}^{(\kappa)} = \text{LCA}(R_{u_i}^{(\kappa,i)}) \). Recall that the good character \( u_i \) in \( T_{\text{opt}}^{(\kappa)} \) changes along the edge \( e_{u_i}^{(\kappa)} = (x_{u_i}^{(\kappa)}, x_{u_i}^{(\kappa)}) \).

**Optimality:** Under the assumption stated above that all chosen characters \( u_1, \ldots, u_\kappa \) are good characters of \( T_{\text{opt}} \) or equivalently \( T_{\text{opt}}^{(\kappa)} \), the character \( u_i \), chosen during iteration \( i \), will be conflicting in exactly one subset \( S \) and invariant in all other subsets. Then \( \text{OPT}_S = \text{OPT}_{S_u \cup \{x_{u_i}^{(\kappa)}\}} + \text{OPT}_{S_u \cup \{x_{u_i}^{(\kappa)}\}} + 1 \). This validates dividing the subproblem \( S \) into two other subproblems using \( x_{u_i} \).

To find \( x_{u_i} \), note that \( \text{LCA}(S_u) = \text{LCA}(R_{u_i}^{(\kappa,i)}) \) (because \( R_{u_i}^{(\kappa,i)} \) is a set of input/au-
gmented terminals in the subtree \( T' \) of \( T_{\text{opt}}^{(\kappa)} \) rooted at \( x_{u_i}^{(\kappa)} \), and \( S_u \subseteq R_{u_i}^{(\kappa,i)} \) is the set of input/augmented terminals in \( T' \) after pruning subtrees in \( T' \), none of which affect the LCA computation; in detail, \( \forall j \in [1,i-1] \) such that \( x_{u_i} \) is a descendant of \( x_{u_i} \) in \( T_{\text{opt}}^{(\kappa)} \), the pruning removes the subtree of \( T' \) rooted at \( x_{u_j} \) successively).

Thus, the algorithm generates \( T_{\text{opt}}^{(\kappa)} \), if the chosen vertices are all good characters of \( T_{\text{opt}} \), \( x_{u_i}^{(\kappa)} \) is set to \( \text{LCA}(S_u) \) and an exact algorithm is used to construct an optimal Steiner arborescence on each of the remaining sub-

**Loop Convergence:** \([2, 23]\) Let the subtrees spanning \( S, S_u \cup \{x_{u_i}^{(\kappa)}\} \) and \( S_u \cup \{x_{u_i}^{(\kappa)}\} \) in \( T_{\text{opt}}^{(\kappa)} \) be \( T, T_u \) and \( T_u \), respectively. Since \( u_i \) is a conflicting character, there exists at least one character \( v \) that conflicts with \( u_i \) in \( S \); and thus \( v \) changes at least once in both \( T_u \) and \( T_u \). If \( q(v), q_u(v) \text{ and } q_{u_i}(v) \) denotes the penalty incurred due to \( v \) (i.e., number of edges along which \( v \) changes -1) in \( T, T_u \) and \( T_u \), respectively, then \( (q_{u_i}(v) + 1) + (q_u(v) + 1) = (q(v) + 1) \). This implies \( q_{u_i} + q_{u_i} + 1 \leq q_T \).

So if all choices are correct, we are reducing the total penalty (sum of penalty of subtrees of \( T_{\text{opt}}^{(\kappa)} \) spanning each subset in the current partition) by at least 1 with each iteration. Given that \( q_{\text{opt}} \leq q \), there can be at most \( q \) iterations (otherwise, total penalty will become 0, and there will be no conflicting characters).

**Success Probability:** We now compute the probability that all the chosen characters are good characters of \( T_{\text{opt}} \). Let the loop runs \( \kappa \) times (\( \kappa \leq q \)). Then before the start of the first iteration there are at least \( q + \kappa \) conflicting characters, out of which at most \( q \) are bad characters in \( T_{\text{opt}} \). With each iteration, we are reducing the number of characters by at least 1, implying that before the start of \( j \)th iteration, there must be at least \( (q + \kappa) - (j - 1) \) conflicting characters. Thus,

\[
\Pr[u_j \in \text{GOOD}(T_{\text{opt}}) \mid u_j-1, \ldots, u_1 \in \text{GOOD}(T_{\text{opt}})] \geq 1 - \frac{q}{q + \kappa - (j - 1)} = \frac{\kappa - j + 1}{q + \kappa - j + 1}
\]

\[
\Pr[\text{all chosen characters are good}] = \Pr[u_\kappa, \ldots, u_1 \in \text{GOOD}(T_{\text{opt}})] = \Pr[u_\kappa \in \text{GOOD}(T_{\text{opt}}) \mid u_{k-1}, \ldots, u_1 \in \text{GOOD}(T_{\text{opt}})] \ldots \Pr[u_1 \in \text{GOOD}(T_{\text{opt}})]
\]

\[
\geq \prod_{j=1}^{\kappa} \frac{\kappa - (j - 1)}{q + \kappa - (j - 1)} = \frac{\kappa!q!}{(q + \kappa)!} = \frac{1}{2^{q+\kappa}} \geq \frac{1}{4^q}
\]
Time Complexity: Every run of the iteration takes $O(|R|m^2)$ time. In each iteration we are partitioning exactly one subset into two, and we have at most $q$ iterations. Hence we can have at most $q + 1$ subsets remaining at the termination of the iteration. In any preprocessed subset $S$, there are at most $q$ conflicting characters (termination condition of loop) and hence at most $2q + 1$ terminals (by Corollary 4). So, Algorithm 1 takes $O(3^{2q+1}(2q + 1)q + ((2q + 1)q)^2) = O(9q^2)$ time. Thus, construction of the optimal Steiner arborescence on all the remaining subsets require $O(9q^2q + ((2q + 1)q)^2)$ time.

4.2.2 Derandomization

To derandomize Algorithm 2, we propose a method inspired from the derandomization algorithm for the undirected version of the problem [2, 23]. Consider an optimal Steiner arborescence $T_{\text{opt}}$ for an instance $I$, preprocessed by Lemma 1 and Lemma 3. Every conflicting character in $I$ is either a good character or a bad character in $T_{\text{opt}}$ – our proposed algorithm uses a bounded search tree that branches on these two choices up to a maximum depth of $2q$. This algorithm’s pseudocode is in Algorithm 3 and analysis is in Lemma 4.

**Lemma 4.** The MSA-DH problem can be solved in $\tilde{O}(36^q)$ time.

**Proof.** 

**Correctness:** The algorithm cannot return a Steiner arborescence of cost at most $m + q$ for a no-instance. So it suffices to consider a yes-instance $I$, and an optimal arborescence $T_{\text{opt}}$ of $I$. Call a node $N$ in the depth-bounded search tree (DBST) as “correct” if the algorithm’s choices so far until this node are correct, i.e., if $G(N) \subseteq \text{GOOD}(T_{\text{opt}})$ and $B(N) \subseteq \text{BAD}(T_{\text{opt}})$. Then, the algorithm is correct if there exists a “correct” leaf node $N'$ in the DBST that is also marked “valid” – the arborescence constructed at this node will yield the desired solution with cost at most $m + q$.

We first prove the existence of a correct leaf node. The root node of the DBST is trivially correct. If the algorithm doesn’t split $N$ further, then $N$ itself is the correct leaf. Otherwise, let $u$ be the conflicting character chosen by the algorithm to split $N$. Since $u$ is either a good or a bad character in $T_{\text{opt}}$, and we branch out and explore both these options in the DBST, one of the two children ($N_c$) of $N$ is also correct. Repeating this argument (for at most $2q$ iterations), the correct leaf is either $N_c$ itself, or a descendant of $N_c$. This proof also shows that each ancestor node of this correct leaf is also correct.

We now show that the correct leaf, denoted $N'$, is marked valid in Algorithm 3, Line 6. Since $B(N') \subseteq \text{BAD}(T_{\text{opt}})$, $|B(N')| \leq q$. Since $G(N') \subseteq \text{GOOD}(T_{\text{opt}})$, we can prove that $|G(N')| \leq q$ by contradiction as follows. Assume $|G(N')| > q$, and let $N''$ be the ancestor of $N'$ such that $|G(N'')| = q$. Then, the total number of conflicting characters will have become 0 at $N''$ (cf. “Loop Convergence” section in the proof of Theorem 2), and hence $N''$ would not have been split further. Finally, we claim the total number of conflicting characters at $N'$ is at most $q$ – otherwise, $N'$ would’ve been split further. The depth bound of $2q$ won’t prevent this split, as the algorithm chooses $q$ bad characters of $T_{\text{opt}}$ in the worst case and has at least $q$ other iterations
Algorithm 3 Deterministic Algorithm for the Parameterized MSA-DH problem.

Input: Dimension of the hypercube $m$, terminal set $R$, an integer $q \geq 0$.

Parameter: Penalty on the cost $q$.

Output: An optimal Steiner arborescence $T_{opt}$.

Data Structure: A depth-bounded search tree (DBST) rooted at $N_0$. Every node $N$ in the DBST stores a partition $\mathcal{P}(N)$ of the set of input terminals and other augmented terminals; and two sets for bookkeeping $- G(N)$ and $B(N)$. Every leaf node $N$ in the DBST is marked either “valid” or “pruned” at the end of the algorithm.

1: $\mathcal{P}(N_0) \leftarrow \{R\}$, $G(N_0) \leftarrow \phi$ and $B(N_0) \leftarrow \phi$.
2: while there is an unmarked leaf node $N$ at a depth $\leq 2q$ do
   3: if $|G(N)| > q$ or $|B(N)| > q$ then
      4: Mark $N$ as “pruned”.
   5: else if total number of conflicting characters $\leq q$ then
      6: Mark $N$ as “valid”.
   7: else if depth of the node $N$ is $2q$ then
      8: Mark $N$ as “pruned”.
   9: else
      10: Choose a conflicting character $u$ arbitrarily and let $S \in \mathcal{P}(N)$ be a subset in which $u$ is conflicting.
      11: Branch $N$ into two unmarked nodes $N_1$ and $N_2$.
      12: At $N_1$, $G(N_1) \leftarrow G(N)$ and $B(N_1) \leftarrow B(N) \cup \{u\}$ and $\mathcal{P}(N_1) \leftarrow \mathcal{P}(N)$.
      13: At $N_2$, $G(N_2) \leftarrow G(N) \cup \{u\}$, $B(N_2) \leftarrow B(N)$ and $\mathcal{P}(N_2) \leftarrow \mathcal{P}(N) \cup \{S_u \cup \{x_u\}, S_u \cup \{x_u\} \setminus \{S\}\} \text{ (cf. Algorithm 2 Lines 4–6).}$
      14: end if
   15: end while
16: for every leaf node $N$ marked as “valid” do
   17: Construct a Steiner arborescence as per Algorithm 2 Lines 11–15.
18: end for
19: if there are no “valid” leaf nodes or every arborescence has cost $> m + q$ then
   20: return No-Instance.
21: else
   22: return the Steiner arborescence with minimum cost.
23: end if

to choose a sufficient number of good characters to bring down the total number of conflicting characters to at most $q$.

Time Complexity: As the search tree has a maximum depth of $2q$ and branches into exactly two nodes from every node, there are at most $2^{2q+1} - 1$ nodes in the tree and each node takes $O(|R|m^2)$ time. There can be at most $2^{2q}$ leaf nodes in the search tree, each consisting of at most $q + 1$ subsets in its partition. Further, in a “valid” leaf’s partition, every subset $S$ has at most $q$ conflicting characters and therefore at most $2q + 1$ terminals. Thus, Algorithm 1 takes $O(3^{2q+1}(q+1))$ time to construct the optimal Steiner arborescences of the subsets in a valid leaf’s partition.

17
4.3 Parameterized Approximation based on Vertex Cover

To design an approximation algorithm related to the optimal penalty $q_{\text{opt}}$, we first explore a lower bound on $q_{\text{opt}}$ using the minimum vertex cover of the conflict graph.

4.3.1 A Relationship between Penalty and Minimum Vertex Cover

Observe that if characters $u$ and $v$ conflict with each other in $R$, then in every Steiner arborescence $T$, $u$ and $v$ both cannot be good characters of $T$. We use this to relate the number of bad characters in any arborescence $T$ to the vertex cover of the conflict graph in Proposition 5.

**Proposition 5.** For every Steiner arborescence $T$ of $R$, $\text{BAD}(T)$ is a vertex cover of $CG(R)$. Further, for every minimal vertex cover $\text{VC}$ of $CG(R)$, there is a Steiner arborescence $T$ such that $\text{BAD}(T) = \text{VC}$.

Proof. Let $\{u, v\} \in E(CG(R))$. Then there are terminals $t_1, t_2, t_3$ such that $t_1[u] = 0 \land t_1[v] = 1, t_2[u] = 1 \land t_2[v] = 0$ and $t_3[u] = 1 \land t_3[v] = 1$. Let $u$ change exactly once in $T$, along the edge $e_u = (x_u, x_u)$. Let $T_u$ be the subtree rooted at $x_u$, and $T_u$ be the subtree obtained after deleting $e_u$ and $T_u$. Because $t_2$ is in $T_u$, the character $v$ cannot change along the path $P$ from $0^m$ to $x_u$. So, to span $t_1$ and $t_3$, there must be edges $e_1 = (x_1, y_1)$ and $e_2 = (x_2, y_2)$ along which $v$ changes in $T_u \setminus P$ and $T_u$ respectively. Also there must be some node $y$ along the path from $x_u$ to $x_2$ such that the subtree rooted at $y$ contains $t_2$. Thus if $u$ is a good character, then $v$ is a bad character.

For the second part of the claim, consider the independent set formed after removing VC i.e., characters $[m'] = [m] \setminus \text{VC}$. The arborescence on the terminals projected onto the independent set $[m']$ can be constructed in polynomial time by Corollary 2. Then, in this arborescence, the characters corresponding to $\text{VC}$ are added back to each node (both Steiner and terminal) initially with a value 0, and the arborescence is changed as necessary (by including relevant nodes/edges) to span $R$. Since every character in the minimal vertex cover conflicts with at least one character in the independent set and every character in the independent set is a good character, we can use the same proof as above to show that each of these $\text{VC}$ characters are bad in the resulting arborescence.

Due to Proposition 5 and as $|\text{BAD}(T_{\text{opt}})| \leq q_{\text{opt}}$, we get the following corollary.

**Corollary 5.** The optimal penalty $q_{\text{opt}} \geq \tau$, where $\tau$ is the size of the minimum vertex cover of the conflict graph.

4.3.2 Algorithm Outline

By Proposition 5, there is a Steiner arborescence $T$ such that the set of bad characters of $T$ is a minimum vertex cover $\text{MVC}$ of the conflict graph $CG(R)$. We present a parameterized approximation algorithm that attempts to construct such an arborescence. The main idea behind our proposed algorithm is to delete the characters in $\text{MVC}$ from the input instance $I$, and thus construct the optimal Steiner arborescence on the new instance $I'$ in polynomial time. Finally, the removed characters are added back. The pseudocode is given in Algorithm 4 and is analyzed in Theorem 3.
**Algorithm 4** Parameterized Approximation Algorithm for the Parameterized MSA-DH problem.

**Input:** Dimension of the hypercube $m$, terminal set $R$, an integer $q \geq 0$.

**Parameter:** Penalty on the cost $q$.

**Output:** A near-optimal Steiner arborescence $T$.

1. Construct $CG(R)$ and find its minimum vertex cover $MVC$.
2. if size of $MVC$ $\tau > q$ then
3.   return No-Instance
4. end if
5. Reduce $I$ to the instance $I' = \langle m', R' \rangle$, where $[m'] = [m] \setminus MVC$ and $R' = \{t' | \exists t \in R \forall u \in [m'] t'[u] = t[u]\}$.
6. Use Corollary 2 to compute the optimal solution $T$ of $I'$.
7. $\forall t \in V(T)$ add the MVC characters back and set them to 0.
8. for all $t \in R$ do
9.   Find $t_a$ such that $t_a = \arg \min_{t' \in V(T) \cap \text{Ancestor}(t)} d(t', t)$.
10. Add any path $P(t_a, t)$ from $t_a$ to $t$ in $Q_m$.
11. end for

**Theorem 3.** There is a $(1 + q)$-approximation algorithm for the parameterized MSA-DH problem that runs in $\tilde{O}(1.25284^q)$ time, where $\tilde{O}$ hides the polynomial factors in the input size.

**Proof.** By Corollary 5, if $q < \tau$ then $q < q_{opt}$; clearly this is a No-Instance. But when $\tau \leq q < q_{opt}$ the output of the algorithm is arbitrary. Henceforth, we assume $q \geq q_{opt}$ to prove the approximation guarantees of the output Steiner arborescence.

Let the vertex cover $VC$ of $CG(R)$ used to construct $T$ be an $\alpha$-approximation of the minimum vertex cover $MVC$. The skeleton of $T$ will consist of an optimal Steiner arborescence on $I'$, and each terminal shall be connected to its respective ancestor. The structure of the output has been depicted in Figure 2.

![Fig. 2: Structure of the output Steiner arborescence SOL. Triangles represent subtrees while circles represent nodes in the arborescence. Dotted arrows represent paths.](image-url)
Let the optimal Steiner arborescence of \( I' \) be \( T_{opt}^{R'} \). Thus, cost of the output arborescence, \( SOL \), is the sum of \( \text{COST}(T_{opt}^{R'}) \) and the total cost of all paths to the terminals from their corresponding ancestors in \( T_{opt}^{R'} \). Now, \( \text{COST}(T_{opt}^{R'}) = m' = m - |VC| \).

\[
\text{SOL} = m - |VC| + \sum_{t \in R} |P(t_a,t)| = m - |VC| + \sum_{t \in R} d(t_a,t) \leq m - |VC| + \sum_{t \in R} |VC| \\
\leq m - |VC| + |VC||R| = m + |VC||R| - 1 \leq OPT + \alpha \tau \times OPT \\
= (1 + \alpha \tau)OPT \leq (1 + \alpha q)OPT \quad \text{(from Corollary 5)}
\]

If we use the FPT\([q]\)-algorithm to find an exact minimum vertex cover \([5, 17, 22]\), then \( \alpha = 1 \). While construction of the conflict graph takes \( \mathcal{O}(|R|m^2) \) time, computing the minimum vertex cover can be achieved in \( \tilde{\mathcal{O}}(1.25284^q) \) \([17]\). All the remaining parts of the algorithm can be performed in \( \mathcal{O}(|R|m) \) time. Thus the theorem holds true. \( \square \)

**Remark 3.** If we use the 2-approximation algorithm for the minimum vertex cover problem \([24]\), then the algorithm outputs a \((1 + 2\text{opt})\)-approximation Steiner arborescence in polynomial time.

## 5 Parameterized Approximation Algorithm on Number of Steiner Nodes

In this section, we present an algorithm for parameter \( p \) that involves finding a minimum set of Steiner nodes recursively in a level that is adjacent to the subset of nodes of interest in the immediate higher level. We formulate computing this set as the minimum hitting set (MHS) problem.

For some level \( \ell \geq 2 \), let \( S_\ell \subseteq \hat{Q}_m^\ell \). Let \( S_{\ell-1} \subseteq \hat{Q}_{m}^{\ell-1} \) be a minimum set of nodes such that \( \forall t \in S_\ell \text{ \ parent}(t) \cap S_{\ell-1} \neq \emptyset \). Consider the collection of sets \( \mathcal{F} = \{\text{parent}(t) \mid t \in S_\ell\} \) over the universe \( U = \bigcup_{t \in S_\ell} \text{parent}(t) \). Clearly, \( S_{\ell-1} \) is a minimum hitting set of \( \mathcal{F} \), and is represented as \( S_{\ell-1} = \text{MHS}(\mathcal{F}) \). We abuse the notation slightly and use \( \text{MHS}(S_\ell) \) to represent \( \text{MHS}(\mathcal{F}) \). Notice that, since they are at the same level in \( \hat{Q}_m^\ell \), every node in \( S_\ell \) has exactly \( \ell \) parents; and hence, to compute \( S_{\ell-1} \), we can use the FPT\([\ell, k]\) algorithm for the \( \ell \)-MHS problem \([22]\) (every set in the input set system is of size \( \ell \) and \( k \) is the bound on the solution size). So, if \( R_\ell \) is a subset of terminals at level \( \ell \), we can recursively apply the FPT\([\ell, p]\) algorithm for the \( \ell \)-MHS problem and construct a Steiner arborescence on \( R_\ell \).

Our proposed algorithm exploits this idea. The set of all non-root terminals \( (R \setminus \{0^m\}) \) are partitioned based on their levels and a Steiner arborescence is constructed for each partition independently. Finally, the union of all the constructed arborescences is returned. For any partition \( R_\ell \) of the terminal set, the arborescence is constructed using the following recurrence relation:

\[
S'_\ell = \begin{cases} 
\text{MHS}(R'_\ell), & \text{if } i = \ell - 1 \\
\text{MHS}(S'_{\ell+1}), & \text{otherwise}
\end{cases}
\] (2)
Here, \( R'_i \subseteq R_i \) is the subset of terminals at level \( \ell \) without any parents in the terminal set \( R \) and \( S^\ell_i \) represents the set of Steiner nodes at level \( i \) in the Steiner arborescence spanning \( R'_i \). All the terminals in \( X_\ell = R_\ell \setminus R'_\ell \) contain a parent in \( R_\ell-1 \), and hence are directly added as children to their respective parents. The pseudocode is given in Algorithm 5 and analyzed in Theorem 4.

**Algorithm 5** Parameterized Approximation algorithm for the Parameterized MSA-DH problem.

**Input:** Dimension of the hypercube \( m \), terminal set \( R \), an integer \( p \geq 0 \).

**Parameter:** Number of Steiner nodes \( p \)

**Output:** A near-optimal Steiner arborescence \( T \).

1. Partition \( R \) into \( R_1, R_2, \ldots, R_{\ell_{\max}} \).
2. Add \( R_1 \) as children of \( 0^m \) in \( T \).
3. for all \( \ell \in [2, \ell_{\max}] \) do

   4. Let \( X_\ell = \{ t \in R_\ell \mid \text{parent}(t) \cap R_{\ell-1} \neq \emptyset \} \) and \( R'_\ell = R_\ell \setminus X_\ell \).
   5. \( \forall t \in X_\ell \), add an edge \((t', t)\) to \( T \) for some \( t' \in \text{parent}(t) \cap R_{\ell-1} \).
   6. Add \( R'_\ell \) to \( T_\ell \) and \( S^\ell_i \leftarrow R'_\ell \).
   7. for \( i \in [\ell - 1, 1] \) do

      8. Compute \( S^\ell_i \) using Equation 2 with solution size at most \( |S_i^\ell| \).
      9. \( \forall t \in S^\ell_{i+1} \), add an edge \((t', t)\) to \( T_\ell \) for some \( t' \in \text{parent}(t) \cap S^\ell_i \).
   10. if \( |S^\ell_{\ell-1}| > p \) then

        11. return No-Instance
   12. end if
   13. end for
14. \( T \leftarrow T \cup T_\ell \) (here we find the union of the edges and prune the extra nodes/edges, so that the resulting arborescence is valid).
15. end for

**Theorem 4.** There is a \( O(p^{\ell_{\max}}) \)-additive approximation algorithm for the parameterized MSA-DH problem that runs in \( O(p^{\ell_{\max}} + \ell_{\max}|R|^2m) \) time, where \( \ell_{\max} \) is the maximum distance of any terminal from the root.

**Proof.** We first prove that if \( p \geq p_{\text{opt}} \), then \( \forall \ell \in [2, \ell_{\max}] \), \( |S^\ell_{\ell-1}| = |\text{MHS}(R'_\ell)| \leq p \). Let \( p_{\text{opt}}^i \) be the number of Steiner nodes at level \( i \) in \( T_{\text{opt}} \). So \( \sum_{i=1}^{\ell_{\max}-1} p_{\text{opt}}^i = p_{\text{opt}} \). For any level \( \ell \in [2, \ell_{\max}] \), since every terminal \( t \in R'_\ell \) does not have a parent in \( R_{\ell-1} \), the parent of \( t \) in \( T_{\text{opt}} \) is a Steiner node. Also the set of Steiner nodes spanning \( R'_\ell \) in \( T_{\text{opt}} \) is a hitting set. Hence \( |\text{MHS}(R'_\ell)| \leq p_{\ell-1} \leq p_{\text{opt}} \leq p \). Thus, if \( \exists \ell \in [2, \ell_{\max}] \), \( |\text{MHS}(R'_\ell)| > p \), then it is a No-Instance. It may be that \( \forall \ell \in [2, \ell_{\max}] \), \( |\text{MHS}(R'_\ell)| \leq p \) even when \( p < p_{\text{opt}} \) in which case the output is arbitrary. Henceforth we assume \( p \geq p_{\text{opt}} \) to prove the approximation guarantee of the algorithm.

We first note that when the algorithm is constructing the arborescence on any level \( \ell \in [2, \ell_{\max}] \), \( R_{\ell-1} \) is already in \( T \). So every terminal \( t \in X_\ell \) already has a parent in \( T \) and \( T_\ell \) is constructed only on \( R'_\ell \). Also, \( \forall i \in [1, \ell - 1] \), \( |S^\ell_i| \leq |S^\ell_{i+1}| \). If we are using an \( \alpha \)-approximation algorithm for the \( \ell \)-MHS problem, the total number of Steiner
nodes in $T_\ell$ can be computed as:

$$p_{T_\ell} = \sum_{i=\ell-1}^{1} |S^\ell_i| \leq \sum_{i=\ell-1}^{1} \alpha |\text{MHS}(S^\ell_{i+1})| \leq \sum_{i=\ell-1}^{1} \alpha |\text{MHS}(R^\ell_i)| \leq \sum_{i=\ell-1}^{1} \alpha p^{\ell-1}_\text{opt} = (\ell - 1)\alpha p^{\ell-1}_\text{opt}$$

Thus, the cost of the output arborescence is:

$$SOL = |R| - 1 + p_T \leq OPT + \sum_{\ell=2}^{\ell_{\text{max}}} p_{T_\ell} \leq OPT + \sum_{\ell=2}^{\ell_{\text{max}}} (\ell - 1)\alpha p^{\ell-1}_\text{opt}$$

$$\leq OPT + (\ell_{\text{max}} - 1)\alpha \sum_{\ell=1}^{\ell_{\text{max}}-1} p^{\ell}_\text{opt} = OPT + (\ell_{\text{max}} - 1)\alpha p^{\ell}_{\text{opt}}$$

$$\leq OPT + (\ell_{\text{max}} - 1)\alpha p$$

If we use the exact algorithm for the $\ell$-MHS problem [22], then $\alpha = 1$ and $SOL \leq OPT + (\ell_{\text{max}} - 1)p$. Now, the time taken to compute $T_\ell$ is the total time to run the FPT[$\ell$, $p$] algorithm (depth-bounded search tree algorithm [22]) $\ell$ times and the time taken to add the edges i.e., $O(\ell |p_r^\ell + |R^\ell_i|^2m|)$. This is performed $\ell_{\text{max}} - 1$ times, so the total time taken is $O(\ell_{\text{max}}^2 + \ell_{\text{max}}^2 |R|^2m)$. $\square$

**Remark 4.** If we use the $\ln |R|$-approximation algorithm for the MHS problem [24], then the algorithm generates a $O(p^{\text{opt}}\ell_{\text{max}} \ln |R|)$-additive approximation Steiner arborescence in polynomial time. Similarly, if we use the $\ell$-approximation algorithm for the $\ell$-MHS problem [25], then the algorithm generates a $O(p^{\text{opt}}\ell_{\text{max}}^2)$-additive approximation Steiner arborescence.

### 6 Conclusion and Future Work

We explored the MSA-DH problem from different viewpoints of parameterization and presented FPT algorithms for the same. Specifically, we showed that the problem is in FPT on the number of terminals $|R|$ and penalty $q$. We also present a parameterized approximation algorithm for the parameter $q$ that can be improved to run in polynomial time at the expense of the approximation factor, and another parameterized approximation algorithm involving parameters $p$ and $\ell_{\text{max}}$. The major key point of all our algorithms is the polynomial dependence on $m$, and hence the logarithmic dependence on the size of the underlying graph. All the algorithms exploit the nature of the directed hypercube to accomplish this motive.

Future work could focus on the open questions of whether parameterized algorithms with better running times or approximation ratios are possible for these problems (e.g., can the exponential factor in the time complexity of the FPT[$|R|$] algorithm be improved from $3^{|R|}$ to $2^{|R|}$?). The W-hardness of the problem on the parameter $p$, remains an open question (i.e., is there an FPT[$p$] algorithm that is independent of $\ell_{\text{max}}$?).

The Steiner arborescence problem and algorithms presented here also opens up interesting directions/applications in other fields. Steiner arborescence may provide a
data structure to represent an ensemble of related graphs (using bit vectors that indicate the presence/absence of edges in each graph), and thereby help efficiently compute certain properties of these graphs. An alternative representation of the terminal set as a set system may be used to explore connections to VC (Vapnik–Chervonenkis) dimension. For instance, it will be interesting to explore if the cost of the Steiner arborescence can be exploited to compute the VC dimension of a set family.

Acknowledgements. This work was supported in part by the Wellcome Trust/DBT India Alliance Intermediate Fellowship IA/I/17/2/503323 awarded to Manikandan Narayanan. The authors thank Dr. Vijayaragunathan Ramamoorthi for his invaluable suggestions and contribution towards the completion of this work.

Author contribution. This work is performed as part of the doctoral thesis of Sugyani Mahapatra with extensive inputs from Manikandan Narayanan and N. S. Narayanaswamy.

References

[1] Awasthi P, Blum A, Morgenstern J, et al (2012) Additive approximation for near-perfect phylogeny construction. In: Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques. Springer, p 25–36

[2] Blelloch GE, Dhamdhere K, Halperin E, et al (2006) Fixed parameter tractability of binary near-perfect phylogenetic tree reconstruction. In: International Colloquium on Automata, Languages, and Programming, Springer, pp 667–678

[3] Book RV (1980) Michael R. Garey and David S. Johnson, Computers and intractability: A guide to the theory of NP-completeness. American Mathematical Society

[4] Camin JH, Sokal RR (1965) A method for deducing branching sequences in phylogeny. Evolution 19(3):311–326

[5] Cygan M, Fomin FV, Kowalik L, et al (2015) Parameterized Algorithms, vol 4. Springer

[6] Day WH, Johnson DS, Sankoff D (1986) The computational complexity of inferring rooted phylogenies by parsimony. Mathematical Biosciences 81(1):33–42

[7] Ding Z, Filkov V, Gusfield D (2005) A linear-time algorithm for the perfect phylogeny haplotyping (pph) problem. In: Annual International Conference on Research in Computational Molecular Biology, Springer, pp 585–600

[8] Ding Z, Filkov V, Gusfield D (2006) A linear-time algorithm for the perfect phylogeny haplotyping (pph) problem. Journal of Computational Biology 13(2):522–553
[9] Dreyfus SE, Wagner RA (1971) The Steiner problem in graphs. Networks 1(3):195–207

[10] Dvorák P, Feldmann AE, Knop D, et al (2021) Parameterized approximation schemes for Steiner trees with small number of Steiner vertices. SIAM Journal on Discrete Mathematics 35(1):546–574

[11] Felsenstein J (2004) Inferring Phylogenies, vol 2. Sinauer Associates Sunderland, MA

[12] Fernández-Baca D, Lagergren J (1998) On the approximability of the Steiner tree problem in phylogeny. Discrete Applied Mathematics 88(1-3):129–145

[13] Foulds LR, Graham RL (1982) The Steiner problem in phylogeny is NP-complete. Advances in Applied Mathematics 3(1):43–49

[14] Goranci G, Räcke H, Saranurak T, et al (2021) The expander hierarchy and its applications to dynamic graph algorithms. In: Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA), SIAM, pp 2212–2228

[15] Gusfield D, Bansal V (2005) A fundamental decomposition theory for phylogenetic networks and incompatible characters. In: Annual International Conference on Research in Computational Molecular Biology, Springer, pp 217–232

[16] Halperin E, Krauthgamer R (2003) Polylogarithmic inapproximability. In: Proceedings of the thirty-fifth annual ACM symposium on Theory of computing, pp 585–594

[17] Harris DG, Narayanaswamy N (2024) A faster algorithm for vertex cover parameterized by solution size. In: 41st International Symposium on Theoretical Aspects of Computer Science (STACS 2024), Schloss Dagstuhl-Leibniz-Zentrum für Informatik

[18] Hsu TS, Tsai KH, Wang DW, et al (2005) Two variations of the minimum Steiner problem. Journal of Combinatorial Optimization 9(1):101–120

[19] Jansson J (2014) Directed perfect phylogeny (binary characters). Encyclopedia of Algorithms pp 1–5

[20] Lam F, Tarpine R, Istrail S (2010) The imperfect ancestral recombination graph reconstruction problem: upper bounds for recombination and homoplasy. Journal of Computational Biology 17(6):767–781

[21] Nederlof J (2013) Fast polynomial-space algorithms using inclusion-exclusion: Improving on Steiner tree and related problems. Algorithmica 65(4):868–884

[22] Niedermeier R (2006) Invitation to fixed-parameter algorithms, vol 31. OUP Oxford
[23] Sridhar S, Dhamdhere K, Blelloch G, et al (2007) Algorithms for efficient near-perfect phylogenetic tree reconstruction in theory and practice. IEEE/ACM Transactions on Computational Biology And Bioinformatics 4(4):561–571

[24] Vazirani VV (2013) Approximation Algorithms. Springer Science & Business Media

[25] Williamson DP, Shmoys DB (2011) The design of approximation algorithms. Cambridge university press

[26] Zelikovsky A (1997) A series of approximation algorithms for the acyclic directed Steiner tree problem. Algorithmica 18(1):99–110