

# THE COMPUTATIONAL COMPLEXITY OF CALCULATING PARTITION FUNCTIONS OF OPTIMAL MEDIANS WITH HAMMING DISTANCE

ISTVÁN MIKLÓS<sup>1,2</sup> AND HEATHER SMITH<sup>3,†</sup>

<sup>1</sup> MTA Rényi Institute, Reáltanoda u. 13-15, 1053 Budapest, Hungary

<sup>2</sup> MTA SZTAKI, Lágymányosi u. 11, 1111 Budapest, Hungary

<sup>3</sup> Department of Mathematics, University of South Carolina, Columbia, SC 29208, USA

**ABSTRACT.** Single Cut-or-Join is, computationally, the simplest mathematical model of genome rearrangement. Given a collection of species, we fix a tree which represents their ancestral histories, labeling the leaves with the observed genomes. To assess the likelihood of an assignment of ancestors, we use the parsimony criterion. For each most parsimonious labeling of the ancestors, we relate each pair of vertices along a common edge with an Single Cut-or-Join scenario to create a most parsimonious SCJ scenario.

Here, we determine that the problem of counting the number of most parsimonious SCJ scenarios is #P-complete when the ancestral tree is a star or a binary tree. Further, we extend the result for star trees to analogous questions, weighting each most parsimonious labeling in various ways in the summation over all most parsimonious labelings.

## 1. GENOME REARRANGEMENT

A *genome* is represented by an edge-labeled digraph where each vertex has total degree at most two. Each edge represents a *gene*. Each gene has a head and a tail, which will be collectively referred to as *extremities*. The vertices of the genome can be viewed as partition classes of the gene extremities such that classes have size at most two. Vertices of size two are called *adjacencies* while vertices of size one are called *telomeres*. A genome, on a specified set of genes, is characterized by its list of adjacencies. Given a multiset of genomes, each having the same set of genes, we can use binary strings to encode the genomes, assigning one coordinate to each adjacency that appears in at least one of the genomes.

For a multiset of observed genomes with the same genes, we use a tree to represent the phylogenetic history, labeling the leaves with the observed genomes. The most ancient vertex in the tree can be considered the root, a common ancestor of the sequences observed in the leaves. Each internal vertex represents an unknown species. We are interested in determining the most likely ancestors according to some criterion.

Along each edge of the tree, subsequent mutations occurred to transform the genome of the ancestor into the genome of its descendant. To describe these mutations, we use the Single Cut-or-Join model.

**Definition 1.1** (Feijão and Meidanis [4]). *A Single Cut or Join (SCJ) operation transforms one genome into another genome by altering the set of adjacencies in one of the following ways:*

- *replace adjacency  $u = \{x, y\}$  with telomeres  $u_1 = \{x\}$  and  $u_2 = \{y\}$  (called a cut);*
- *replace telomeres  $v = \{x\}$  and  $w = \{y\}$  with a single adjacency  $v' = \{x, y\}$  (called a join).*

The parsimony principle asserts that the true phylogenetic history minimizes the number of mutations. Feijão and Meidanis [4] showed that the minimum number of SCJ operations needed to transform one genome into another is precisely the Hamming distance between their binary string

---

Email: miklos.istvan@renyi.mta.hu (István Miklós) smithhc5@math.sc.edu (Heather Smith).

representations. As we consider the possible ancestors which could label the internal nodes of the tree, we use the parsimony score to determine the likelihood of a labeling. The parsimony score of a labeling  $\varphi$  of the vertices of  $T$  with binary strings is precisely the sum of the Hamming distances between labelings on adjacent edges. The labelings with a minimum parsimony score are called “most parsimonious labelings” and are considered to be the most likely.

Mutations are rare events, so only one happens at a time. A time order sequence of these SCJ operations, transforming one genome into another with the fewest number of SCJ operations is an *SCJ scenario*. We may note that the number of SCJ scenarios between two genomes is at most the factorial of the Hamming distance between their binary strings. In practice, the number of SCJ scenarios may be less because the sequence of digraphs produced from the sequences of cuts and joins must be valid genomes. However, we restrict our attention to multisets of genomes which differ in adjacencies which are independent. In other words, we may assume that the number of possible SCJ scenarios along each edge of the ancestral tree is the factorial of the Hamming distance between the binary strings.

We further make the assumption that the time order of mutations on one edge does not give any information about the time order of mutations on a different edge.

In summary, we will be given a multiset of observed genomes and an ancestral tree  $T$  relating them. This can be viewed as a tree  $T$  with binary strings labeling the leaves. The first goal is to determine the most parsimonious labelings. For each most parsimonious labeling, we label each edge of the tree with an SCJ scenario which details the time order that mutations occurred. The result is a most parsimonious SCJ scenario.

## 2. DEFINITIONS

Now let us fix some notation and formalize a few of the definitions which were introduced above. Throughout the paper, we let  $[m] := \{1, 2, \dots, m\}$ . For two binary strings  $\eta$  and  $\eta'$  of the same length, we let  $H(\eta, \eta')$  be the Hamming distance between  $\eta$  and  $\eta'$ , which is the number of coordinates in which  $\eta$  and  $\eta'$  differ. Whenever we are given a multiset,  $S$ , of binary strings, we assume that no two strings in  $S$  differ in length. For an arbitrary binary string  $\eta$  of length  $n$  and  $z \in [n]$ , we will use the notation  $\eta[z]$  to denote the value of  $\eta$  in the  $z$  position. For a multiset  $A$ , we define  $\#[x, A]$  to be the multiplicity of the element  $x$  in  $A$ . (If  $x \notin A$  then  $\#[x, A] = 0$ .) To explicitly write out a multiset, we use subscripts in parentheses to indicate the multiplicity of an element. For example,

$$\{a_{(3)}, b_{(4)}, c_{(1)}\} := \{a, a, a, b, b, b, b, c\}.$$

For two multisets  $A$  and  $B$ , we define  $A \uplus B$  to be the multiset with

$$\#[x, A \uplus B] = \#[x, A] + \#[x, B].$$

**Definition 2.1** (Most Parsimonious Labeling). *Let  $T$  be a tree with  $s$  leaves. Let  $B = \{v_1, v_2, \dots, v_s\}$  be a multiset of binary strings, each of length  $t$ . Let  $\phi : L(T) \rightarrow B$  be a surjection which assigns a binary string to each leaf of  $T$ . A most parsimonious labeling of the vertices of  $T$  is a labeling  $\phi' : V(T) \rightarrow \{0, 1\}^t$  with*

- $\phi'(\ell) = \phi(\ell)$  for each  $\ell \in L(T)$  (i.e.  $\phi'$  extends  $\phi$ ), and
- $\sum_{uv \in E(T)} H(\phi'(u), \phi'(v))$  is minimum among the possible functions  $\phi'$ .

**Definition 2.2** (SCJ scenarios). *Let  $\eta$  and  $\eta'$  be two binary strings of equal length with  $k := H(\eta, \eta')$ . Say  $\eta$  and  $\eta'$  differ on the coordinates  $C = \{b_{i_1}, b_{i_2}, \dots, b_{i_k}\}$ . An SCJ scenario for the pair  $(\eta, \eta')$  is a permutation of  $C$ .*

*Let  $T$  be a tree with most parsimonious labeling  $\phi'$ . For  $uv \in E(T)$ , we say that an SCJ scenario for  $(\phi'(u), \phi'(v))$  is an SCJ scenario for the edge  $uv$ .*

**Definition 2.3.** *Let  $B$  be a multiset of genomes on the same collection of genes. Consider the set of adjacencies which appear in at least one genome of  $B$ , but not in every genome of  $B$ . If*

no two of these adjacencies shares a common extremity, then we say that the adjacencies of  $B$  are independent.

Note that if  $\eta$  and  $\eta'$  have independent adjacencies, then the number of SCJ scenarios for  $(\eta, \eta')$  is precisely the symmetric difference between their adjacencies, which is the Hamming distance between their binary string representations.

**Definition 2.4** (Most parsimonious SCJ scenario). *Let  $T$  be a tree and  $\phi$  be a function labeling the leaves of  $T$  with binary strings. A most parsimonious SCJ scenario for  $T$  and  $\phi$  consists of a most parsimonious labeling  $\phi'$  and an SCJ scenario for each edge of  $T$  under vertex labeling  $\phi'$ .*

**Definition 2.5.** *Given a tree  $T$  and leaf labeling  $\phi$ , let  $\phi'$  be a most parsimonious labeling. We say that the number of SCJ scenarios admitted by  $\phi'$  is precisely the number of ways to assign an SCJ scenario to each edge of  $T$  for vertex labeling  $\phi'$ .*

When the tree  $T$  is a star with leaf labeling  $\phi$ , each most parsimonious labeling  $\phi'$  of the vertices is characterized by the binary string  $\mu$  it assigns to the center vertex of the star. We call  $\mu$  a median. Formally, we define a median as follows:

**Definition 2.6** (Median). *Let  $m$  be an arbitrary integer and  $S = \{\nu_i\}_{i=1}^m$  an arbitrary multiset of binary strings. A binary string  $\mu$  which minimizes  $\sum_{i \in [m]} H(\nu_i, \mu)$  is called a median for  $S$ .*

**Definition 2.7.** *For an arbitrary  $n, t \in \mathbb{Z}^+$ , let  $S$  be an arbitrary multiset of binary strings defined on the coordinates*

$$(x_1, y_1, x_2, y_2, \dots, x_n, y_n, e_1, e_2, \dots, e_t).$$

*We use  $\mathcal{M}(S)$  to denote the set of all medians for  $S$ . We use  $\mathcal{M}'(S)$  to denote the subset of  $\mathcal{M}(S)$  containing only those medians  $\mu$  with  $\mu[x_i] \neq \mu[y_i]$  for all  $i \in [n]$ .*

### 3. COMPUTATIONAL COMPLEXITY

While P and NP are complexity classes for decision problems, the following two classes are for counting problems.

**Definition 3.1** (First defined by Valiant [9]).  *$\#P$  is the class of all counting problems whose corresponding decision problem is in NP. A counting problem is  $\#P$ -hard if every problem in  $\#P$  can be reduced to it in polynomial time (see Definition 3.2). A counting problem is  $\#P$ -complete if it is in  $\#P$  and is  $\#P$ -hard.*

**Definition 3.2** (First defined by Cook [2]). *A polynomial time reduction from one decision (counting) problem  $A$  to another decision (counting) problem  $B$  is an algorithm which, for an arbitrary instance of  $A$ ,*

- *runs in time polynomial in the inputs of the instance of  $A$ ,*
- *creates an instance of  $B$ ,*
- *the answer to the instance of  $A$  can be computed in polynomial time given the instance of  $B$ .*

Note that the class of  $\#P$ -complete problems is closed under polynomial time reductions. Therefore, we will use polynomial time reductions to prove that a problem is in one of the above complexity classes. In order to do it, we also need some known computational complexity results. But first, we need a some more terminology.

A *3CNF* is a Boolean formula  $\Gamma$  which is the conjunction of clauses and each clause is the disjunction of 3 literals. A 3CNF,  $\Gamma$ , with  $n$  variables  $\{v_1, v_2, \dots, v_n\}$  and  $k$  clauses takes the form  $\Gamma = c_1 \wedge c_2 \wedge \dots \wedge c_k$  where each  $c_i$  is a clause of three literals joined by  $\vee$  and the literals are from  $\{v_i\}_{i=1}^n \cup \{\bar{v}_i\}_{i=1}^n$ . We may assume that, for each  $i \in [n]$ ,  $v_i$  or  $\bar{v}_i$  appears in some clause of  $\Gamma$ . Each  $v_i$  is a *positive literal* while each  $\bar{v}_i$  is a *negative literal*. The negative literal  $\bar{v}_i$  is the negation of  $v_i$ . We identify  $\bar{\bar{v}}_i$  with the literal  $v_i$ . We refer to  $\{v_i\}_{i=1}^n$  as the *variables* of  $\Gamma$  and always assume that the set of variables comes with some well ordering.

A *truth assignment* for  $\Gamma$  is a function  $f : \{v_i\}_{i=1}^n \rightarrow \{T, F\}$  which assigns a value of true or false to each variable. If a truth assignment makes  $\Gamma$  true, we say it satisfies  $\Gamma$ . Otherwise, a truth assignment does not satisfy  $\Gamma$  in which case there is at least one clause which is not satisfied.

**Definition 3.3** (3SAT). *Given an arbitrary Boolean formula  $\Gamma$  in 3CNF with  $n$  variables and  $k$  clauses, decide if there is a truth assignment for  $\Gamma$  which satisfies  $\Gamma$ .*

**Definition 3.4** (#3SAT). *Given an arbitrary Boolean formula  $\Gamma$  in 3CNF with  $n$  variables and  $k$  clauses, count the number of truth assignments which satisfy  $\Gamma$ .*

**Theorem 3.5** (Cook [2]). *3SAT  $\in$  NP-complete.*

**Theorem 3.6** (Cook [2]). *#3SAT  $\in$  #P-complete.*

Define *D3CNF* to be the subset of 3CNF containing only those  $\Gamma = \bigwedge_{i \in [k]} c_i$  such that for each  $i \in [k]$ ,

- $c_i$  contains three distinct literals, and
- $c_i$  does not contain both  $v_j$  and  $\overline{v_j}$  for any  $j \in [n]$ .

This defines the following two problems.

**Definition 3.7** (D3SAT). *For an arbitrary  $\Gamma$  in D3CNF with  $n$  variables and  $k$  clauses, decide if there a truth assignment which satisfies  $\Gamma$ .*

**Definition 3.8** (#D3SAT). *For an arbitrary  $\Gamma$  in D3CNF with  $n$  variables and  $k$  clauses, count the number of truth assignments which satisfy  $\Gamma$ .*

The following two results are proven through reductions from #3SAT and 3SAT.

**Lemma 3.9.** *#D3SAT  $\in$  #P-complete.*

*Proof.* This is a reduction from #3SAT. Let  $\Gamma$  be a 3CNF with  $n$  variables and  $k$  clauses. Let  $v_\alpha, v_\beta, v_\gamma$  be literals in  $\Gamma$  with  $\alpha < \beta < \gamma$ . Observe that each of the following pairs have the same satisfying truth assignments.

$$\begin{aligned} & (v_\alpha \vee v_\beta \vee v_\beta) \text{ and } (v_\alpha \vee v_\beta \vee v_\gamma) \wedge (v_\alpha \vee v_\beta \vee \overline{v_\gamma}). \\ & (v_\alpha \vee v_\alpha \vee v_\alpha) \text{ and } (v_\alpha \vee v_\beta \vee v_\gamma) \wedge (v_\alpha \vee \overline{v_\beta} \vee v_\gamma) \wedge (v_\alpha \vee v_\beta \vee \overline{v_\gamma}) \wedge (v_\alpha \vee \overline{v_\beta} \vee \overline{v_\gamma}). \end{aligned}$$

Further, a clause of the form  $(v_\alpha \vee \overline{v_\alpha} \vee v_\beta)$  is always true, so it can be removed.

Making these replacements in  $\Gamma$  will result in a D3CNF  $\Gamma'$  with  $n'$  variables ( $n' \leq n$ ) and at most  $4k$  clauses that have distinct literals. Because some clauses like  $(v_\alpha \vee \overline{v_\alpha} \vee v_\beta)$  are in  $\Gamma$  but not in  $\Gamma'$ , it is possible that  $n' < n$ .

Given a satisfying truth assignment for  $\Gamma'$ , we may extend it to a satisfying truth assignment for  $\Gamma$  in  $2^{n-n'}$  ways. This is because the variables in  $\Gamma$  which are not in  $\Gamma'$  do not affect the ability of a truth assignment to satisfy  $\Gamma$ . On the other hand, each satisfying truth assignment for  $\Gamma$ , restricted to the variables of  $\Gamma'$ , will be a satisfying truth assignment for  $\Gamma'$ . ■

**Lemma 3.10.** *D3SAT  $\in$  NP-complete.*

*Proof.* As described in the last proof, for any 3CNF  $\Gamma$ , there is a D3CNF  $\Gamma'$  which is computable in polynomial time such that  $\Gamma'$  has at least one satisfying truth assignment exactly when  $\Gamma$  has at least one satisfying truth assignment. ■

Next, we return our attention to the phylogenetic histories which were introduced in Section 1. The complexity results of this paper address subquestions and analogues of the following problems.

The small parsimony SCJ problem is stated as follows:

**Definition 3.11** (SPSCJ). *For  $z \in \mathbb{Z}^{\geq 0}$ , given an tree  $T$  and a labeling  $\varphi$  of its leaves with binary strings of length  $\ell$ , the SPSCJ problem asks if there at least  $z$  most parsimonious SCJ scenario.*

**Definition 3.12** (#SPSCJ). *Given a tree  $T$  and a labeling  $\varphi$  of the leaves of  $T$  with binary strings of equal length, #SPSCJ asks for the exact number of most parsimonious SCJ scenarios.*

**Lemma 3.13.** *SPSCJ  $\in$  NP and #SPSCJ  $\in$  #P.*

*Proof.* The proof for a positive answer to SPSCJ would include a list of  $z$  most parsimonious SCJ scenarios. We can verify that the vertex labeling is indeed a most parsimonious labeling in time polynomial in  $\ell$  and the size of  $T$ . Further, we only need  $O(\ell)$  time to verify that an edge label is indeed a scenario for the most parsimonious labeling. Thus SPSCJ  $\in$  NP. By the definition of #P, this implies #SPSCJ is in #P.  $\blacksquare$

When the tree  $T$  is a star and the genomes labeling the leaves have independent adjacencies (Definition 2.3), we can state the following special case of #SPSCJ:

**Definition 3.14** (#StarSPSCJ). *Given an arbitrary  $m \in \mathbb{Z}^+$ , let  $S = \{\nu_i\}_{i=1}^m$  be an arbitrary multiset of binary strings. Determine the value of*

$$\sum_{\mu \in \mathcal{M}(S)} \prod_{i \in [m]} H(\nu_i, \mu)!$$

When the tree  $T$  is a binary tree and the genomes labeling the leaves have independent adjacencies (Definition 2.3), we can state the following special case of #SPSCJ:

**Definition 3.15** (#BinSPSCJ). *Given arbitrary integer  $m \geq 2$ , let  $T$  be a binary tree with  $m$  leaves. Let  $S = \{\nu_i\}_{i=1}^m$  be an arbitrary multiset of binary strings with surjective function  $\varphi : L(T) \rightarrow S$ . Define  $F$  to be the set of most parsimonious labelings  $\varphi'$  which extend  $\varphi$  to  $V(T)$ . Determine the value of*

$$\sum_{\varphi' \in F} \prod_{uv \in E(T)} H(\varphi'(u), \varphi'(v))!$$

The results in the next two sections examine #SPSCJ and some analogues for classes of trees such as binary trees and star trees. Most of the complexity results are reductions from #D3SAT. In other words, given a D3CNF  $\Gamma$  with  $n$  variables and  $k$  clauses, we create a multiset of  $m$  binary strings of length  $2n + t$  (where  $t$  and  $m$  are polynomials of  $n$  and  $k$ ) to label the leaves of the tree. These strings are chosen so that the number of most parsimonious SCJ scenarios is related to the number of satisfying truth assignments for  $\Gamma$ .

#### 4. SET-UP FOR RESULTS ON STAR TREES

The discussion in this section and the next is specific to #StarSPSCJ. The main result of Section 5 is Theorem 5.4 which states #StarSPSCJ is #P-complete. In this section, we will give a general approach and develop some tools needed for the proof.

The proof will define a polynomial reduction from #D3SAT (Definition 3.8). Fix an arbitrary D3CNF,  $\Gamma$ , with  $n$  variables and  $k$  clauses. We will define a multiset of strings  $\mathcal{D}(\Gamma)$  to label the leaves of a star. This multiset of strings will encode  $\Gamma$ . Once the leaves of the star tree are labeled with our binary strings, we have an instance of #StarSPSCJ. For this instance, #StarSPSCJ asks us to count the number of most parsimonious labelings. To do this, fix a most parsimonious labeling and count the number of SCJ scenarios it admits. Then sum this quantity over all most parsimonious labelings.

For the star, each most parsimonious labeling is identified by the median (Definition 2.6) which it assigns to the center of the star. As in Definition 2.6, we let  $\mathcal{M}(\mathcal{D}(\Gamma))$  be the set of all medians for the star tree with leaf labels  $\mathcal{D}(\Gamma)$ .

Our task is to define a multiset of binary strings  $\mathcal{D}(\Gamma)$  which will represent a multiset of genomes with independent adjacencies. The multiset will be chosen so that  $\mathcal{M}(\mathcal{D}(\Gamma))$  will have a set of desired

characteristics. First, each of our strings in  $\mathcal{D}(\Gamma)$  and the medians  $\mathcal{M}(\mathcal{D}(\Gamma))$  will have length  $2n + t$  with coordinates

$$(x_1, y_1, x_2, y_2, \dots, x_n, y_n, e_1, e_2, \dots, e_t)$$

where  $n$  is the number of variables in  $\Gamma$  and the quantity  $t$  is a polynomial of  $n$  and  $k$  which will be defined later. Second,  $\mathcal{D}(\Gamma)$  will be defined so that the medians,  $\mathcal{M}(\mathcal{D}(\Gamma))$ , will be the set of all binary strings  $\mu$  of length  $2n + t$  that have  $\mu[e_i] = 0$  for each  $i \in [t]$ . In other words,  $\mathcal{M}(\mathcal{D}(\Gamma))$  should be equal to  $\{0, 1\}^{2n} \times \{0\}^t$ . Recall  $\mathcal{M}'(\mathcal{D}(\Gamma))$ , from Definition 2.7, is the subset of  $\mathcal{M}(\mathcal{D}(\Gamma))$  with the additional property that  $\mu[x_i] \neq \mu[y_i]$  for all  $i \in [n]$ . Once we have established that  $\mathcal{M}(\mathcal{D}(\Gamma)) = \{0, 1\}^{2n} \times \{0\}^t$ , we can conclude  $\mathcal{M}'(\mathcal{D}(\Gamma)) = \{01, 10\}^n \times \{0\}^t$ . This allows for a connection with truth assignments for  $\Gamma$ .

**Definition 4.1.** *Let  $n \in \mathbb{Z}^+$ . For arbitrary  $\Gamma$  in D3CNF with  $n$  variables, let  $S$  be a multiset of binary strings on the coordinates  $(x_1, y_1, \dots, x_n, y_n, e_1, \dots, e_t)$ . There is an injective function  $f$  which assigns to each median  $\mu \in \mathcal{M}'(S)$  a truth assignment for  $\Gamma$ . In particular,  $f(\mu)$  will assign a value of true to the  $i^{\text{th}}$  variable of  $\Gamma$  if  $\mu[x_i] = 1$  and false if  $\mu[x_i] = 0$ .*

**Remark 4.2.** *If multiset  $S$  is chosen so that  $\mathcal{M}'(S) = \{01, 10\}^n \times \{0\}^t$ , then Definition 4.1 provides a bijection between  $\mathcal{M}'(S)$  and the truth assignments for  $\Gamma$ .*

**Definition 4.3.** *Let  $n \in \mathbb{Z}^+$ . Given an arbitrary D3CNF,  $\Gamma$ , with  $n$  variables, let  $S$  be an arbitrary multiset of binary strings on the coordinates  $(x_1, y_1, \dots, x_n, y_n, e_1, \dots, e_t)$  for some  $t \in \mathbb{Z}^+$ . Define  $\mathcal{M}'_\Gamma(S)$  to a subset of  $\mathcal{M}'(S)$ , containing only those medians which, through the bijection in Definition 4.1, correspond to truth assignments which satisfy  $\Gamma$ . Since a single clause  $c$  in  $\Gamma$  is also a D3CNF, this defines  $\mathcal{M}'_c(S)$  as well.*

For #StarSPSCJ, we would like to calculate

$$\sum_{\mu \in \mathcal{M}(\mathcal{D}(\Gamma))} \prod_{i \in [m]} H(\mu, \nu_i)!$$

To do this, we first calculate  $\prod_{i \in [m]} H(\mu, \nu_i)!$  for each median  $\mu \in \mathcal{M}(\mathcal{D}(\Gamma))$  which is the number of scenarios admitted by median  $\mu$  (Definition 2.5). The multiset  $\mathcal{D}(\Gamma)$  will be constructed so that there is a constant  $K$  with each string in  $\mathcal{M}'_\Gamma(\mathcal{D}(\Gamma))$  admitting  $K$  scenarios and each string in  $\mathcal{M}(\mathcal{D}(\Gamma)) \setminus \mathcal{M}'_\Gamma(\mathcal{D}(\Gamma))$  admitting some number of scenarios which is not equal to  $K$ . Later we will see that the difficult part is distinguishing the number of scenarios admitted by each median in  $\mathcal{M}'_\Gamma(\mathcal{D}(\Gamma))$  from the number of scenarios admitted by each median in  $\mathcal{M}'(\mathcal{D}(\Gamma)) \setminus \mathcal{M}'_\Gamma(\mathcal{D}(\Gamma))$ . In Section 4.1, we define the strings  $\mathcal{D}(\Gamma)$  which are used in the proofs to distinguish  $\mathcal{M}'_\Gamma(\mathcal{D}(\Gamma))$  from  $\mathcal{M}'(\mathcal{D}(\Gamma)) \setminus \mathcal{M}'_\Gamma(\mathcal{D}(\Gamma))$ .

**4.1. Encoding clauses in binary strings.** A truth assignment satisfies  $\Gamma$  if and only if it satisfies every clause in  $\Gamma$ . Hence, we will encode each clause  $c_i$  of  $\Gamma$  in a set of 50 strings

$$\mathcal{C}_i := \{\nu_1^i, \nu_2^i, \dots, \nu_{50}^i\}$$

which will be defined through Table 1. These 50 strings, which will be a subset of  $\mathcal{D}(\Gamma)$ , are designed to distinguish those medians in  $\mathcal{M}'_{c_i}(\mathcal{C}_i)$  from those in  $\mathcal{M}'(\mathcal{C}_i) \setminus \mathcal{M}'_{c_i}(\mathcal{C}_i)$ . Confirmation of this will come in Section 4.3. Because every truth assignment which does not satisfy  $\Gamma$  does not satisfy some clause in  $\Gamma$ , we will see that  $\biguplus_{i \in [k]} \mathcal{C}_i$  distinguishes between  $\mathcal{M}'_\Gamma(\biguplus_{i \in [k]} \mathcal{C}_i)$  and  $\mathcal{M}'(\biguplus_{i \in [k]} \mathcal{C}_i) \setminus \mathcal{M}'_\Gamma(\biguplus_{i \in [k]} \mathcal{C}_i)$ .

The following definition gives a guide for defining a collection of binary strings.

**Definition 4.4** (Defining strings). *For arbitrary  $m, n \in \mathbb{Z}^+$  and  $t \in \mathbb{Z}^{\geq 0}$ , to define a multiset of binary strings  $\{\eta_1, \eta_2, \dots, \eta_m\}$  on coordinates  $(x_1, y_1, \dots, x_n, y_n, e_1, \dots, e_t)$ , it suffices to*

- define  $\eta_j[x_\ell]$  and  $\eta_j[y_\ell]$  for each  $j \in [m]$  and  $\ell \in [n]$ , and
- define a function  $e : [m] \rightarrow \mathbb{Z}^{\geq 0}$ .

We say  $\eta_j$  has  $e(j)$  additional ones. In order to infer the values  $\eta_j[e_\ell]$  for each  $j \in [m]$  and  $\ell \in [t]$ , follow this procedure:

Partition  $[t]$  into subsets  $E, E_1, E_2, \dots, E_m$  so that the size of  $E_j$  is precisely  $e(j)$ , and  $E = [t] \setminus \bigcup_{j \in [m]} E_j$ . For each  $j \in [m]$  and each  $\ell \in E_j$ , define  $\eta_j[e_\ell] = 1$ , and for  $j' \neq j$ , let  $\eta_{j'}[e_\ell] = 0$ .

**Remark 4.5.** Let  $m \in \mathbb{Z}^+$ . For an arbitrary multiset  $\{\eta_j\}_{j=1}^m$  of binary strings built using Definition 4.4, for each  $\ell \in [t]$ , there is a unique  $j \in [m]$  such that  $\eta_j[e_\ell] = 1$ . Consequently, each  $\mu \in \mathcal{M}(\{\eta_j\}_{j=1}^m)$  will have  $\mu[e_\ell] = 0$  for all  $\ell \in [t]$  because  $\mu$  must minimize  $\sum_{j \in [m]} H(\eta_j, \mu)$ .

**Definition 4.6.** Let  $n \in \mathbb{Z}^+$  and  $t \in \mathbb{Z}^{\geq 0}$  be arbitrary. Two binary strings  $\eta$  and  $\bar{\eta}$  with coordinates  $(x_1, y_1, \dots, x_n, y_n, e_1, \dots, e_t)$ , are said to be complementary on the first  $2n$  coordinates if  $\eta[x_i] = 1 - \bar{\eta}[x_i]$  and  $\eta[y_i] = 1 - \bar{\eta}[y_i]$  for each  $i \in [n]$ .

The following fact will be useful.

**Fact 4.7.** Let  $\eta$  and  $\bar{\eta}$  be binary strings on coordinates

$$(x_1, y_1, \dots, x_n, y_n, e_1, \dots, e_t).$$

Set  $e(\eta) := \sum_{i \in [t]} \eta[e_i]$ , the number of additional ones in  $\eta$ . Define  $e(\bar{\eta})$  similarly. If  $\eta$  and  $\bar{\eta}$  are complementary on the first  $2n$  coordinates, then for any  $\mu \in \{0, 1\}^{2n} \times \{0\}^t$ , we have

$$H(\mu, \eta) + H(\mu, \bar{\eta}) = 2n + e(\eta) + e(\bar{\eta}).$$

*Proof.* For each  $i \in [n]$ , either  $\mu[x_i] = \eta[x_i]$  or  $\mu[x_i] = \bar{\eta}[x_i]$ , but not both. This is also true for each  $y_i$ . This accounts for the  $2n$  in the sum. Because  $\mu[e_i] = 0$  for all  $i \in [t]$ , each  $i$  with  $\eta[e_i] = 1$  will contribute one to the sum. Also each  $i$  with  $\bar{\eta}[e_i]$  will contribute one to the sum. This completes the proof.  $\blacksquare$

**Definition 4.8.** Given an arbitrary multiset of binary strings  $S$ , we say that a coordinate  $s$  is ambiguous if there are exactly  $\frac{1}{2}|S|$  binary strings  $\eta \in S$ , counted with multiplicity, such that  $\eta[s] = 0$ .

Consequently, if you change the value of a median at an ambiguous coordinate, you obtain another median.

**Fact 4.9.** Let  $S$  be a multiset of binary strings which are defined on the coordinates

$$(x_1, y_1, \dots, x_n, y_n, e_1, \dots, e_t).$$

If  $S$  can be partitioned into pairs of vertices where the two strings in a pair are complementary on the first  $2n$  coordinates, then each  $x_i$  and each  $y_i$  is an ambiguous coordinate.

Fix an arbitrary D3CNF,  $\Gamma$ , with  $n$  variables and  $k$  clauses. Fix a clause  $c_i$  in  $\Gamma$ . For this clause, we are now ready to define a set of 50 strings

$$\mathcal{C}_i = \{\nu_1^i, \nu_2^i, \dots, \nu_{50}^i\}.$$

First assume that  $c_i = v_\alpha \vee v_\beta \vee v_\gamma$ , a disjunction of three positive literals. Because  $\Gamma$  is in D3CNF, we may assume  $\alpha < \beta < \gamma$ .

For each  $j \in [50]$ , we give the following three pieces of information for  $\nu_j^i$ .

- (a) The values for  $\nu_j^i[x_\alpha], \nu_j^i[y_\alpha], \nu_j^i[x_\beta], \nu_j^i[y_\beta], \nu_j^i[x_\gamma], \nu_j^i[y_\gamma]$  will be explicitly defined.
- (b) A constant  $\kappa_{ij} \in \{0, 1\}$  will be given so that  $\nu_j^i[x_\ell] = \nu_j^i[y_{\ell'}] = \kappa_{ij}$  for all  $\ell, \ell' \in [n] \setminus \{\alpha, \beta, \gamma\}$ .
- (c) The string will be assigned some number of additional ones.

By Definition 4.4, this is sufficient to explicitly define  $\nu_j^i$ .

For each string in  $\mathcal{C}_i$ , these three defining pieces of information are found in the corresponding row headings of Table 1. There are 50 rows in the table, one for each  $\nu_j^i$ . The remainder of the table will be explained in Subsection 4.2.

For each  $j \in [50]$ , row  $j$  of Table 1 supplies the three ingredients needed to define  $\nu_j^i$ . By matching the 6-bit string in Column A of row  $j$  with

$$(\nu_j^i[x_\alpha], \nu_j^i[y_\alpha], \nu_j^i[x_\beta], \nu_j^i[y_\beta], \nu_j^i[x_\gamma], \nu_j^i[y_\gamma])$$

we obtain the 6 values for (a). The constant  $\kappa_{ij}$  for (b) is found in Column (B) of row  $j$ . For (c), the number of additional ones in  $\nu_j^i$  is found in Column C of row  $j$ .

With a slight modification in the reading of Column A, the 50 rows of Table 1 will also supply the 50 strings for a clause with negative literals. Fix an arbitrary clause  $c_i$  in  $\Gamma$  which now may have negative literals. For each  $j \in [50]$ , the definition of string  $\nu_j^i$  will again be based on Columns A, B, C of row  $j$  in Table 1. The same information will be gleaned from Columns B and C as in the case when  $c_i$  had no negative literals. The only difference is with Column A which will be explained next.

Let  $S_i$  denote the *support set* of clause  $c_i$ . If  $c_i$  contains variables  $v_\alpha, v_\beta, v_\gamma$  ( $\alpha < \beta < \gamma$ ), then  $S_i := \{x_\alpha, y_\alpha, x_\beta, y_\beta, x_\gamma, y_\gamma\}$ . Clause  $c_i$  must be one of the 8 clauses listed Column A of Table 2. For  $j \in [50]$ ,  $\nu_j^i$  is defined on the coordinates  $S_i$  by matching the entry in the right column of Table 2 for  $c_i$  with the 6-bit string in Column A of the  $j^{\text{th}}$  row of Table 1.

**Example 4.10.** Here is an example of how to obtain a string from Table 1 when  $c_i = v_\alpha \vee \overline{v_\beta} \vee \overline{v_\gamma}$ . The last row of Table 1 says that the string  $\nu_{50}^i$  must have

$$(\nu_{50}^i[x_\alpha], \nu_{50}^i[y_\alpha], \nu_{50}^i[x_\beta], \nu_{50}^i[y_\beta], \nu_{50}^i[x_\gamma], \nu_{50}^i[y_\gamma]) = (101010).$$

Therefore,  $\nu_{50}^i[x_\alpha] = 1$ ,  $\nu_{50}^i[y_\alpha] = 0$ ,  $\nu_{50}^i[x_\beta] = 0$ ,  $\nu_{50}^i[y_\beta] = 1$ ,  $\nu_{50}^i[x_\gamma] = 0$ , and  $\nu_{50}^i[y_\gamma] = 1$ . Further, the Column B implies  $\nu_{50}^i(x_\ell) = \nu_{50}^i(y_\ell) = 1$  for all  $\ell \in [n] \setminus \{\alpha, \beta, \gamma\}$  and, from Column C,  $\nu_{50}^i$  will have 2 additional ones.

Now that we have defined  $\mathcal{C}_i$  for any clause  $c_i$ , let us analyze  $\mathcal{M}(\mathcal{C}_i)$ . By Fact 4.5 for any  $\mu \in \mathcal{M}(\mathcal{C}_i)$ ,  $\mu[e_\ell] = 0$  for all  $\ell \in [t]$ .

In Column B of Table 1, it is evident that for any  $\ell \in [n] \setminus \{\alpha, \beta, \gamma\}$ , the number of strings  $\nu_j^i$  with  $\nu_j^i[x_\ell] = 0$  is  $25 = \frac{1}{2}|\mathcal{C}_i|$ . Therefore, by Definition 4.8, the coordinates  $x_\ell$  and  $y_\ell$  are ambiguous. With a careful inspection of the strings Column A of Table 1, we see that coordinates  $x'_\ell$  and  $y'_\ell$  are also ambiguous for each  $\ell' \in \{\alpha, \beta, \gamma\}$ . Therefore we have proven the following fact, which was one of our goals:

**Fact 4.11.** For an arbitrary  $\Gamma$  with clause  $c_i$ ,

$$\mathcal{M}(\mathcal{C}_i) = \{0, 1\}^{2n} \times \{0\}^t.$$

**Remark 4.12.** By visual inspection of Table 1, the binary strings  $\mathcal{C}_i$  can be partitioned into pairs where the two strings in a pair are complementary on the first  $2n$  coordinates.

**4.2. Hamming distances.** Fix a clause  $c_i$  in  $\Gamma$  which will be used throughout this subsection. Suppose  $c_i$  has variables  $v_\alpha, v_\beta$ , and  $v_\gamma$ . By Fact 4.11,  $\mathcal{M}(\mathcal{C}_i) = \{0, 1\}^{2n} \times \{0\}^t$ . Therefore,  $\mathcal{M}'(\mathcal{C}_i)$ , from Definition 2.6, must be equal to  $\{01, 10\}^n \times \{0\}^t$ . For this subsection, define

$$\mathcal{M} := \mathcal{M}(\mathcal{C}_i), \quad \mathcal{M}' := \mathcal{M}'(\mathcal{C}_i).$$

Define an equivalence relation  $\sim_i$  on  $\mathcal{M}'$  such that two medians are equivalent if they agree on the coordinates in the support set  $S_i$  of  $c_i$ . The result will be 8 equivalence classes because  $\mu[x_\ell] \neq \mu[y_\ell]$  for each  $\ell \in \{\alpha, \beta, \gamma\}$  for each  $\mu \in \mathcal{M}'$ .

Here we define a one-to-one correspondence between the equivalence class of  $\mathcal{M}'$  under  $\sim_i$  and the 6-bit strings heading Columns M1 through M8 in Table 1.



TABLE 1. The 50 strings in  $\mathcal{C}_i$  for a single clause  $c_i$  along with their Hamming distance from medians in  $\mathcal{M}'$ .

		A	B	C	M1	M2	M3	M4	M5	M6	M7	M8
	Row #	Values of $\nu_j^i$ on its support set	$\nu_j^i[x_\ell], \nu_j^i[y_\ell]$ ( $\nu_\ell \notin c_i$ )	Add'l Ones	101010	101001	100110	011010	100101	011001	010110	010101
$\mathcal{M}_1^{(+3)}$	1	010000	0	+3	$n+4$	$n+4$	$n+4$	$n+2$	$n+4$	$n+2$	$n+2$	$n+2$
	2	000100	0	+3	$n+4$	$n+4$	$n+2$	$n+4$	$n+2$	$n+4$	$n+2$	$n+2$
	3	000001	0	+3	$n+4$	$n+2$	$n+4$	$n+4$	$n+2$	$n+2$	$n+4$	$n+2$
$\mathcal{M}_1^{(+0)}$	4	101111	1	+0	$n-1$	$n-1$	$n-1$	$n+1$	$n-1$	$n+1$	$n+1$	$n+1$
	5	111011	1	+0	$n-1$	$n-1$	$n+1$	$n-1$	$n+1$	$n-1$	$n+1$	$n+1$
	6	111110	1	+0	$n-1$	$n+1$	$n-1$	$n-1$	$n+1$	$n+1$	$n-1$	$n+1$
$\mathcal{I}_2^{(+2)} \setminus \mathcal{N}_2^{(+2)}$	7	101000	0	+2	$n$	$n$	$n+2$	$n+2$	$n+2$	$n+2$	$n+4$	$n+4$
	8	100010	0	+2	$n$	$n+2$	$n$	$n+2$	$n+2$	$n+4$	$n+2$	$n+4$
	9	001010	0	+2	$n$	$n+2$	$n+2$	$n$	$n+4$	$n+2$	$n+2$	$n+4$
	10	101000	0	+2	$n$	$n$	$n+2$	$n+2$	$n+2$	$n+2$	$n+4$	$n+4$
	11	100001	0	+2	$n+2$	$n$	$n+2$	$n+4$	$n$	$n+2$	$n+4$	$n+2$
	12	001001	0	+2	$n+2$	$n$	$n+4$	$n+2$	$n+2$	$n$	$n+4$	$n+2$
	13	100100	0	+2	$n+2$	$n+2$	$n$	$n+4$	$n$	$n+4$	$n+2$	$n+2$
	14	100010	0	+2	$n$	$n+2$	$n$	$n+2$	$n+2$	$n+4$	$n+2$	$n+4$
	15	000110	0	+2	5	$n+4$	$n$	$n+2$	$n+2$	$n+4$	$n$	$n+2$
	16	011000	0	+2	$n+2$	$n+2$	$n+4$	$n$	$n+4$	$n$	$n+2$	$n+2$
	17	010010	0	+2	$n+2$	$n+4$	$n+2$	$n$	$n+4$	$n+2$	$n$	$n+2$
	18	001010	0	+2	$n$	$n+2$	$n+2$	$n$	$n+4$	$n+2$	$n+2$	$n+4$
	19	100100	0	+2	$n+2$	$n+2$	$n$	$n+4$	$n$	$n+4$	$n+2$	$n+2$
	20	100001	0	+2	$n+2$	$n$	$n+2$	$n+4$	$n$	$n+2$	$n+4$	$n+2$
	21	000101	0	+2	$n+4$	$n+4$	$n+2$	$n+4$	$n$	$n+2$	$n+2$	$n$
	22	011000	0	+2	$n+2$	$n+2$	$n+4$	$n$	$n+4$	$n$	$n+2$	$n+2$
	23	010001	0	+2	$n+4$	$n+2$	$n+4$	$n+2$	$n+2$	$n$	$n+2$	$n$
	24	001001	0	+2	$n+2$	$n$	$n+4$	$n+2$	$n+2$	$n$	$n+4$	$n+2$
	25	010100	0	+2	$n+4$	$n+4$	$n+2$	$n+2$	$n+2$	$n+2$	$n+2$	$n$
	26	010010	0	+2	$n+2$	$n+4$	$n+2$	$n$	$n+4$	$n+2$	$n$	$n+2$
27	000110	0	+2	$n+2$	$n+4$	$n$	$n+2$	$n+2$	$n+4$	$n$	$n+2$	
$\mathcal{I}_2^{(+1)} \setminus \mathcal{N}_2^{(+1)}$	28	101011	1	+1	$n-1$	$n-1$	$n+1$	$n+1$	$n+1$	$n+1$	$n+3$	$n+3$
	29	101101	1	+1	$n+1$	$n-1$	$n+1$	$n+3$	$n-1$	$n+1$	$n+3$	$n+1$
	30	111001	1	+1	$n+1$	$n-1$	$n+3$	$n+1$	$n+1$	$n-1$	$n+3$	$n+1$
	31	100111	1	+1	$n+1$	$n+1$	$n-1$	$n+3$	$n-1$	$n+3$	$n+1$	$n+1$
	32	101110	1	+1	$n-1$	$n+1$	$n-1$	$n+1$	$n+1$	$n+3$	$n+1$	$n+3$
	33	110110	1	+1	$n+1$	$n+3$	$n-1$	$n+1$	$n+1$	$n+3$	$n-1$	$n+1$
	34	011011	1	+1	$n+1$	$n+1$	$n+3$	$n-1$	$n+3$	$n-1$	$n+1$	$n+1$
	35	011110	1	+1	$n+1$	$n+3$	$n+1$	$n-1$	$n+3$	$n+1$	$n-1$	$n+1$
	36	111010	1	+1	$n-1$	$n+1$	$n+1$	$n-1$	$n+3$	$n+1$	$n+1$	$n+3$
	37	100111	1	+1	$n+1$	$n+1$	$n-1$	$n+3$	$n-1$	$n+3$	$n+1$	$n+1$
	38	101101	1	+1	$n+1$	$n-1$	$n+1$	$n+3$	$n-1$	$n+1$	$n+3$	$n+1$
	39	110101	1	+1	$n+3$	$n+1$	$n+1$	$n+3$	$n-1$	$n+1$	$n+1$	$n-1$
$\mathcal{I}_2^{(+1)} \setminus \mathcal{N}_2^{(+1)}$	40	011011	1	+1	$n+1$	$n+1$	$n+3$	$n-1$	$n+3$	$n-1$	$n+1$	$n+1$
	41	011101	1	+1	$n+3$	$n+1$	$n+3$	$n+1$	$n+1$	$n-1$	$n+1$	$n-1$
	42	111001	1	+1	$n+1$	$n-1$	$n+3$	$n+1$	$n+1$	$n-1$	$n+3$	$n+1$
	43	010111	1	+1	$n+3$	$n+3$	$n+1$	$n+1$	$n+1$	$n+1$	$n-1$	$n-1$
	44	011110	1	+1	$n+1$	$n+3$	$n+1$	$n-1$	$n+3$	$n+1$	$n-1$	$n+1$
	45	110110	1	+1	$n+1$	$n+3$	$n-1$	$n+1$	$n+1$	$n+3$	$n-1$	$n+1$
	46	010111	1	+1	$n+3$	$n+3$	$n+1$	$n+1$	$n+1$	$n+1$	$n-1$	$n-1$
	47	011101	1	+1	$n+3$	$n+1$	$n+3$	$n+1$	$n+1$	$n-1$	$n+1$	$n-1$
48	110101	1	+1	$n+3$	$n+1$	$n+1$	$n+3$	$n-1$	$n+1$	$n+1$	$n-1$	
$\mathcal{M}_3^{(+1)}$	49	010101	0	+1	$n+3$	$n+2$	$n+2$	$n+4$	$n$	$n$	$n$	$n-2$
$\mathcal{M}_3^{(+2)}$	50	101010	1	+2	$n-1$	$n+1$	$n+1$	$n+1$	$n+3$	$n+3$	$n+3$	$n-3$

For a clause  $c_i$ , the left three columns define the 50 strings in  $\mathcal{C}_i$ . In row  $j$ , the 6-bit string gives the values of  $\nu_j^i$  on the support set  $S_i$  as described by Table 2. The second column gives the constant value to be assigned to all  $x_\ell$  and  $y_\ell$  which are not in  $S_i$ . The third column specifies the number of extra ones in  $\nu_j^i$ . The collection  $\{01, 10\}^3$  is listed along the top row. The entry in row  $j$  and column  $\ell$  is the number of additional ones in  $\nu_j^i$  added to the Hamming distance between the 6-bit string in row  $j$  and the 6-bit string at the top of column  $\ell$ .

**Definition 4.13.** Fix a clause  $c_i$  and an integer  $\ell \in [8]$ . Consider the 6-bit string  $\delta$  which heads column  $M\ell$ . In Table 2, locate the tuple in the right column corresponding to our fixed clause  $c_i$ . After replacing each  $\nu_j^i$  with  $\mu$  in the tuple, match this tuple with  $\delta$ . This gives six values that a median  $\mu \in \mathcal{M}'$  must have if it is in the equivalence class represented by the column heading  $\delta$ .

TABLE 2. A key for interpreting Column A of Table 1.

Clause	Key to interpret Column A of Table 1
$v_\alpha \vee v_\beta \vee v_\gamma$	$(\nu_j^\alpha[x_\alpha], \nu_j^\alpha[y_\alpha], \nu_j^\alpha[x_\beta], \nu_j^\alpha[y_\beta], \nu_j^\alpha[x_\gamma], \nu_j^\alpha[y_\gamma])$
$\overline{v_\alpha} \vee v_\beta \vee v_\gamma$	$(\nu_j^\alpha[y_\alpha], \nu_j^\alpha[x_\alpha], \nu_j^\alpha[x_\beta], \nu_j^\alpha[y_\beta], \nu_j^\alpha[x_\gamma], \nu_j^\alpha[y_\gamma])$
$v_\alpha \vee \overline{v_\beta} \vee v_\gamma$	$(\nu_j^\alpha[x_\alpha], \nu_j^\alpha[y_\alpha], \nu_j^\alpha[y_\beta], \nu_j^\alpha[x_\beta], \nu_j^\alpha[x_\gamma], \nu_j^\alpha[y_\gamma])$
$v_\alpha \vee v_\beta \vee \overline{v_\gamma}$	$(\nu_j^\alpha[x_\alpha], \nu_j^\alpha[y_\alpha], \nu_j^\alpha[x_\beta], \nu_j^\alpha[y_\beta], \nu_j^\alpha[y_\gamma], \nu_j^\alpha[x_\gamma])$
$\overline{v_\alpha} \vee \overline{v_\beta} \vee v_\gamma$	$(\nu_j^\alpha[y_\alpha], \nu_j^\alpha[x_\alpha], \nu_j^\alpha[y_\beta], \nu_j^\alpha[x_\beta], \nu_j^\alpha[x_\gamma], \nu_j^\alpha[y_\gamma])$
$\overline{v_\alpha} \vee v_\beta \vee \overline{v_\gamma}$	$(\nu_j^\alpha[y_\alpha], \nu_j^\alpha[x_\alpha], \nu_j^\alpha[x_\beta], \nu_j^\alpha[y_\beta], \nu_j^\alpha[y_\gamma], \nu_j^\alpha[x_\gamma])$
$v_\alpha \vee \overline{v_\beta} \vee \overline{v_\gamma}$	$(\nu_j^\alpha[x_\alpha], \nu_j^\alpha[y_\alpha], \nu_j^\alpha[y_\beta], \nu_j^\alpha[x_\beta], \nu_j^\alpha[y_\gamma], \nu_j^\alpha[x_\gamma])$
$\overline{v_\alpha} \vee \overline{v_\beta} \vee \overline{v_\gamma}$	$(\nu_j^\alpha[y_\alpha], \nu_j^\alpha[x_\alpha], \nu_j^\alpha[y_\beta], \nu_j^\alpha[x_\beta], \nu_j^\alpha[y_\gamma], \nu_j^\alpha[x_\gamma])$

For any clause in the left column, the corresponding entry in the right column above will be matched with the 6-bit string in Column A of row  $j$  of Table 1 to determine the value of  $\nu_j^i$  at each bit in the support set  $S_i$ .

In Definition 4.1, we defined a correspondence between  $\mathcal{M}'$  and truth assignments for  $\Gamma$ . In Definition 4.3, we introduced the notation  $\mathcal{M}'_\Gamma(\mathcal{C}_i)$  for the collection of medians in  $\mathcal{M}'$  which correspond to satisfying truth assignments for  $\Gamma$ . For each clause  $c_i$  in  $\Gamma$ , we can similarly define  $\mathcal{M}'_{c_i}(\mathcal{C}_i)$  to be the collection of medians in  $\mathcal{M}'$  which correspond to satisfying truth assignments for  $c_i$ . For the remainder of this subsection, set

$$\mathcal{M}'_\Gamma := \mathcal{M}'_\Gamma(\mathcal{C}_i), \quad \mathcal{M}'_{c_i} := \mathcal{M}'_{c_i}(\mathcal{C}_i).$$

The following claim uses the correspondence in Definition 4.13 to connect  $\mathcal{M}' \setminus \mathcal{M}'_{c_i}$  with a particular equivalence class.

**Claim 4.14.** *Let  $c_i$  be a clause in  $\Gamma$ . For any  $\mu \in \mathcal{M}'$ ,  $\mu$  is in the equivalence class represented by Column M8 of Table 1, if and only if  $\mu \in \mathcal{M}' \setminus \mathcal{M}'_{c_i}$ .*

*Proof.* Fix a clause  $c_i$  with variables  $v_\alpha, v_\beta, v_\gamma$ . This clause may have some negative literals. We focus our attention on  $v_\alpha$ . The arguments for  $v_\beta$  and  $v_\gamma$  are exactly the same.

There are two cases depending on whether  $v_\alpha$  appears as a positive literal or a negative literal in  $c_i$ .

In the case where  $v_\alpha$  appears in  $c_i$  as a positive literal, the truth assignment which makes  $c_i$  false assigns a value of false to  $v_\alpha$ . A corresponding median  $\mu \in \mathcal{M}'$  has  $\mu[x_\alpha] = 0$  and  $\mu[y_\alpha] = 1$ . Because  $v_\alpha$  appears as a positive literal in  $c_i$ , the entry in the second column of Table 2 has  $\mu[x_\alpha]$  followed by  $\mu[y_\alpha]$ . So, in this case, the 6-bit string which heads the column for medians in  $\mathcal{M}' \setminus \mathcal{M}'_{c_i}$  has 01 in the first two entries.

In the case where  $v_\alpha$  appears as a negative literal in  $c_i$ , the non-satisfying truth assignments for  $c_i$  must have  $v_\alpha$  true. The corresponding medians  $\mu \in \mathcal{M}'$  will have  $\mu[x_\alpha] = 1$  and  $\mu[y_\alpha] = 0$ . For the clauses with variable  $v_\alpha$  appearing as a negative literal in  $c_i$ , a quick glance at Table 2 reveals that  $\mu[y_\alpha]$  immediately precedes  $\mu[x_\alpha]$  in the 6-bit column headings in Table 1. As a result, the column representing medians in  $\mathcal{M}' \setminus \mathcal{M}'_{c_i}$  has 01 in the first two entries.

Repeating this argument for  $v_\beta$  and  $v_\gamma$ , we see that medians in  $\mathcal{M}' \setminus \mathcal{M}'_{c_i}$  are represented by the column with heading 010101.  $\blacksquare$

Now that we have defined the rows and columns of Table 1, we conclude this subsection by defining the entries within Table 1 for fixed clause  $c_i$ .

Let  $\mu \in \mathcal{M}'$  be an arbitrary median that falls into the equivalence class represented by Column  $M\ell$  for some  $\ell \in [8]$ . The entry  $a_{j\ell}$  in Row  $j$  and Column  $M\ell$  of Table 1 is  $H(\mu, \nu_j^i)$ . This value can be calculated as follows:

- First, take the Hamming distance between the 6-bit string in Column  $A$  of Row  $j$  and the 6-bit string in the header of Column  $M\ell$ . This is equal to the Hamming distance between the restrictions of  $\mu$  and  $\nu_j^i$  to the support set  $S_i$  for  $c_i$ .
- For any  $s \notin \{\alpha, \beta, \gamma\}$ ,  $\mu[x_s] \neq \mu[y_s]$  and  $\nu_j^i[x_s] = \nu_j^i[y_s]$ . Therefore the Hamming distance between  $(\mu[x_s], \mu[y_s])$  and  $(\nu_j^i[x_s], \nu_j^i[y_s])$  is 1 for each  $s \in [n] \setminus \{\alpha, \beta, \gamma\}$ .
- Finally, because  $\mu[e_s] = 0$  for all  $s \in [t]$ , the Hamming distance between the restrictions of  $\mu$  and  $\nu_j^i$  to the coordinates  $(e_1, e_2, \dots, e_t)$  is the number of additional ones in  $\nu_j^i$  which is found in Column  $C$  of Row  $j$ .

Adding these three values together gives the entry  $a_{j\ell}$ .

**4.3. Distinguishing satisfying truth assignments from those which are not.** Fix a clause  $c_i$  in arbitrary D3CNF  $\Gamma$ . For this subsection, we again set  $\mathcal{M}' := \mathcal{M}'(\mathcal{C}_i)$ ,  $\mathcal{M}'_\Gamma := \mathcal{M}'_\Gamma(\mathcal{C}_i)$ , and  $\mathcal{M}'_{c_i} := \mathcal{M}'_{c_i}(\mathcal{C}_i)$ . For each  $\mu \in \mathcal{M}' \setminus \mathcal{M}'_{c_i}$ ,  $\mu$  is in the equivalence class represented by 010101 by Claim 4.14. Then reading the entries in Column M8 of Table 1, we find

$$(1) \quad \{H(\mu, \nu_j^i) : j \in [50]\} = \{(n-2)_{(1)}, (n-1)_{(6)}, n_{(3)}, (n+1)_{(15)}, \\ (n+2)_{(15)}, (n+3)_{(3)}, (n+4)_{(6)}, (n+5)_{(1)}\}.$$

Otherwise, for each median  $\mu \in \mathcal{M}'_{c_i}$ ,  $\mu$  is in one of 7 equivalence classes represented in Columns M1 through M7. The entries in each of these columns reveal

$$(2) \quad \{H(\mu, \nu_j^i) : j \in [50]\} = \{(n-1)_{(7)}, n_{(6)}, (n+1)_{(12)}, (n+2)_{(12)}, (n+3)_{(6)}, (n+4)_{(7)}\}.$$

Thus, we can use  $\mathcal{C}_i$  to distinguish between the medians in  $\mathcal{M}'_{c_i}$  and the medians in  $\mathcal{M}' \setminus \mathcal{M}'_{c_i}$ . For example, for  $\mu \in \mathcal{M}' = \{01, 10\} \times \{0\}^t$ , if  $(n+5) \in \{H(\mu, \nu_j^i) : j \in [50]\}$ , then  $\mu \in \mathcal{M}' \setminus \mathcal{M}'_{c_i}$ .

Because  $\mathcal{M}'(\mathcal{C}_i) = \{01, 10\} \times \{0\}^t$  for all  $i \in [k]$ , we have  $\mathcal{M}'\left(\biguplus_{i \in [k]} \mathcal{C}_i\right) = \{01, 10\} \times \{0\}^t$ . As a result

$$\mathcal{M}'_{c_i} := \mathcal{M}'_{c_i}(\mathcal{C}_i) = \mathcal{M}'_{c_i}\left(\biguplus_{i \in [k]} \mathcal{C}_i\right) \\ \mathcal{M}'_\Gamma := \mathcal{M}'_\Gamma(\mathcal{C}_i) = \mathcal{M}'_\Gamma\left(\biguplus_{i \in [k]} \mathcal{C}_i\right).$$

Notice that

$$(3) \quad \mathcal{M}'_\Gamma = \bigcap_{i \in [k]} \mathcal{M}'_{c_i}$$

$$(4) \quad \mathcal{M}' \setminus \mathcal{M}'_\Gamma = \mathcal{M}' \setminus \bigcap_{i \in [k]} \mathcal{M}'_{c_i} = \bigcup_{i \in [k]} \mathcal{M}' \setminus \mathcal{M}'_{c_i}.$$

Therefore  $\biguplus_{i \in [k]} \mathcal{C}_i$  will be a tool to distinguish  $\mathcal{M}'_\Gamma$  from  $\mathcal{M}' \setminus \mathcal{M}'_\Gamma$ . For example, for any  $\mu \in \mathcal{M}' = \{01, 10\} \times \{0\}^t$ , if  $(n+5) \notin \{H(\mu, \nu_j^i) : j \in [50], i \in [k]\}$ , then  $\mu \in \mathcal{M}'_{c_i}$  for all  $i \in [k]$  which implies  $\mu \in \mathcal{M}'_\Gamma$ .

## 5. MOST PARSIMONIOUS SCJ SCENARIOS FOR STAR TREES

Before stating Theorem 5.4, we need a result which is equivalent to the Prime Number Theorem. Define

$$\theta(x) := \sum_{p \leq x, p \text{ prime}} \log p.$$

**Theorem 5.1.**  $\theta(x) \sim x$ .

More specifically, Rosser [7] proved

**Lemma 5.2.** For  $2 \leq x$ ,

$$\left(1 - \frac{2.85}{\log x}\right)x \leq \theta(x) \leq \left(1 + \frac{2.85}{\log x}\right)x.$$

As a consequence,

**Corollary 5.3.** For any  $n \geq 300$ ,

$$e^{n/2} \leq \prod_{p \leq np \text{ prime}} p \leq e^{3n/2}.$$

Now we can prove the main result for this section.

**Theorem 5.4.** When  $T$  is a star, it is  $\#P$ -complete to compute  $\#SPSCJ$ .

*Proof.* We have already verified that  $\#SPSCJ$  is in  $\#P$  in Lemma 3.13. To show  $\#P$ -complete, we give a polynomial time reduction from  $\#D3SAT$ .

Fix an arbitrary  $D3CNF$   $\Gamma = c_1 \wedge c_2 \wedge \dots \wedge c_k$  where each  $c_i$  is a clause and  $\Gamma$  has  $n$  variables.

Using the bound in Corollary 5.3, let  $n' = \max\{300, n + 5\}$ . Fix a prime number  $p$  which is greater than  $n'$  and at most  $5n'$ . Let

$$q := p - (n + 5).$$

Define a multiset  $\mathcal{D}(p) = \mathcal{A}(p) \cup \bigcup_{i \in [n]} \mathcal{B}_i(p) \cup \bigcup_{i \in [k]} \mathcal{C}_i(p)$  consisting of  $2 + 2n + 50k$  binary strings with coordinates

$$(x_1, y_1, x_2, y_2, \dots, x_n, y_n, e_1, \dots, e_{t(p)})$$

where

$$(5) \quad t(p) := 2(q + 4) + 2n(q + 3) + k(75 + 50q).$$

The coordinates  $e_1, e_2, \dots, e_{t(p)}$  are for the *additional ones*. In order to define each  $\eta \in \mathcal{D}(p)$ , we will give exact values for  $\eta[x_j]$  and  $\eta[y_j]$  for each  $j \in [n]$  and specify the number of additional ones that  $\eta$  will have. Definition 4.4 tells how to obtain the values of  $\eta[e_j]$  for each  $j \in [t(p)]$  from this information.

All strings in  $\mathcal{D}(p)$  come in pairs which are complementary on the first  $2n$  entries (Definition 4.8). As a result, we can use Fact 4.9 to each that the first  $2n$  coordinates are all ambiguous for  $\mathcal{D}(p)$ .

The set  $\mathcal{A}(p)$  consists of two strings,  $\alpha$  and  $\bar{\alpha}$ . Define  $\alpha$  to have  $\alpha[x_i] = \alpha[y_i] = 1$  for all  $i \in [n]$  and  $q + 4$  additional ones. Define  $\bar{\alpha}$  to be complementary to  $\alpha$  on the first  $2n$  entries and have  $q + 4$  additional ones.

For each  $j \in [n]$ , the set  $\mathcal{B}_j(p)$  will consist of two strings,  $\beta_j$  and  $\bar{\beta}_j$ . Define  $\beta_j$  to be the string with  $\beta_j[x_j] = \beta_j[y_j] = 1$  and for all  $j' \in [n]$  with  $j' \neq j$ ,  $\beta_j[x_{j'}] = \beta_j[y_{j'}] = 0$  and  $q + 3$  additional ones. Define  $\bar{\beta}_j$  to be complementary to  $\beta_j$  on the first  $2n$  entries and have  $q + 3$  additional ones.

For each  $i \in [k]$ , the set  $\mathcal{C}_i(p)$  will have 50 strings. These are obtained by adding  $q$  more additional ones to the 50 strings in  $\mathcal{C}_i$  which were defined through Table 1 (see Section 4.1). In other words, increase each entry in Column C of Table 1 by  $q$  to obtain  $\mathcal{C}_i(p)$ .

In summary, we have constructed the strings

$$\mathcal{D}(p) := \mathcal{A}(p) \cup \bigcup_{i \in [n]} \mathcal{B}_i(p) \cup \bigcup_{i \in [k]} \mathcal{C}_i(p).$$

From Definitions 2.7 and 4.3 and for each clause  $c_i$  in  $\Gamma$ , set

$$\begin{aligned} \mathcal{M}(p) &:= \mathcal{M}(\mathcal{D}(p)) & \mathcal{M}'(p) &:= \mathcal{M}'(\mathcal{D}(p)) \\ \mathcal{M}'_{c_i}(p) &:= \mathcal{M}'_{c_i}(\mathcal{D}(p)) & \mathcal{M}'_{\Gamma}(p) &:= \mathcal{M}'_{\Gamma}(\mathcal{D}(p)) \end{aligned}$$

As stated in Fact 4.5, each  $\mu \in \mathcal{M}(p)$  has  $\mu[e_j] = 0$  for all  $j \in [t]$ . Additionally, because all of the strings in  $\mathcal{D}(p)$  come in complementary pairs, the coordinates  $x_j$  and  $y_j$  are ambiguous for each  $j \in [n]$  (Fact 4.9). Thus there are  $2^{2n}$  medians  $\mu$ . More precisely,

$$(6) \quad \mathcal{M}(p) = \{0, 1\}^{2n} \times \{0\}^{t(p)}$$

which implies

$$\mathcal{M}'(p) = \{01, 10\}^n \times \{0\}^{t(p)}.$$

Define

$$\mathcal{H}(\mu, \mathcal{A}(p)) := \prod_{a \in \mathcal{A}(p)} H(\mu, a)!$$

and likewise we define  $\mathcal{H}(\mu, \mathcal{B}_j(p))$  and  $\mathcal{H}(\mu, \mathcal{C}_i(p))$  for each  $j \in [n]$  and  $i \in [k]$ . Therefore the number of SCJ scenarios admitted by  $\mu$  (Definition 2.5) can be expressed by

$$\mathcal{H}(\mu) := \mathcal{H}(\mu, \mathcal{A}(p)) \cdot \prod_{i \in [n]} \mathcal{H}(\mu, \mathcal{B}_i(p)) \cdot \prod_{i \in [k]} \mathcal{H}(\mu, \mathcal{C}_i(p)).$$

For each median  $\mu \in \mathcal{M}(p)$ , we will calculate  $d(\mu) \pmod p$ . To analyze  $\mathcal{H}(\mu)$  for each  $\mu \in \mathcal{M}$ , we define the following 3 properties that a  $\mu \in \mathcal{M}(p)$  may have.

*Property 1.*  $\sum_{i \in [n]} (\mu[x_i] + \mu[y_i]) = n$ .

*Property 2.*  $\mu \in \mathcal{M}'(p)$ .

*Property 3.*  $\mu \in \mathcal{M}'_\Gamma(p)$ .

First notice that these properties are nested. Any  $\mu \in \mathcal{M}(p)$  with Property 2 must also have Property 1. Likewise, if  $\mu$  has Property 3, it will also have Property 2. The next 4 claims divide  $\mathcal{M}(p)$  into 4 classes and examine  $\mathcal{H}(\mu)$  for medians in each class.

**Claim 5.5.** *For an arbitrary  $\mu \in \mathcal{M}(p)$ , if  $\mu$  does not have Property 1, then  $\mathcal{H}(\mu) \equiv 0 \pmod p$ .*

*Proof.* Let  $\mu$  be an arbitrary median in  $\mathcal{M}(p)$ . For  $\alpha \in \mathcal{A}(p)$ , Fact 4.7 gives  $H(\mu, \alpha) + H(\mu, \bar{\alpha}) = 2n + (q + 4) + (q + 4) = 2p - 2$ . Hence, there is an integer  $r$ ,  $q + 4 \leq r \leq 2n + q + 4$  such that  $H(\mu, \alpha) = r$  and

$$\mathcal{H}(\mu, \mathcal{A}(p)) = r!(2p - 2 - r)!.$$

Since  $\mu$  does not have Property 1, either  $H(\mu, \alpha) \geq (n+1) + (q+4) = p$  or  $H(\mu, \bar{\alpha}) \geq (n+1) + (q+4) = p$ . Therefore, either  $r \geq p$  or  $(2p - 2 - r) \geq p$ . In the first case,  $r!$  is divisible by  $p$  and in the second  $(2p - 2 - r)!$  is divisible by  $p$ . Therefore  $\mathcal{H}(\mu, \mathcal{A}(p)) \equiv 0 \pmod p$  and consequently  $\mathcal{H}(\mu) \equiv 0 \pmod p$ .  $\blacksquare$

**Claim 5.6.** *For an arbitrary  $\mu \in \mathcal{M}(p)$ , if  $\mu$  has Property 1, but does not have Property 2, then  $\mathcal{H}(\mu) \equiv 0 \pmod p$ .*

*Proof.* Suppose  $\mu \notin \mathcal{M}'(p)$  but  $\mu$  has Property 1. Because  $\mu \notin \mathcal{M}'(p)$ , there is a  $j_0 \in [n]$  such that  $\mu[x_{j_0}] = \mu[y_{j_0}]$ . If  $\mu[x_{j_0}] = 0$ , we have  $H(\mu, \beta_{j_0}) = (n + 2) + (q + 3) = p$ . Otherwise  $\mu[x_i] = 1$  which implies  $H(\mu, \bar{\beta}_{j_0}) = (n + 2) + (q + 3) = p$ . In either case,

$$\mathcal{H}(\mu, \mathcal{B}_{j_0}) = p!(p - 4)!$$

and consequently  $\mathcal{H}(\mu) \equiv 0 \pmod p$ .  $\blacksquare$

**Claim 5.7.** *For an arbitrary  $\mu \in \mathcal{M}(p)$ , if  $\mu$  has Properties 1 and 2, but does not have Property 3, then  $\mathcal{H}(\mu) \equiv 0 \pmod p$ .*

*Proof.* Let  $\mu$  have Properties 1 and 2, but not Property 3. Then  $\mu \in \mathcal{M}'(p) \setminus \mathcal{M}'_\Gamma(p)$ . Since  $\mu$  corresponds to a truth assignment which does not satisfy  $\Gamma$ , there is a clause  $c_{i_0}$  in  $\Gamma$  which is not

satisfied by this truth assignment. Therefore  $\mu \in \mathcal{M}'(p) \setminus \mathcal{M}'_{\mathcal{C}_{i_0}}(p)$ . By (1), before adding the  $q$  additional ones to each string from  $\mathcal{C}_{i_0}$ , we have

$$(7) \quad \{H(\mu, \nu_j^{i_0}) : \nu_j^{i_0} \in \mathcal{C}_{i_0}\} = \{(n-2)_{(1)}, (n-1)_{(6)}, n_{(3)}, (n+1)_{(15)}, \\ (n+2)_{(15)}, (n+3)_{(3)}, (n+4)_{(6)}, (n+5)_{(1)}\}.$$

To create  $\mathcal{C}_{i_0}(p)$ , we added  $q$  additional ones to each string in  $\mathcal{C}_{i_0}$ . Therefore

$$\{H(\mu, \nu_j^{i_0}) : \nu_j^{i_0} \in \mathcal{C}_{i_0}(p)\} = \{(p-7)_{(1)}, (p-6)_{(6)}, (p-5)_{(3)}, (p-4)_{(15)}, \\ (p-3)_{(15)}, (p-2)_{(3)}, (p-1)_{(6)}, p_{(1)}\}.$$

Therefore

$$\mathcal{H}(\mu, \mathcal{C}_{i_0}(p)) = (p-7)!(p-6)!^6(p-5)!^3(p-4)!^{15}(p-3)!^{15}(p-2)!^3(p-1)!^6p!$$

which is divisible by  $p$ . Therefore  $\mathcal{H}(\mu) \equiv 0 \pmod{p}$ .  $\blacksquare$

**Claim 5.8.** For an arbitrary  $\mu \in \mathcal{M}(p)$  having Properties 1, 2, and 3 (i.e.  $\mu \in \mathcal{M}'_\Gamma(p)$ ),  $d(\mu) \not\equiv 0 \pmod{p}$ .

*Proof.* Let  $\mu \in \mathcal{M}'_\Gamma(p)$ . Because it has Property 1,

$$\mathcal{H}(\mu, \mathcal{A}(p)) = (n+(q+4))!^2 = (p-1)!^2.$$

Since  $\mu$  has Property 2, for any  $i \in [n]$ ,

$$\mathcal{H}(\mu, \mathcal{B}_i(p)) = (n+(q+3))!^2 = (p-2)!^2.$$

Finally,  $\mu$  satisfies Property 3 which means  $\mu \in \mathcal{M}'_{\mathcal{C}_i}(p)$  for all clauses  $\mathcal{C}_i$  in  $\Gamma$ .

Recall that each string  $\eta \in \mathcal{C}_i(p)$  is a string  $\eta' \in \mathcal{C}_i(p)$  with  $q$  more additional ones. Therefore  $H(\mu, \eta) = H(\mu, \eta') + q$ . So, the multiset  $\mathcal{H}(\mu, \mathcal{C}_i(p))$  can be obtained from  $\mathcal{H}(\mu, \mathcal{C}_i)$  found in (2) by adding  $q$  to each element. As a result,

$$\mathcal{H}(\mu, \mathcal{C}_i(p)) = (p-6)!^7(p-5)!^6(p-4)!^{12}(p-3)!^{12}(p-2)!^6(p-1)!^7.$$

Therefore

$$(8) \quad \mathcal{H}(\mu) = (p-6)!^{7k}(p-5)!^{6k}(p-4)!^{12k}(p-3)!^{12k}(p-2)!^{6k+2n}(p-1)!^{7k+2}.$$

Because  $p$  is prime,  $\mathcal{H}(\mu) \not\equiv 0 \pmod{p}$ .  $\blacksquare$

Set

$$T(p) := \sum_{\mu \in \mathcal{M}(p)} \mathcal{H}(\mu).$$

Set  $S(p)$  to be the function of  $p$ , displayed in (8), for  $\mathcal{H}(\mu)$  which is the number of SCJ scenarios admitted by an arbitrary  $\mu \in \mathcal{M}'_\Gamma(p)$ . If we calculate  $T(p) \pmod{p}$ , the 4 claims show that

$$(9) \quad T(p) \equiv \sum_{\mu \in \mathcal{M}'_\Gamma(p)} \mathcal{H}(\mu) \equiv |\mathcal{M}'_\Gamma(p)|S(p) \pmod{p}.$$

If  $\gamma$  is the number of satisfying truth assignments for  $\Gamma$ , then  $\gamma = |\mathcal{M}'_\Gamma(p)|$  by Definition 4.3. Therefore

$$\gamma \cdot S(p) \equiv T(p) \pmod{p}.$$

Since  $p$  does not divide  $S(p)$  (Claim 5.8), there exists  $S'(p)$  such that  $S(p) \cdot S'(p) \equiv 1 \pmod{p}$ . Thus

$$\gamma \equiv S'(p) \cdot T(p) \pmod{p}.$$

While this alone is not sufficient to determine the value of  $\gamma$ , we can repeat this construction for many different prime values to obtain more congruences.

Recall  $p$  was fixed to be a prime greater than  $n'$  and at most  $5n'$ . Repeat the above construction for each prime  $p_1, p_2, \dots, p_m$  in this range. The result is a list of congruences:

$$\begin{aligned} \gamma &\equiv S'(p_1) \cdot T(p_1) \pmod{p_1}, \\ \gamma &\equiv S'(p_2) \cdot T(p_2) \pmod{p_2}, \\ &\vdots \\ \gamma &\equiv S'(p_m) \cdot T(p_m) \pmod{p_m}. \end{aligned}$$

Because  $p_1, p_2, \dots, p_m$  are all prime, the Chinese Remainder Theorem guarantees a solution for  $\gamma$  which is unique modulo  $\prod_{i \in [m]} p_i$ . By the lemma,

$$\prod_{i \in [m]} p_i = \frac{\prod_{\substack{p \leq 5n' \\ p \text{ prime}}} p}{\prod_{\substack{p \leq n' \\ p \text{ prime}}} p} \geq \frac{e^{5n'/2}}{e^{3n'/2}} = e^{n'} \geq e^n.$$

Since  $\gamma$  is the number of satisfying truth assignments for  $\Gamma$ , and there are only  $n$  literals which can realize one of two values,  $\gamma \leq 2^n$ . Since  $\prod_{i \in [m]} p_i \geq e^n > 2^n$ , the Chinese Remainder Theorem gives the exact value of  $\gamma$ .

In summary, for D3CNF  $\Gamma$  with  $n$  variables and  $k$  clauses, we use the Sieve of Eratosthenes to identify the primes between  $n'$  and  $5n'$ . This runs in  $O(n^2)$  time. Then for each prime  $p$  in this interval (which is at most  $2n$  primes), we create  $50k + 2n + 2$  binary strings of length  $2n + t(p)$  where  $t(p)$  is a polynomial (5) in  $n$  and  $p$  with  $p \in O(n)$ . Finally, the Chinese Remainder Theorem will solve the system of congruences in  $O(\log(p_1 p_2 \dots p_m))^2$  time [1]. For us, this is  $O(n \log n)^2$  because each prime is at most  $5n$  and  $m \leq 2n$ .

Therefore, if we had a polynomial time algorithm to determine the total number of most parsimonious scenarios for a collection of binary strings, then we have created here a polynomial time algorithm to determine the number of satisfying truth assignments for a D3CNF, a problem which is known to be #P-complete. This finishes the proof.  $\blacksquare$

## 6. GENERALIZATIONS FOR THE STAR TREE

In this section, we consider a generalization of #StarSPSCJ. First, fix a continuous function  $f : \mathbb{R} \rightarrow \mathbb{R}$ . Then define the following problem:

**Definition 6.1** (#StarSPSCJ( $f$ )). *Given an arbitrary  $m \in \mathbb{Z}^+$ , let  $S = \{\nu_i\}_{i=1}^m$  be an arbitrary multiset of binary strings. Determine the value of*

$$\sum_{\mu \in \mathcal{M}(S)} \prod_{i \in [m]} f(H(\nu_i, \mu)).$$

In the previous section, we showed that #StarSPSCJ( $f$ ) is #P-complete when  $f(x) := x!$ . Here we work toward the determining the computational complexity of #StarSPSCJ( $f$ ) for various functions  $f$ . First, we formalize a definition and develop a couple of tools.

**Definition 6.2.** *A function  $g : \mathbb{R} \rightarrow \mathbb{R}$  is strictly concave up if for any  $x, y, z \in \mathbb{R}$ ,  $x < y < z$ ,*

$$\frac{g(z) - g(x)}{z - x} > g(y).$$

*Equivalently,  $g'(x)$  is a strictly increasing function.*

**Lemma 6.3.** *If  $\log f(x)$  is a strictly concave up function, then for any  $x < y$  and  $a > 0$ ,*

$$\frac{f(x)f(y)}{f(x-a)f(y+a)} < 1.$$

*Proof.* By the intermediate value theorem, there is a  $c \in (x - a, x)$  and a  $d \in (y, y + a)$  such that  $(\log f)'(c) = \frac{1}{a}(\log f(x) - \log f(x - a))$  and  $(\log f)'(d) = \frac{1}{a}(\log f(y + a) - \log f(y))$ . Because  $\log f(x)$  is strictly increasing, by Definition 6.2,  $g'(c) < g'(d)$ . Therefore,

$$\begin{aligned} \frac{1}{a}(\log f(x) - \log f(x - a)) &< \frac{1}{a}(\log f(y + a) - \log f(y)) \\ \log f(x) - \log f(x - a) &< \log f(y + a) - \log f(y) \\ \log \frac{f(x)}{f(x - a)} &< \log \frac{f(y + a)}{f(y)} \\ \frac{f(x)}{f(x - a)} &< \frac{f(y + a)}{f(y)} \\ \frac{f(x)f(y)}{f(x - a)f(y + a)} &< 1 \end{aligned}$$

■

**Fact 6.4.** Fix  $k \in \mathbb{Z}^{\geq 0}$ . Let  $f(x)$  be a function such that  $\log f(x)$  is strictly concave up. Then

$$\min_{\substack{a, b \in \mathbb{Z}^{\geq 0} \\ a + b = k}} f(a)f(b) = f\left(\left\lfloor \frac{k}{2} \right\rfloor\right) f\left(\left\lceil \frac{k}{2} \right\rceil\right).$$

*Proof.* This is an immediate corollary of Lemma 6.3. Lemma 6.3

■

**Theorem 6.5.** Fix a function  $f(x) : \mathbb{Z}^{\geq 0} \rightarrow \mathbb{R}$  which satisfies the following properties:

- $\log f(x)$  is strictly concave up and
- for all but finitely many  $n \in \mathbb{Z}$ ,  $n \geq 2$ ,

$$\frac{f(n - 2)[f(n + 1)]^3[f(n + 2)]^3 f(n + 5)}{f(n - 1)[f(n)]^3[f(n + 3)]^3 f(n + 4)} > 1.$$

For arbitrary  $m, s \in \mathbb{Z}^{\geq 0}$  and  $D \in \mathbb{R}$ , let  $S := \{\nu_1, \nu_2, \dots, \nu_m\}$  be a multiset of binary strings, each of length  $s$ . Then it is NP-hard to determine if there is a median  $\mu$  for  $S$  such that

$$(10) \quad \prod_{i \in [m]} f(H(\nu_i, \mu)) \leq D.$$

*Proof.* Fix a function  $f(x)$  with the properties listed in the theorem.

To prove NP-hardness, we will provide a reduction from D3CNF. Given a D3CNF, the idea is to define a multiset,  $\mathcal{D}$ , of binary strings with the following properties:

- Each median which corresponds to a satisfying truth assignment for  $\Gamma$  will have

$$\prod_{\eta \in \mathcal{D}} f(H(\eta, \mu)) = \kappa$$

for some  $\kappa \in \mathbb{R}$  which is dependent on the number of variables and clauses in  $\Gamma$ .

- Each other median will have

$$\prod_{\eta \in \mathcal{D}} f(H(\eta, \mu)) > \kappa.$$

For arbitrary  $n, k \in \mathbb{Z}^{\geq 0}$ , let  $\Gamma = c_1 \wedge c_2 \wedge \dots \wedge c_k$  be an arbitrary D3CNF with  $n$  variables. Create a total of  $158k + 28kn$  strings of length  $2n + 260k + 35kn$  with coordinates

$$(x_1, y_1, x_2, y_2, \dots, x_n, y_n, e_1, e_2, \dots, e_t)$$

where  $t = 260k + 35kn$ .



This multiset of binary strings will be defined as the union of three multisets:

$$\mathcal{D} = \mathcal{A} \uplus \bigsqcup_{i \in [n]} \mathcal{B}_i \uplus \bigsqcup_{i \in [k]} \mathcal{C}_i.$$

As in Definition 4.4, we will define each string  $\eta \in \mathcal{D}$  by explicitly giving the values of  $\eta[x_j]$  and  $\eta[y_j]$  for each  $i \in [n]$  and telling the number of additional ones in  $\eta$ .

The collection  $\mathcal{A}$  contains  $108k$  strings. For  $a \in [t]$ , let  $\alpha^{(+a)}$  be the string with  $\alpha[x_i] = \alpha[y_i] = 1$  for all  $1 \leq i \leq n$  and  $a$  additional ones. Define  $\bar{\alpha}^{(+a)}$  to be the binary string which is complementary to  $\alpha^{(+a)}$  on the first  $2n$  coordinates and has  $a$  additional ones. The multiset  $\mathcal{A}$  will consist of the following strings:

- $k$  copies each of  $\alpha^{(+0)}$  and  $\bar{\alpha}^{(+0)}$ ,
- $8k$  copies each of  $\alpha^{(+1)}$  and  $\bar{\alpha}^{(+1)}$ ,
- $18k$  copies each of  $\alpha^{(+2)}$  and  $\bar{\alpha}^{(+2)}$ ,
- $18k$  copies each of  $\alpha^{(+3)}$  and  $\bar{\alpha}^{(+3)}$ ,
- $8k$  copies each of  $\alpha^{(+4)}$  and  $\bar{\alpha}^{(+4)}$ ,
- $k$  copies each of  $\alpha^{(+5)}$  and  $\bar{\alpha}^{(+5)}$ .

The collection  $\mathcal{B} = \bigsqcup_{i \in [n]} \mathcal{B}_i$  contains  $28kn$  strings. For each  $i \in [n]$ ,  $a \in [t]$ , let  $\beta_i^{(+a)}$  be the string with  $\beta_i[x_i] = \beta_i[y_i] = 1$ ,  $\beta_i[x_j] = \beta_i[y_j] = 0$  when  $j \neq i$ , and  $a$  additional ones. Define the binary string  $\bar{\beta}_i^{(+a)}$  to be complementary to  $\beta_i^{(+a)}$  on the first  $2n$  coordinates and have  $a$  additional ones. The collection  $\mathcal{B}_i$  consists of the following  $28k$  strings.

- $k$  copies each of  $\beta_i^{(+1)}$  and  $\bar{\beta}_i^{(+1)}$ ,
- $6k$  copies each of  $\beta_i^{(+2)}$  and  $\bar{\beta}_i^{(+2)}$ ,
- $6k$  copies each of  $\beta_i^{(+3)}$  and  $\bar{\beta}_i^{(+3)}$ ,
- $k$  copies each of  $\beta_i^{(+4)}$  and  $\bar{\beta}_i^{(+4)}$ .

The collection  $\mathcal{C} = \bigsqcup_{i \in [k]} \mathcal{C}'_i$  contains  $50k$  strings. Each set  $\mathcal{C}'_i$ , which is associated with clause  $c_i$ , consists of  $50$  strings. In Section 4.1, we defined set  $\mathcal{C}_i$  through Table 1. For each  $\nu_j^i \in \mathcal{C}_i$ , create  $\hat{\nu}_j^i$  by increasing the number of additional ones in  $\nu_j^i$  by one. Then

$$\mathcal{C}'_i = \{\hat{\nu}_j^i : \nu_j^i \in \mathcal{C}_i\}.$$

Let  $\mathcal{M}$  be the set of all medians for  $\mathcal{D}$ . From Definitions 2.7 and 4.3 and for each clause  $c_i$  in  $\Gamma$ , set

$$\begin{aligned} \mathcal{M} &:= \mathcal{M}(\mathcal{D}) & \mathcal{M}' &:= \mathcal{M}'(\mathcal{D}) \\ \mathcal{M}'_{c_i} &:= \mathcal{M}'_{c_i}(\mathcal{D}) & \mathcal{M}'_{\Gamma} &:= \mathcal{M}'_{\Gamma}(\mathcal{D}) \end{aligned}$$

According to Remark 4.4, all medians  $\mu$  must have  $\mu[e_i] = 0$  for all  $i \in [t]$ . In  $\mathcal{A}$ ,  $\mathcal{B}$ , and  $\mathcal{C}$ , the strings come in pairs where one is complementary to the other on the first  $2n$  coordinates. By Fact 4.9, each of the  $x_i$  and  $y_i$  coordinates are ambiguous. Therefore

$$\begin{aligned} \mathcal{M} &= \{0, 1\}^{2n} \times \{0\}^t. \\ \mathcal{M}' &= \{01, 10\}^n \times \{0\}^t. \end{aligned}$$

Define

$$\mathcal{H}(\mu, \mathcal{A}) := \prod_{a \in \mathcal{A}} f(H(\mu, a)).$$

Similarly define  $\mathcal{H}(\mu, \mathcal{B}_i)$  and  $\mathcal{H}(\mu, \mathcal{C}'_j)$ . Set

$$\mathcal{H}(\mu) := \mathcal{H}(\mu, \mathcal{A}) \cdot \prod_{i \in [n]} \mathcal{H}(\mu, \mathcal{B}_i) \cdot \prod_{j \in [k]} \mathcal{H}(\mu, \mathcal{C}'_j).$$

For each  $\mu \in \mathcal{M}$ , we need a lower bound for  $\mathcal{H}(\mu)$  and for each  $\mu \in \mathcal{M}'_\Gamma$  we need an exact value for  $\mathcal{H}(\mu)$ . We divide  $\mathcal{M}$  into 4 classes using the following three properties of a median  $\mu \in \mathcal{M}$ .

*Property 1.*  $\sum_{i \in [n]} (\mu[x_i] + \mu[y_i]) = n$ .

*Property 2.*  $\mu \in \mathcal{M}'$ .

*Property 3.*  $\mu \in \mathcal{M}'_\Gamma$ .

Notice that these properties are nested. Any median  $\mu \in \mathcal{M}$  with Property 2, must also have Property 1. Further, any  $\mu \in \mathcal{M}$  with Property 3 must also have Property 2. The following properties provide lower bounds for medians according to their properties.

**Claim 6.6.** *If  $\mu \in \mathcal{M}$  has Property 1, then*

$$(11) \quad \mathcal{H}(\mu, \mathcal{A}) = [f(n)]^{2k} [f(n+1)]^{16k} [f(n+2)]^{36k} [f(n+3)]^{36k} [f(n+4)]^{16k} [f(n+5)]^{2k} \\ =: \alpha_{good}.$$

*Otherwise,*

$$(12) \quad \mathcal{H}(\mu, \mathcal{A}) \geq [f(n-1)f(n+1)]^k [f(n)f(n+2)]^{8k} [f(n+1)f(n+3)]^{18k} \\ \cdot [f(n+2)f(n+4)]^{18k} [f(n+3)f(n+5)]^{8k} [f(n+4)f(n+6)]^k \\ =: \alpha_{bad}.$$

*Proof.* If  $\mu$  has Property 1, then  $H(\mu, \alpha^{(+0)}) = H(\mu, \bar{\alpha}^{(+0)}) = n$  because  $\alpha^{(+0)}[x_i] = \alpha^{(+0)}[y_i] = 1$  for all  $i \in [n]$  while  $\mu$  only has  $n$  ones in the first  $2n$  entries. Because  $\mu[e_i] = 0$  for all  $i \in [t]$ , by Definition 4.4,

$$H(\mu, \alpha^{(+a)}) = H(\mu, \bar{\alpha}^{(+a)}) = n + a.$$

The equation (11) follows directly from this.

If  $\mu$  does not have Property 1, then either  $\mu$  has more than  $n$  ones in the first  $2n$  entries, implying  $H(\mu, \alpha^{(+0)}) > n$  or  $\mu$  has less than  $n$  ones in the first  $2n$  entries, implying  $H(\mu, \bar{\alpha}^{(+0)}) > n$ . By Fact 4.7,  $H(\mu, \alpha^{(+0)}) + H(\mu, \bar{\alpha}^{(+0)}) = 2n$ . By Fact 6.4 and the above observations,

$$H(\mu, \alpha^{(+0)})H(\mu, \bar{\alpha}^{(+0)}) \geq f(n-1)f(n+1) \\ H(\mu, \alpha^{(+a)})H(\mu, \bar{\alpha}^{(+a)}) \geq f(n-1+a)f(n+1+a).$$

This suggests the lower bound in (12). ■

**Claim 6.7.** *For any  $\mu \in \mathcal{M}$  and each  $i \in [n]$ ,*

$$(13) \quad \mathcal{H}(\mu, \mathcal{B}_i) \geq [f(n+1)]^{2k} [f(n+2)]^{12k} [f(n+3)]^{12k} [f(n+4)]^{2k} =: \beta_{good}$$

*If  $\mu$  has Property 2, then for every  $i \in [n]$ ,*

$$\mathcal{H}(\mu, \mathcal{B}_i) = \beta_{good}$$

*If  $\mu$  satisfies Property 1, but not Property 2, then there exists  $i_0 \in [n]$  such that*

$$(14) \quad \mathcal{H}(\mu, \mathcal{B}_{i_0}) = [f(n-1)f(n+3)]^k [f(n)f(n+4)]^{6k} \\ \cdot [f(n+1)f(n+5)]^{6k} [f(n+2)f(n+6)]^k \\ =: \beta_{bad}.$$

*Proof.* For any  $\mu \in \mathcal{M}$ , by Fact 4.7,

$$H(\mu, \beta^{(+0)}) + H(\mu, \bar{\beta}^{(+0)}) = 2n.$$

By Fact 6.4, for each  $a \in \mathbb{Z}^{\geq 0}$ ,

$$\begin{aligned} H(\mu, \beta^{(+0)}) \cdot H(\mu, \bar{\beta}^{(+0)}) &\geq [f(n)]^2, \text{ and} \\ H(\mu, \beta^{(+a)}) \cdot H(\mu, \bar{\beta}^{(+a)}) &\geq [f(n+a)]^2. \end{aligned}$$

Therefore, for any  $\mu \in \mathcal{M}$ ,

$$\mathcal{H}(\mu, \mathcal{B}_i) \geq \beta_{good}.$$

If  $\mu \in \mathcal{M}'$ , then for each  $i \in [n]$ ,  $\mu[x_i] \neq \mu[y_i]$ . On the other hand, for each  $i \in [n]$ ,  $j \in [n]$ ,  $\beta_i^{(+a)}[x_j] = \beta_i^{(+a)}[y_j]$ . Therefore for any  $i, j \in [n]$ ,

$$H((\mu[x_j], \mu[y_j]), (\beta_i^{(+a)}[x_j], \beta_i^{(+a)}[y_j])) = 1.$$

The same holds if  $\beta_i^{(+a)}$  is replaced with  $\bar{\beta}_i^{(+a)}$ . Therefore,

$$\begin{aligned} h(\mu, \beta_i^{(+0)}) &= h(\mu, \bar{\beta}_i^{(+0)}) = n, \text{ and} \\ h(\mu, \beta_i^{(+a)}) &= h(\mu, \bar{\beta}_i^{(+a)}) = n + a. \end{aligned}$$

As a result  $\mathcal{H}(\mu) = \beta_{good}$ .

If  $\mu$  satisfies Property 1 but not Property 2, then we can define a tighter lower bound on  $\mathcal{H}(\mu)$ . In particular, because  $\mu \notin \mathcal{M}'$ , there exists  $i_0 \in [n]$  such that  $\mu[x_{i_0}] = \mu[y_{i_0}]$ . Recall  $\beta_{i_0}^{(+a)}[x_{i_0}] = \beta_{i_0}^{(+a)}[y_{i_0}] = 1$  and  $\bar{\beta}_{i_0}^{(+a)}[x_{i_0}] = \bar{\beta}_{i_0}^{(+a)}[y_{i_0}] = 0$ . Therefore,

$$\begin{aligned} \mu[x_{i_0}] = 1 &\Rightarrow H((\mu[x_{i_0}], \mu[y_{i_0}]), (\beta_{i_0}^{(+a)}[x_{i_0}], \beta_{i_0}^{(+a)}[y_{i_0}])) = 0 \text{ and} \\ &H((\mu[x_{i_0}], \mu[y_{i_0}]), (\bar{\beta}_{i_0}^{(+a)}[x_{i_0}], \bar{\beta}_{i_0}^{(+a)}[y_{i_0}])) = 2, \text{ and} \\ \mu[x_{i_0}] = 0 &\Rightarrow H((\mu[x_{i_0}], \mu[y_{i_0}]), (\bar{\beta}_{i_0}^{(+a)}[x_{i_0}], \bar{\beta}_{i_0}^{(+a)}[y_{i_0}])) = 0 \text{ and} \\ &H((\mu[x_{i_0}], \mu[y_{i_0}]), (\beta_{i_0}^{(+a)}[x_{i_0}], \beta_{i_0}^{(+a)}[y_{i_0}])) = 2. \end{aligned}$$

Because  $\mu$  satisfies Property 1, there are exactly  $n$  ones among the first  $2n$  coordinates. Suppose  $\mu[x_{i_0}] = \mu[y_{i_0}] = 1$ . Let  $S := \{x_j, y_j : j \in [n], j \neq i_0\}$ . Then  $\mu$  has  $n-2$  ones and  $n$  zeros among the coordinates in  $S$ . However,  $\beta_{i_0}^{(+a)}$  takes the value 0 on the coordinates of  $S$  and  $\bar{\beta}_{i_0}^{(+a)}$  takes the values 1 on the coordinates of  $S$ . Therefore,

$$\begin{aligned} \mu[x_{i_0}] = 1 &\Rightarrow H(\mu, \beta_{i_0}^{(+0)}) = 0 + (n-2) \text{ and} \\ &H(\mu, \bar{\beta}_{i_0}^{(+0)}) = 2 + n, \text{ and} \\ \mu[x_{i_0}] = 0 &\Rightarrow H(\mu, \bar{\beta}_{i_0}^{(+0)}) = 0 + (n-2) \text{ and} \\ &H(\mu, \beta_{i_0}^{(+0)}) = 2 + n \end{aligned}$$

As a result,

$$\begin{aligned} H(\mu, \beta_{i_0}^{(+0)})H(\mu, \bar{\beta}_{i_0}^{(+0)}) &= (n-2)(n+2) \\ H(\mu, \beta_{i_0}^{(+a)})H(\mu, \bar{\beta}_{i_0}^{(+a)}) &= (n-2+a)(n+2+a) \end{aligned}$$

Taking into account all binary strings in  $\mathcal{B}_{i_0}$ , we conclude  $\mathcal{H}(\mu, \mathcal{B}_{i_0}) = \beta_{bad}$  in (14).  $\blacksquare$

**Fact 6.8.** For the quantities defined in Claim 6.7,  $\beta_{good} < \beta_{bad}$ . Further, if  $\mu \in \mathcal{M} \setminus \mathcal{M}'$  and satisfies Property 1, then  $\prod_{i \in [n]} \mathcal{H}(\mu, \mathcal{B}_i) \geq \beta_{bad} \beta_{good}^{k-1}$ . If  $\mu \in \mathcal{M} \setminus \mathcal{M}'$  and does not satisfy Property 1, then  $\prod_{i \in [n]} \mathcal{H}(\mu, \mathcal{B}_i) \geq \beta_{good}^k$ .

*Proof.* Observe

$$\begin{aligned} \frac{\beta_{bad}}{\beta_{good}} &= \left[ \frac{f(n-1)f(n)^6f(n+1)^4f(n+4)^4f(n+5)^6f(n+6)}{f(n+2)^{11}f(n+3)^{11}} \right]^k \\ &= \left[ \frac{f(n-1)f(n+6)}{f(n+2)f(n+3)} \right]^k \cdot \left[ \frac{f(n)f(n+5)}{f(n+2)f(n+3)} \right]^{6k} \left[ \frac{f(n+1)f(n+4)}{f(n+2)f(n+3)} \right]^{4k} \\ &> 1 \end{aligned}$$

where the last inequality follows from Lemma 6.3. ■

**Claim 6.9.** For any  $\mu \in \mathcal{M}$  and for each  $j \in [k]$ ,

$$\mathcal{H}(\mu, \mathcal{C}'_i) \geq [f(n+2)]^{25}[f(n+3)]^{25} =: \gamma_{min}.$$

If  $\mu \in \mathcal{M}'_\Gamma$ , then for each  $j \in [k]$ ,

$$(15) \quad \mathcal{H}(\mu, \mathcal{C}'_i) = [f(n)]^7[f(n+1)]^6[f(n+2)]^{12}[f(n+3)]^{12}[f(n+4)]^6[f(n+5)]^7 =: \gamma_{good}.$$

If  $\mu \in \mathcal{M}' \setminus \mathcal{M}'_\Gamma$ , then there exists  $i_0 \in [k]$  such that

$$(16) \quad \begin{aligned} \mathcal{H}(\mu, \mathcal{C}'_{i_0}) &= f(n-1)[f(n)]^6[f(n+1)]^3[f(n+2)]^{15} \\ &\quad \cdot [f(n+3)]^{15}[f(n+4)]^3[f(n+5)]^6f(n+6) \\ &=: \gamma_{bad}. \end{aligned}$$

*Proof.* Let  $\mu$  be an arbitrary median in  $\mathcal{M}$ . By Remark 4.12, the binary strings in  $\mathcal{C}_i$  come in pairs that are complementary on the first  $2n$  entries. Further, if  $\eta$  and  $\bar{\eta}$  are in  $\mathcal{C}_i$  and are complementary on the first  $2n$  coordinates, then  $e(\eta) + e(\bar{\eta}) = 3$  where  $e$  is the function specifying the number of additional ones. By the definition of  $\mathcal{C}'_i$ , the strings still come in complementary pairs,  $(\eta, \bar{\eta})$ , but here  $e(\eta) + e(\bar{\eta}) = 5$ . By Fact 4.7, for each of the 25 pairs in  $\mathcal{C}'_i$ ,

$$H(\mu, \eta) + H(\mu, \bar{\eta}) = 2n + 5.$$

Then by Fact 6.4,

$$f(H(\mu, \eta))f(H(\mu, \bar{\eta})) \geq f(n+2)f(n+3).$$

Now suppose  $\mu \in \mathcal{M}'_\Gamma$ . This implies  $\mu \in \mathcal{M}'_{c_i}$  for all clauses  $c_i$  in  $\Gamma$ . By the definition of  $\mathcal{C}'_i$ , for each  $\hat{\nu}_j^i \in \mathcal{C}'_i$ ,  $H(\mu, \hat{\nu}_j^i) = H(\mu, \nu_j^i) + 1$  where  $\nu_j^i \in \mathcal{C}_i$ . From (2), we see

$$\{H(\mu, \hat{\nu}_j^i) : j \in [50]\} = \{n_{(7)}, (n+1)_{(6)}, (n+2)_{(12)}, (n+3)_{(12)}, (n+4)_{(6)}, (n+5)_{(7)}\}.$$

This immediately implies  $\mathcal{H}(\mu, \mathcal{C}'_i) = \gamma_{good}$  in (15).

Finally, suppose  $\mu \in \mathcal{M}' \setminus \mathcal{M}'_\Gamma$ . Using the bijection in Definition 4.1,  $\mu$  must correspond to a truth assignment which does not satisfy  $\Gamma$ . So there must be a clause  $c_{i_0}$  in  $\Gamma$  which is not satisfied. Therefore  $\mu \in \mathcal{M}' \setminus \mathcal{M}'_{c_{i_0}}$ . From (1), adding 1 to each Hamming for the 1 extra additional one, we obtain

$$(17) \quad \begin{aligned} \{H(\mu, \nu_j^{i_0}) : j \in [50]\} &= \{(n-1)_{(1)}, n_{(6)}, (n+1)_{(3)}, (n+2)_{(15)}, \\ &\quad (n+3)_{(15)}, (n+4)_{(3)}, (n+5)_{(6)}, (n+6)_{(1)}\}. \end{aligned}$$

This directly implies  $\mathcal{H}(\mu, \mathcal{C}_{i_0}) = \gamma_{bad}$  in (16). ■

**Fact 6.10.** For the quantities defined in Claim 6.9,  $\gamma_{good} < \gamma_{bad}$ . As a result, when  $\mu \in \mathcal{M}' \setminus \mathcal{M}'_\Gamma$ , we obtain the bound

$$\mathcal{H}(\mu, \mathcal{C}) \geq \gamma_{bad}\gamma_{good}^{k-1}.$$

*Proof.* Indeed, this was our initial assumption:

$$\frac{\gamma_{bad}}{\gamma_{good}} = \frac{f(n-1)[f(n+2)]^3[f(n+3)]^3f(n+6)}{f(n)[f(n+1)]^3[f(n+4)]^3[f(n+5)]} > 1.$$

The bound for  $\mathcal{H}(\mu, \mathcal{C})$  results from the fact that  $\mu \in \mathcal{M}'$  and so for each clause  $c_i$ , it either corresponds to a satisfying truth assignment for  $c_i$  or a non-satisfying truth assignment for  $c_i$ . ■

In summary, Claims 6.6, 6.7, and 6.9 along with Facts 6.8 and 6.10, we obtain the following bounds. If  $\mu \in \mathcal{M}'_\Gamma$ ,

$$\mathcal{H}(\mu) = \alpha_{good}\beta_{good}^n\gamma_{good}^k =: h_3.$$

If  $\mu \in \mathcal{M}' \setminus \mathcal{M}'_\Gamma$ ,

$$\mathcal{H}(\mu) \geq \alpha_{good}\beta_{good}^n\gamma_{bad}\gamma_{good}^{k-1} =: h_2.$$

If  $\mu \in \mathcal{M} \setminus \mathcal{M}'$  and has Property 1,

$$\mathcal{H}(\mu) \geq \alpha_{good}\beta_{bad}\beta_{good}^{n-1}\gamma_{min}^k =: h_1.$$

If  $\mu \in \mathcal{M}$  but does not have Property 1,

$$\mathcal{H}(\mu) \geq \alpha_{bad}\beta_{good}^n\gamma_{min}^k =: h_0.$$

In order to complete the proof, we only need to show  $h_3 < h_i$  for  $i \in \{0, 1, 2\}$ . By one of our assumptions about  $f(x)$ , we have already verified in Fact 6.10

$$\frac{h_2}{h_3} = \frac{\gamma_{bad}}{\gamma_{good}} > 1.$$

Next observe

$$\begin{aligned} \frac{h_1}{h_2} &= \frac{\beta_{bad}}{\beta_{good}} \cdot \frac{\gamma_{min}^k}{\gamma_{bad}\gamma_{good}^{k-1}} \\ &> \frac{\beta_{bad}}{\beta_{good}} \cdot \frac{\gamma_{min}^k}{\gamma_{bad}^k} \\ &= \left[ \frac{f(n-1)f(n+3)}{[f(n+1)]^2} \left[ \frac{f(n)f(n+4)}{[f(n+2)]^2} \right]^6 \left[ \frac{f(n+1)f(n+5)}{[f(n+3)]^2} \right]^6 \frac{f(n+2)f(n+6)}{[f(n+4)]^2} \right]^k \\ &\quad \cdot \left[ \frac{[f(n+2)]^{10}[f(n+3)]^{10}}{[f(n-1)[f(n)]^6[f(n+1)]^3[f(n+4)]^3[f(n+5)]^6f(n+6)} \right]^k \\ &= \left[ \frac{f(n+1)f(n+4)}{f(n+2)f(n+3)} \right]^k \\ &> 1 \end{aligned}$$

where the last inequality follows from Lemma 6.3.

Finally

$$\begin{aligned}
 \frac{h_0}{h_2} &= \frac{\alpha_{bad}}{\alpha_{good}} \frac{\gamma_{min}^k}{\gamma_{bad} \gamma_{good}^{k-1}} \\
 &> \frac{\alpha_{bad}}{\alpha_{good}} \cdot \frac{\gamma_{min}^k}{\gamma_{bad}^k} \\
 &= \left[ \frac{f(n-1)f(n+1)}{[f(n)]^2} \left[ \frac{f(n)f(n+2)}{[f(n+1)]^2} \right]^8 \left[ \frac{f(n+1)f(n+3)}{[f(n+2)]^2} \right]^{18} \right. \\
 &\quad \cdot \left. \left[ \frac{f(n+2)f(n+4)}{[f(n+3)]^2} \right]^{18} \left[ \frac{f(n+3)f(n+5)}{[f(n+4)]^2} \right]^8 \frac{f(n+4)f(n+6)}{[f(n+5)]^2} \right]^k \\
 &\quad \cdot \left[ \frac{[f(n+2)]^{10} [f(n+3)]^{10}}{f(n-1)[f(n)]^6 [f(n+1)]^3 [f(n+4)]^3 [f(n+5)]^6 f(n+6)} \right]^k \\
 &= 1.
 \end{aligned}$$

Therefore for any  $\hat{\mu} \in \mathcal{M}'_\Gamma$  and  $\mu \in \mathcal{M} \setminus \mathcal{M}'_\Gamma$ , then  $\mathcal{H}(\hat{\mu}) < \mathcal{H}(\mu)$ . Thus, if we could determine, in polynomial time, if there was a most parsimonious  $\mu$  with  $h(\mu) \leq h_3$ , then we could determine whether or not  $\Gamma$  had a satisfying truth assignment in polynomial time.  $\blacksquare$

**Proposition 6.11.** *Fix a function  $f(x) : \mathbb{Z}^{\geq 0} \rightarrow \mathbb{R}$  which satisfies the following properties:*

- $\log f(x)$  is strictly concave up, and
- for all but finitely many  $n \in \mathbb{Z}$ ,  $n \geq 2$ ,

$$\frac{f(n-2)[f(n+1)]^3[f(n+2)]^3f(n+5)}{f(n-1)[f(n)]^3[f(n+3)]^3f(n+4)} > 1.$$

For arbitrary  $m, s \in \mathbb{Z}^{>0}$  and  $D \in \mathbb{R}$ , let  $S := \{\nu_1, \nu_2, \dots, \nu_m\}$  be a multiset of binary strings, each of length  $s$ . Then it is  $\#P$ -hard to determine the number of medians  $\mu$  for  $S$  such that

$$(18) \quad \prod_{i \in [m]} f(H(\nu_i, \mu)) \leq D.$$

*Proof.* This follows from the proof of Theorem 6.5. In the proof, we showed that each median  $\mu \in \mathcal{M}'_\Gamma$  had

$$\prod_{i \in [m]} f(H(\nu_i, \mu)) = \alpha_{good} \beta_{good}^n \gamma_{good}^k.$$

For all other medians  $\mu \in \mathcal{M} \setminus \mathcal{M}'_\Gamma$ ,

$$\prod_{i \in [m]} f(H(\nu_i, \mu)) > \alpha_{good} \beta_{good}^n \gamma_{good}^k.$$

Because  $\mathcal{M}'_\Gamma$  is in one-to-one correspondence with the satisfying truth assignments for  $\Gamma$ , and  $\#D3CNF$  is  $\#P$ -complete, we have proved the proposition.  $\blacksquare$

**Theorem 6.12.** *Fix a function  $f(x) : \mathbb{Z}^{\geq 0} \rightarrow \mathbb{R}$  which satisfies the following properties:*

- $\log f(x)$  is strictly concave up and
- for all but finitely many  $n \in \mathbb{Z}$ ,  $n \geq 2$ ,

$$\frac{f(n-2)[f(n+1)]^3[f(n+2)]^3f(n+5)}{f(n-1)[f(n)]^3[f(n+3)]^3f(n+4)} < 1.$$

For arbitrary  $m, s \in \mathbb{N}$  and  $D \in \mathbb{R}$ , let  $S := \{\nu_1, \nu_2, \dots, \nu_m\}$  be a multiset of binary strings, each of length  $s$ . Then it is NP-hard to determine if there is a median  $\mu$  for  $S$  such that

$$\prod_{i \in [m]} f(H(\nu_i, \mu)) \leq D.$$

*Proof.* This proof closely mirrors the proof of Theorem 6.5. Here we will note the changes that need to be made.

This time, we define  $98k + 24kn$  binary strings, each of length  $2n + 245k + 60kn$  with coordinates

$$(x_1, y_1, \dots, x_n, y_n, e_1, \dots, e_t).$$

Let  $\alpha^{(+a)}$  and  $\bar{\alpha}^{(+a)}$  be defined as before. The collection  $\mathcal{A}$  will now consist of the following  $72k$  strings:

- $4k$  copies each of  $\alpha^{(+1)}$  and  $\bar{\alpha}^{(+1)}$ ,
- $14k$  copies each of  $\alpha^{(+2)}$  and  $\bar{\alpha}^{(+2)}$ ,
- $14k$  copies each of  $\alpha^{(+3)}$  and  $\bar{\alpha}^{(+3)}$ ,
- $4k$  copies each of  $\alpha^{(+4)}$  and  $\bar{\alpha}^{(+4)}$ .

Define  $\beta_i^{(+a)}$  and  $\bar{\beta}_i^{(+a)}$  as before. The collection  $\mathcal{B}_i$  will now consist of the following  $24k$  strings:

- $6k$  copies each of  $\beta_i^{(+2)}$  and  $\bar{\beta}_i^{(+2)}$ ,
- $6k$  copies each of  $\beta_i^{(+3)}$  and  $\bar{\beta}_i^{(+3)}$ .

Following the explanation found in Section 4.1, Table 3 defines 26 binary strings  $\bar{\mathcal{C}}_i$  for a clause. This time, we will add 1 additional one to each of the 26 strings in  $\bar{\mathcal{C}}_i$  to create  $\mathcal{C}'_i$ .

Using the same Properties 1, 2, and 3 as before, we obtain the following values which are analogous to the bounds in Claims 6.6, 6.7, and 6.9:

$$\begin{aligned} \alpha_{good} &:= [f(n+1)]^{8k} [f(n+2)]^{28k} [f(n+3)]^{28k} [f(n+4)]^{8k} \\ \alpha_{bad} &:= [f(n)f(n+2)]^{4k} [f(n+1)f(n+3)]^{14k} [f(n+2)f(n+4)]^{14k} [f(n+3)f(n+5)]^{4k} \\ \beta_{good} &:= [f(n+2)]^{12k} [f(n+3)]^{12k} \\ \beta_{min} &:= [f(n+2)]^{12k} [f(n+3)]^{12k} \\ \beta_{bad} &:= [f(n)f(n+4)]^{6k} [f(n+1)f(n+5)]^{6k} \\ \gamma_{good} &:= f(n-1)[f(n)]^3 [f(n+1)]^3 [f(n+2)]^6 [f(n+3)]^6 [f(n+4)]^3 [f(n+5)]^3 f(n+6) \\ \gamma_{bad} &:= [f(n)]^4 [f(n+1)]^6 [f(n+2)]^3 [f(n+3)]^3 [f(n+4)]^6 [f(n+5)]^4 \\ \gamma_{min} &:= [f(n+2)]^{13} [f(n+3)]^{13} \end{aligned}$$

By our assumption about  $f(x)$ ,

$$\frac{\gamma_{bad}}{\gamma_{good}} = \frac{f(n)[f(n+1)]^3 [f(n+4)]^3 f(n+5)}{f(n-1)[f(n+2)]^3 [f(n+3)]^3 f(n+6)} > 1$$

which implies  $\gamma_{bad} > \gamma_{good}$ .

Next we determine the values of  $h_0, h_1, h_2, h_3$  in this setting. As before, if  $\mu$  has Property  $i$ , but not property  $i+1$ , then  $\mathcal{H}(\mu) \geq h_i$ . Further, if  $\mu$  has Property 3,  $\mathcal{H}(\mu) = h_3$ . If  $\mu$  does not have

Property 1, then  $\mathcal{H}(\mu) \geq h_0$ .

$$\begin{aligned} h_3 &:= \alpha_{good} \beta_{good}^n \gamma_{good}^k. \\ h_2 &:= \alpha_{good} \beta_{good}^n \gamma_{bad} \gamma_{good}^{k-1}. \\ h_1 &:= \alpha_{good} \beta_{bad} \beta_{min}^{n-1} \gamma_{min}^k. \\ h_0 &:= \alpha_{bad} \beta_{min}^n \gamma_{min}^k. \end{aligned}$$

As in Theorem 6.5, we will show  $h_0 < h_1, h_2, h_3$ .

By our assumption about  $f(x)$ ,

$$\frac{h_2}{h_3} = \frac{\gamma_{bad}}{\gamma_{good}} > 1.$$

Also

$$\begin{aligned} \frac{h_1}{h_2} &= \frac{\beta_{bad}}{\beta_{good}} \cdot \frac{\gamma_{min}^k}{\gamma_{bad} \gamma_{good}^{k-1}} \\ &> \frac{\beta_{bad}}{\beta_{good}} \cdot \frac{\gamma_{min}^k}{\gamma_{bad}^k} \\ &= \left[ \frac{f(n)f(n+4)}{[f(n+2)]^2} \right]^6 \left[ \frac{f(n+1)f(n+5)}{[f(n+3)]^2} \right]^6 \Big]^k \\ &\quad \cdot \left[ \frac{[f(n+2)]^{10} [f(n+3)]^{10}}{[f(n)]^4 [f(n+1)]^6 [f(n+4)]^6 [f(n+5)]^4} \right]^k \\ &= \left[ \frac{f(n)f(n+5)}{f(n+2)f(n+3)} \right]^{2k} \\ &> 1 \end{aligned}$$

by Lemma 6.3.

Finally

$$\begin{aligned} \frac{h_0}{h_2} &= \frac{\alpha_{bad}}{\alpha_{good}} \frac{\gamma_{min}^k}{\gamma_{bad} \gamma_{good}^{k-1}} \\ &> \frac{\alpha_{bad}}{\alpha_{good}} \cdot \frac{\gamma_{min}^k}{\gamma_{bad}^k} \\ &= \left[ \frac{f(n)f(n+2)}{[f(n+1)]^2} \right]^4 \left[ \frac{f(n+1)f(n+3)}{[f(n+2)]^2} \right]^{14} \\ &\quad \cdot \left[ \frac{f(n+2)f(n+4)}{[f(n+3)]^2} \right]^{14} \left[ \frac{f(n+3)f(n+5)}{[f(n+4)]^2} \right]^4 \Big]^k \\ &\quad \cdot \left[ \frac{[f(n+2)]^{10} [f(n+3)]^{10}}{[f(n)]^4 [f(n+1)]^6 [f(n+4)]^6 [f(n+5)]^4} \right]^k \\ &= 1. \end{aligned}$$

Making each of these changes in the proof of Theorem 6.5, we complete the proof of this theorem.  $\blacksquare$

**Proposition 6.13.** Fix a function  $f(x) : \mathbb{Z}^{\geq 0} \rightarrow \mathbb{R}$  which satisfies the following properties:

- $\log f(x)$  is strictly concave up, and
- for all but finitely many  $n \in \mathbb{Z}$ ,  $n \geq 2$ ,

$$\frac{f(n-2)[f(n+1)]^3[f(n+2)]^3f(n+5)}{f(n-1)[f(n)]^3[f(n+3)]^3f(n+4)} < 1.$$



TABLE 3. The 26 strings to complement the collection in Table 1 along with their Hamming distance from medians in  $\mathcal{M}'$ .

	A	B	C	M1	M2	M3	M4	M5	M6	M7	M8
Row #	Values of $\nu_j^i$ on its support set	$\nu_j^i[x_\ell]$ , $\nu_j^i[y_\ell]$ ( $v_\ell \notin c_i$ )	Add'l Ones	10 10 10	10 10 01	10 01 10	01 10 10	10 01 01	01 10 01	01 01 10	01 01 01
1	010000	0	+0	$n+1$	$n+1$	$n+1$	$n-1$	$n+1$	$n-1$	$n-1$	$n-1$
2	000100	0	+0	$n+1$	$n+1$	$n-1$	$n+1$	$n-1$	$n+1$	$n-1$	$n-1$
3	000001	0	+0	$n+1$	$n-1$	$n+1$	$n+1$	$n-1$	$n-1$	$n+1$	$n-1$
4	101111	1	+3	$n+2$	$n+2$	$n+2$	$n+4$	$n+2$	$n+4$	$n+4$	$n+4$
5	111011	1	+3	$n+2$	$n+2$	$n+4$	$n+2$	$n+4$	$n+2$	$n+4$	$n+4$
6	111110	1	+3	$n+2$	$n+4$	$n+2$	$n+2$	$n+4$	$n+4$	$n+2$	$n+4$
7	010100	0	+2	$n+4$	$n+4$	$n+2$	$n+2$	$n+2$	$n+2$	$n$	$n$
8	010001	0	+2	$n+4$	$n+2$	$n+4$	$n+2$	$n+2$	$n$	$n+2$	$n$
9	000101	0	+2	$n+4$	$n+2$	$n+2$	$n+4$	$n$	$n+2$	$n+2$	$n$
10	101011	1	+1	$n-1$	$n-1$	$n+1$	$n+1$	$n+1$	$n+1$	$n+3$	$n+3$
11	101110	1	+1	$n-1$	$n+1$	$n-1$	$n+1$	$n+1$	$n+3$	$n+1$	$n+3$
12	111010	1	+1	$n-1$	$n+1$	$n+1$	$n-1$	$n+3$	$n+1$	$n+1$	$n+3$
13	100101	0	+1	$n+2$	$n$	$n$	$n+4$	$n-2$	$n+2$	$n+2$	$n$
14	011001	0	+1	$n+2$	$n$	$n+4$	$n$	$n+2$	$n-2$	$n+2$	$n$
15	010110	0	+1	$n+2$	$n+4$	$n$	$n$	$n+2$	$n+2$	$n-2$	$n$
16	101001	0	+1	$n$	$n-2$	$n+2$	$n+2$	$n$	$n$	$n+4$	$n+2$
17	100110	0	+1	$n$	$n+2$	$n-2$	$n+2$	$n$	$n+4$	$n$	$n+2$
18	011010	0	+1	$n$	$n+2$	$n+2$	$n-2$	$n+4$	$n$	$n$	$n+2$
19	101010	0	+1	$n-2$	$n$	$n$	$n$	$n+2$	$n+2$	$n+2$	$n+4$
20	010101	1	+2	$n+5$	$n+3$	$n+3$	$n+3$	$n+1$	$n+1$	$n+1$	$n-1$
21	100101	1	+2	$n+3$	$n+1$	$n+1$	$n+5$	$n-1$	$n+3$	$n+3$	$n+1$
22	011001	1	+2	$n+3$	$n+1$	$n+5$	$n+1$	$n+3$	$n-1$	$n+3$	$n+1$
23	010110	1	+2	$n+3$	$n+5$	$n+1$	$n+1$	$n+3$	$n+3$	$n-1$	$n+1$
24	101001	1	+2	$n+1$	$n-1$	$n+3$	$n+3$	$n+1$	$n+1$	$n+5$	$n+3$
25	100110	1	+2	$n+1$	$n+3$	$n-1$	$n+3$	$n+1$	$n+5$	$n+1$	$n+3$
26	011010	1	+2	$n+1$	$n+3$	$n+3$	$n-1$	$n+5$	$n+1$	$n+1$	$n+3$

The information in this table is exactly the same as the information in Table 1. This is detailed in Section 4.1.

For arbitrary  $m, s \in \mathbb{Z}^{>0}$  and  $D \in \mathbb{R}$ , let  $S := \{\nu_1, \nu_2, \dots, \nu_m\}$  be a multiset of binary strings, each of length  $s$ . Then it is  $\#P$ -hard to determine the number of medians  $\mu$  for  $S$  such that

$$(19) \quad \prod_{i \in [m]} f(H(\nu_i, \mu)) \leq D.$$

*Proof.* This is true by the same logic in the proof of Proposition 6.11 ■

We can also state the following corollaries for functions which are concave down.

**Corollary 6.14.** Fix a function  $f(x) : \mathbb{Z}^{\geq 0} \rightarrow \mathbb{R}$  which satisfies the following properties:

- $\log f(x)$  is strictly concave down,
- the function values can be computed in time polynomial in the input, and
- for all but finitely many  $n \in \mathbb{Z}$ ,  $n \geq 2$ ,

$$\frac{f(n-2)[f(n+1)]^3[f(n+2)]^3f(n+5)}{f(n-1)[f(n)]^3[f(n+3)]^3f(n+4)} \neq 1.$$

For arbitrary  $m, s \in \mathbb{Z}^{>0}$  and  $D \in \mathbb{R}$ , let  $S := \{\nu_1, \nu_2, \dots, \nu_m\}$  be a multiset of binary strings, each of length  $s$ . Then it is NP-hard to determine if there is a median  $\mu$  for  $S$  such that

$$(20) \quad \prod_{i \in [m]} f(H(\nu_i, \mu)) \geq D.$$

*Proof.* If the function  $f(x)$  has the property that  $\log f(x)$  is strictly concave down, then  $\log(f(x))^{-1}$  is strictly concave up. Therefore by Theorems 6.5 and 6.12 for the function  $(f(x))^{-1}$ , it is NP-hard to determine if there is a most parsimonious labeling which admits at most  $\frac{1}{m}$  scenarios. However asking if there is an  $x$  such that  $\frac{1}{f(x)} < \frac{1}{m}$  equivalent to asking if there is an  $x$  such that  $f(x) > m$ . ■

**Corollary 6.15.** Fix a function  $f(x) : \mathbb{Z}^{\geq 0} \rightarrow \mathbb{R}$  which satisfies the following properties:

- $\log f(x)$  is strictly concave down,
- the function values can be computed in time polynomial in the input, and
- for all but finitely many  $n \in \mathbb{Z}$ ,  $n \geq 2$ ,

$$\frac{f(n-2)[f(n+1)]^3[f(n+2)]^3f(n+5)}{f(n-1)[f(n)]^3[f(n+3)]^3f(n+4)} \neq 1.$$

For arbitrary  $m, s \in \mathbb{Z}^{>0}$  and  $D \in \mathbb{R}$ , let  $S := \{\nu_1, \nu_2, \dots, \nu_m\}$  be a multiset of binary strings, each of length  $s$ . Then it is #P-hard to count the number of medians  $\mu$  for  $S$  such that

$$(21) \quad \prod_{i \in [m]} f(H(\nu_i, \mu)) \geq D.$$

*Proof.* This is a corollary to Propositions 6.11 and 6.13. The proof follows the logic in the proof of Corollary 6.14. ■

**Remark 6.16.** These results for  $f(x)$  being a concave down function tell us more about the complexity of #StarSPSCJ( $f$ ) and the existence of an FPRAS. These proofs will be detailed in a later draft.

## 7. SET-UP FOR RESULTS ON BINARY TREES

In this section and the next, we turn our attention to most parsimonious SCJ scenarios on binary trees. We will use the following definition:

**Definition 7.1.** A tree is a binary tree if it is rooted and every non-leaf vertex has exactly two children.

The main result of Section 8 is the theorem which states #BinSPSCJ is #P-complete. Here we develop several tools and algorithms which lay the foundation for our main theorem in the next section. Let  $\Gamma$  be a D3CNF,  $\Gamma = c_1 \wedge c_2 \wedge \dots \wedge c_k$  with variables  $\{v_1, v_2, \dots, v_n\}$ . Let  $\{w_1, w_2, \dots, w_n\}$  be new variables. For each  $i \in [n]$ , calculating subscripts modulo  $n$ , define the following D3CNF,

$$(22) \quad \Phi_i := (v_i \vee w_i \vee v_{i+1}) \wedge (v_i \vee w_i \vee \overline{v_{i+1}}) \wedge (\overline{v_i} \vee \overline{w_i} \vee v_{i+1}) \wedge (\overline{v_i} \vee \overline{w_i} \vee \overline{v_{i+1}}).$$

Observe that  $\Phi_i$  is equivalent to the “exclusive or”  $(v_i \vee w_i) \wedge (\overline{v_i} \vee \overline{w_i})$ .

Define

$$\Psi(\Gamma) := \Gamma \wedge \bigwedge_{i=1}^n \Phi_i.$$

**Lemma 7.2.** For arbitrary D3CNF  $\Gamma$ , it is #P-complete to determine the number of satisfying truth assignments for  $\Psi(\Gamma)$ .

*Proof.* We have already shown in Lemma 3.9 that #D3SAT is in #P-complete. So to prove this result, we will show that the satisfying truth assignments for  $\Gamma$  and for  $\Psi(\Gamma)$  are in one-to-one correspondence.

Any truth assignment which satisfies  $\Psi(\Gamma)$ , when restricted to  $\{v_1, v_2, \dots, v_n\}$  will necessarily satisfy  $\Gamma$ .

For the other direction, recall that  $\Phi_i$  is equivalent to the “exclusive or” for  $v_i$  and  $w_i$ . Therefore, given a satisfying truth assignment for  $\Gamma$ , we can create a satisfying truth assignment for  $\Psi(\Gamma)$  by assigning each  $w_i$  the opposite value of  $v_i$ . ■

In the proof of Theorem 8.1, we will use two different algorithms to help us explore the most parsimonious SCJ scenarios for a rooted binary tree with a given leaf labeling.

Start with a rooted binary tree  $T$ , with root  $\rho$  and a labeling for the leaves  $\varphi : L(T) \rightarrow \{0, 1\}$ . Let  $\varphi' : V(T) \rightarrow \{0, 1\}$  be a most parsimonious labeling which extends  $\varphi$ . Because each vertex is labeled with a single bit and  $\varphi'$  is a most parsimonious labeling,  $\varphi'$  must minimize the number of edges  $uv$  such that  $\varphi'(u) \neq \varphi'(v)$ .

First, we have Fitch’s algorithm to find most parsimonious solutions.

**Algorithm 7.3** (Fitch’s Algorithm [5]). *Let  $T$  be a binary tree with root  $\rho$  and labeling  $\varphi : L(T) \rightarrow \{0, 1\}$ . The following algorithm, completed in two parts, will find a most parsimonious labeling  $\varphi' : V(T) \rightarrow \{0, 1\}$  which extends  $\varphi$ .*

*Part 1: Define a function  $B$  on the vertices of  $T$  as follows: For each leaf  $\ell$ , set  $B(\ell) := \varphi(\ell)$ . Extend this assignment to all vertices by the following rule: Once the children  $v_1, v_2$  of  $u$  have been assigned a set, define*

$$(23) \quad B(u) := \begin{cases} B(v_1) \cap B(v_2) & \text{if } B(v_1) \cap B(v_2) \neq \emptyset, \\ B(v_1) \cup B(v_2) & \text{otherwise.} \end{cases}$$

*Part 2: Select a single element  $\alpha \in B(\rho)$ . Define a function  $\varphi'$  on the vertices of  $T$  as follows: Set  $\varphi'(\rho) := \alpha$ . Extend  $\varphi'$  to all vertices by the following rule: If  $v$  is a child of  $u$  and  $\varphi'(u)$  is defined, then*

$$(24) \quad \varphi'(v) := \begin{cases} \varphi'(u) \cap B(v) & \text{if } \varphi'(u) \cap B(v) \neq \emptyset, \\ B(v) & \text{if } \varphi'(u) \neq B(v). \end{cases}$$

*The resulting  $\varphi'$ , which is a most parsimonious labeling extending  $\varphi$ , is called a Fitch solution.*

While Fitch solutions are most parsimonious solutions, there are times when a most parsimonious solution will not be found using Fitch’s algorithm. However, Sankoff’s algorithm [3, 8], described below, will produce all most parsimonious solutions [3].

**Algorithm 7.4** (Sankoff’s Algorithm [3, 8]). *Let  $T$  be a binary tree with root  $\rho$  and leaf labeling  $\varphi : L(T) \rightarrow \{0, 1\}$ . This algorithm is also completed in two steps.*

*Part 1: Define functions  $s_0$  and  $s_1$  on the vertices of  $T$  as follows: For each leaf  $\ell$ ,*

$$(25) \quad s_0(\ell) := \begin{cases} 0 & \text{if } \varphi(\ell) = 0 \\ \infty & \text{otherwise.} \end{cases}$$

$$s_1(\ell) := \begin{cases} 0 & \text{if } \varphi(\ell) = 1 \\ \infty & \text{otherwise.} \end{cases}$$

*Extend these functions recursively to all vertices by the following: If  $v_0$  and  $v_1$  are children of  $u$ , then*

$$(26) \quad s_0(u) := \min\{s_0(v_0), s_1(v_0) + 1\} + \min\{s_0(v_1), s_1(v_1) + 1\},$$

$$(27) \quad s_1(u) := \min\{s_0(v_0) + 1, s_1(v_0)\} + \min\{s_0(v_1) + 1, s_1(v_1)\}.$$

Note: For any  $v \in V(T)$ ,  $s_i(v)$  counts the minimum number of edges, within the subtree containing  $v$  and its descendants, that will witness a change if  $\varphi'(v) = i$ .

Part 2: For each  $v \in V(T)$ , select  $\alpha_v \in \{0, 1\}$ . Define the function  $\varphi'$  on the vertices of  $T$  as follows: For root  $\rho$ , define

$$\varphi'(\rho) := \begin{cases} 0 & \text{if } s_0(\rho) < s_1(\rho) \\ \alpha_\rho & \text{if } s_0(\rho) = s_1(\rho) \\ 1 & \text{if } s_0(\rho) > s_1(\rho). \end{cases}$$

Extend  $\varphi'$  to  $V(T)$  by the following rule: If  $v$  is a child of  $u$  and  $\varphi'(u)$  is defined, then define  $\varphi'(v)$  as follows: If  $\varphi'(u) = 0$ , then

$$(28) \quad \varphi'(v) := \begin{cases} 0 & \text{if } s_0(v) < s_1(v) + 1 \\ \alpha_v & \text{if } s_0(v) = s_1(v) + 1 \\ 1 & \text{if } s_0(v) > s_1(v) + 1. \end{cases}$$

If  $\varphi'(u) = 1$ , then

$$(29) \quad \varphi'(v) := \begin{cases} 1 & \text{if } s_1(v) < s_0(v) + 1 \\ \alpha_v & \text{if } s_1(v) = s_0(v) + 1 \\ 0 & \text{if } s_1(v) > s_0(v) + 1. \end{cases}$$

The resulting  $\varphi'$ , which is a most parsimonious labeling for  $T$  extending  $\varphi$ , is called a Sankoff solution.

The following lemma draws a connection between the solutions found from each algorithm. Within the proof, we will use  $u_\ell$  and  $u_r$  to denote the children of a vertex  $u \in V(T) \setminus L(T)$ .

**Lemma 7.5.** *Let  $T$  be a rooted binary tree with leaf labeling  $\varphi : L(T) \rightarrow \{0, 1\}$ . Suppose that, for each  $u, v \in V(T)$  with  $v$  a child of  $u$ , Fitch's algorithm has*

$$(30) \quad B(v) = \{0, 1\} \Rightarrow B(u) = \{0, 1\}$$

*Then for  $T$  and  $\varphi$ , all Sankoff solutions are Fitch solutions. In other words, Fitch's algorithm find all most parsimonious labelings.*

*Proof.* Let  $T$  and  $\varphi$  be as described in the lemma. Fix a path  $P$  of vertices

$$v_0, v_1, \dots, v_{k-1}, v_k = \rho,$$

where  $v_0 \in L(T)$  and  $\rho$  the root of  $T$ .

After running Part 1 of Fitch's algorithm, (30) and (23) imply that there exists  $0 < j \leq k$  such that either

$$\begin{aligned} & B(v_0) = B(v_1) = \dots = B(v_j) = \{0\} \text{ and } B(v_{j+1}) = B(v_{j+2}) = \dots = B(v_k) = \{0, 1\} \text{ or} \\ & B(v_0) = B(v_1) = \dots = B(v_j) = \{1\} \text{ and } B(v_{j+1}) = B(v_{j+2}) = \dots = B(v_k) = \{0, 1\} \end{aligned}$$

We restrict our attention to the setting where  $B(v_0) = \{0\}$  and consider the values of  $s_0$  and  $s_1$  for each vertex along the path.

Because  $v_0$  is a leaf,

$$(31) \quad s_0(v_0) = 0 \text{ and } s_1(v_0) = \infty.$$

**Claim 7.6.** *For any non-leaf vertex  $v_i$ , if  $B(v_i) = \{0\}$ , then in Sankoff's algorithm  $s_0(v_i) = 0$  and  $s_1(v_i) = 2$ . Similarly, if  $B(v_i) = \{1\}$ , in Sankoff's algorithm  $s_0(v_i) = 2$  and  $s_1(v_i) = 0$*

*Proof.* This proof proceeds by induction on the distance from the root where the base case examines non-leaf vertices which are farthest from the root.

For the base case, consider  $v_1$ , a non-leaf vertex which is farthest from the root. Both children of  $v_1$  must be leaves. Since  $B(v_1) = \{0\}$ , we can conclude  $B(v_0) = B(v'_0) = \{0\}$ . Consequently

$\varphi(v_0) = \varphi(v'_0) = 0$  and by (25),  $s_0(v_0) = s_0(v'_0) = 0$  and  $s_1(v_0) = s_1(v'_0) = \infty$ . As desired, (26) implies  $s_0(v_1) = 0$  and (27) implies  $s_1(v_1) = 2$ .

For induction, assume that for each vertex  $v_i$  of distance at least  $d$  from the root has  $s_0(v_i) = 0$  and  $s_1(v_i) = 2$ . Let  $v_{i+1}$  be a vertex of distance  $d - 1$  from the root. This vertex has two children,  $v_i$  and  $v'_i$ . There are three cases to consider.

- (1) If  $v_i$  and  $v'_i$  are leaves, then the argument in the base case gives  $s_0(v_{i+1}) = 0$  and  $s_1(v_{i+1}) = 2$  as desired.
- (2) If  $v'_i$  is a leaf and  $v_i$  is not a leaf, then  $s_0(v'_i) = 0$  and  $s_1(v'_i) = \infty$  and, by the inductive hypothesis  $s_0(v_i) = 0$  and  $s_1(v_i) = 2$ . Therefore (26) implies  $s_0(v_{i+1}) = 0$  and (27) implies  $s_1(v_{i+1}) = 2$ .
- (3) If  $v_i$  and  $v'_i$  are not leaves, then by the inductive hypothesis,  $s_0(v_i) = s_0(v'_i) = 0$  and  $s_1(v_i) = s_1(v'_i) = 2$ . Again, (26) implies  $s_0(v_{i+1}) = 0$  and (27) implies  $s_1(v_{i+1}) = 2$ .

This completes the proof of the claim. ■

We establish one more claim.

**Claim 7.7.** *For any vertex  $v_i$  with  $B(v_i) = \{0, 1\}$  from Fitch's algorithm, we will have  $s_0(v_i) = s_1(v_i)$  in Sankoff's algorithm.*

*Proof.* This claim is also proven by induction on distance from the root where the base case examines those vertices with greatest distance from the root.

For the base case, let  $v_{j+1}$  be a vertex with children  $v_j$  and  $v'_j$  such that  $B(v_{j+1}) = \{0, 1\}$  and  $B(v_j) = \{0\}$ . By (23),  $B(v'_j) = \{1\}$ . By Claim 7.6,

$$s_0(v_j) = s_1(v'_j) = 0 \text{ and } s_0(v'_j) = s_1(v_j) = 2.$$

Therefore  $s_0(v_{j+1}) = s_1(v_{j+1}) = 1$ .

For the inductive hypothesis, suppose all vertices  $v_i$  with  $B(v_i) = \{0, 1\}$  of distance at least  $d \geq 1$  from the root have  $s_0(v_i) = s_1(v_i)$ . Let  $v_{i+1}$  be a vertex at distance  $d - 1$  from the root with  $B(v_{i+1}) = \{0, 1\}$ . There are three cases to consider:

- (1) If  $v_{i+1}$  has a child  $v_i$  with  $B(v_i) = \{0\}$ , then the argument in the base case gives  $s_0(v_j) = s_1(v'_j)$ .
- (2) If  $v_{i+1}$  has a child  $v_i$  with  $B(v_i) = \{1\}$ , then by (23),  $v_{i+1}$  must have another child  $v'_i$  with  $B(v'_i) = \{0\}$ . This puts us back in case 1.
- (3) If  $v_{i+1}$  has a child  $v_i$  with  $B(v_i) = \{0, 1\}$ , then by (23),  $v_{i+1}$  must have another child  $v'_i$  with  $B(v'_i) = \{0, 1\}$ . By the inductive hypothesis,  $s_0(v_i) = s_1(v_i)$  and  $s_0(v'_i) = s_1(v'_i)$ . By (26) and (27),  $s_0(v_{i+1}) = s_1(v_{i+1})$ .

This completes the proof of the claim. ■

Now that we have a connection between the function  $B$  and the functions  $s_0$  and  $s_1$ , let us compare Part 2 of each algorithm where the most parsimonious labeling  $\varphi'$  is defined.

**Claim 7.8.** *For any non-leaf vertex  $v$  with  $B(v) = \{0\}$ , both Fitch's algorithm and Sankoff's algorithm will define  $\varphi'(v) = 0$ . Similarly, if  $B(v) = \{1\}$ , both Fitch's algorithm and Sankoff's algorithm will define  $\varphi'(v) = 1$ .*

*Proof.* In Fitch's algorithm, this is an immediate consequence of (24) because  $\varphi'(v) \in B(v)$ .

Now consider Sankoff's algorithm. From Claim 7.6 recall that  $s_0(v) = 0$  and  $s_1(v) = 2$ . Observe  $s_0(v) < s_1(v) + 1$  and  $s_1(v) > s_0(v) + 1$ . Therefore  $\varphi'(v) = 0$  by (28) and (29). ■

If  $B(\rho) = \{0\}$  or  $B(\rho) = \{1\}$ , then Claim 7.8 completes the proof. If  $B(\rho) = \{0, 1\}$ , then in Fitch's algorithm, there are two choices for  $\varphi'(\rho)$ . By Claim 7.7,  $s_0(\rho) = s_1(\rho)$  in Sankoff's algorithm, which means there are also two choices for  $\varphi'(\rho)$ . This last claim implies that if we make the same choice for the root, both algorithms give the same most parsimonious labeling  $\varphi'$ .

**Claim 7.9.** *Suppose  $B(\rho) = \{0, 1\}$ . For any vertex  $v$  with  $B(v) = \{0, 1\}$ , both Fitch's algorithm and Sankoff's algorithm will define  $\varphi'(v) := 0$ . Likewise, if  $\varphi'(\rho) := 1$ , then  $\varphi'(v) := 1$ .*

*Proof.* For any vertex  $v$  with  $B(v) = \{0, 1\}$ , there is a path  $\rho = u_0, u_1, u_2, \dots, u_{t-1}, u_t = v$  of vertices such that  $B(u_i) = \{0, 1\}$  for each  $i \in [t]$ . It suffices to show that, in both algorithms, if  $\varphi'(u_i) = 0$  for some  $0 \leq i < t$ , then  $\varphi'(u_{i+1}) = 0$ .

In Part 2 of Fitch's algorithm, if  $\varphi'(u_i) = 0$  and  $B(u_{i+1}) = \{0, 1\}$ , then  $\varphi'(u_{i+1}) = 0$  by (23).

In Sankoff's algorithm, if  $\varphi'(u_i) = 0$  and  $B(u_{i+1}) = \{0, 1\}$ , then by Claim 7.9,  $s_0(u_{i+1}) = s_1(u_{i+1})$ . Thus  $s_0(u_{i+1}) < s_1(u_{i+1}) + 1$ . Since  $\varphi'(u_i) = 0$ , (28) implies  $\varphi'(u_{i+1}) = 0$ .

A similar argument can be used to show if  $\varphi'(\rho) := 1$ , then  $\varphi'(v) := 1$ . Therefore Fitch's algorithm and Sankoff's algorithm will agreed on  $\varphi'(v)$  if they agree on  $\varphi'(\rho)$ . ■

Because Sankoff's algorithm is guaranteed to find all most parsimonious labelings and Fitch's algorithm finds the same most parsimonious labelings as Sankoff's algorithm, this implies that Fitch's algorithm finds all most parsimonious labelings. ■

For a tree  $T$  whose leaves are labeled with binary strings in  $\{0, 1\}^N$ , restrict all strings to a single coordinate and run one of the above algorithms to find a most parsimonious labelings for  $V(T)$ . Repeat this for each coordinate. The most parsimonious solutions found for each coordinate can then be combined into a most parsimonious labeling for the original leaf labeling of  $T$ .

## 8. A COMPLEXITY RESULT FOR THE BINARY TREE

Here is our main result on binary trees.

**Theorem 8.1.** *#BinSPSCJ is #P-complete.*

*Proof.* In Lemma 3.13 we saw that  $\#SPSCJ \in \#P$ . To prove  $\#SPSCJ \in \#P$ -complete, we provide a polynomial reduction from  $\#D3CNF$ .

Fix a D3CNF,  $\Gamma = \bigwedge_{i \in [k]} c_i$  with  $k$  clauses and  $n$  variables. Let

$$\Psi(\Gamma) := \bigwedge_{i \in [k]} c_i \wedge \bigwedge_{i \in [n]} \Phi_i$$

with  $2n$  variables,  $\{v_1, v_2, \dots, v_n, w_1, w_2, \dots, w_n\}$ , where each clause  $c_i$  has three distinct literals from  $\{v_i, \bar{v}_i : i \in [n]\}$ , and  $\Phi_i$  is the D3CNF in (22) which guarantees that  $v_i$  and  $w_i$  have different truth values for each  $i \in [n]$ . By Lemma 7.2,  $\Gamma$  and  $\Psi(\Gamma)$  have the same satisfying truth assignments. Given  $\Gamma$ , we will construct a binary tree  $\mathcal{B}$  and define a labeling  $\varphi$  of its leaves such that the number of most parsimonious SCJ scenarios is equal to the number of satisfying truth assignment for  $\Psi(\Gamma)$ .

Each  $\Phi_i$  has 4 clauses, so  $\Psi(\Gamma)$  has  $k + 4n$  clauses. Assign the names  $c_{k+1}, \dots, c_{k+4n}$  to the  $4n$  clauses of  $\bigwedge_{i \in [n]} \Phi_i$ . Then

$$\Psi(\Gamma) = \bigwedge_{i \in [k+4n]} c_i$$

where  $\Gamma = \bigwedge_{i \in [k]} c_i$ .

For each  $i \in [k + 4n]$ , we define a binary tree  $\mathcal{B}_i$  which encodes clause  $c_i$ . The final binary tree  $\mathcal{B}$  will join  $\mathcal{B}_1, \mathcal{B}_2, \dots, \mathcal{B}_{k+4n}$  by a comb. The labeling  $\varphi : L(\mathcal{B}) \rightarrow \{0, 1\}^{2n+t}$ , for  $t = 148(16n^2 + 8kn)(k + 4n)$ , will assign a binary string on coordinates  $(x_1, y_1, \dots, x_n, y_n, e_1, \dots, e_t)$  to each leaf. The  $x_i$  coordinates will correspond to the  $v_i$  variables and the  $y_i$  coordinates will correspond to the  $w_i$  variables. The  $e_i$  coordinates will be for additional ones, similar to those in the star trees of the last chapter.

In our definitions, for each non-leaf vertex  $v$ , we denote the left child of  $v$  by  $v_\ell$  and the right child of  $v$  by  $v_r$ . The height of a vertex is the distance it is from the root. The construction of  $\mathcal{B}_i$  and its leaf labeling will come in Definition 8.6, but first we need some definitions.

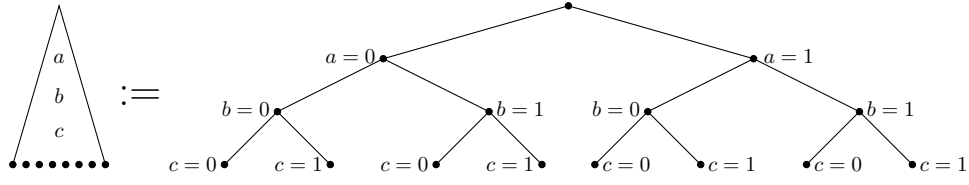


FIGURE 1. The labeled binary tree on the right is  $S(a, b, c)$ . The representation on the left will be used in place of  $S(a, b, c)$  in future figures.

**Definition 8.2.** For three literals  $a, b, c$ , we define  $S(a, b, c)$  to be the complete binary tree of height 3 with root  $\rho$  and vertex labels as follows:

- (1) Assign the label  $a = 0$  to vertex  $\rho_\ell$  and  $a = 1$  to  $\rho_r$ .
- (2) For each vertex  $u$  of height 2, assign the label  $b = 0$  to  $u_\ell$  and  $b = 1$  to  $u_r$ .
- (3) For each vertex  $v$  of height 3, assign the label  $c = 0$  to  $v_\ell$  and  $c = 1$  to  $v_r$ .

This tree is pictured on the right in Figure 1. We will use the representation on the left in place of  $S(a, b, c)$  in future figures.

Next we construct  $\hat{\mathcal{B}}_i$  which will have the same structure as  $\mathcal{B}_i$ . However,  $\hat{\mathcal{B}}_i$  will have equations labeling all of the vertices. The leaf labeling of  $\mathcal{B}_i$  will be induced by the vertex labels of  $\hat{\mathcal{B}}_i$ . Each leaf will essentially inherit the labels of its ancestors.

**Definition 8.3.** Fix  $i \in [k]$ . The clause  $c_i$  has variables  $v_\alpha, v_\beta, v_\gamma$ . Construct  $\hat{\mathcal{B}}_i$ , a binary tree with vertex labels, as follows.

- (1) Draw a vertex  $\rho^i$  with two children,  $\rho_\ell^i$  and  $\rho_r^i$ .
- (2) Label  $\rho_\ell^i$  with  $x_j = y_j = 0$  for all  $j \in [n] \setminus \{\alpha, \beta, \gamma\}$ . Label  $\rho_r^i$  with  $x_j = y_j = 1$  for all  $j \in [n] \setminus \{\alpha, \beta, \gamma\}$ .
- (3) From each of  $\rho_\ell^i$  and  $\rho_r^i$ , hang a copy of  $S(y_\alpha, y_\beta, y_\gamma)$ .
- (4) From each leaf of each copy of  $S(y_\alpha, y_\beta, y_\gamma)$ , hang a copy of  $S(x_\alpha, x_\beta, x_\gamma)$ .
- (5) Delete the left-most copy of  $S(x_\alpha, x_\beta, x_\gamma)$  and replace it with a copy of  $\mathcal{T}_i$  which will be explained in Definition 8.5.

This is drawn in Figure 8.

The construction of  $\mathcal{B}_i$  for  $i \in \{k+1, \dots, k+4n\}$  is very similar to the construction in Definition 8.3. We just need a few changes of variables.

**Definition 8.4.** Fix  $i \in \{k+1, \dots, k+4n\}$ . The clause  $c_i$  relates variables  $v_i, w_i, v_{i+1}$ . Construct  $\hat{\mathcal{B}}_i$ , a binary tree with vertex labels, as follows:

- (1) Draw a vertex  $\rho^i$  with two children,  $\rho_\ell^i$  and  $\rho_r^i$ .
- (2) Label  $\rho_\ell^i$  with  $x_j = y_j = 0$  for all  $j \in [n] \setminus \{i, i+1, i+2\}$ . Label  $\rho_r^i$  with  $x_j = y_j = 1$  for all  $j \in [n] \setminus \{i, i+1, i+2\}$ .
- (3) From each of  $\rho_\ell^i$  and  $\rho_r^i$ , hang a copy of  $S(y_{i+1}, x_{i+2}, y_{i+2})$ .
- (4) From each leaf vertex of each copy of  $S(y_{i+1}, x_{i+2}, y_{i+2})$ , hang a copy of  $S(x_i, y_i, x_{i+1})$ .
- (5) Delete the left-most copy of  $S(x_i, y_i, x_{i+1})$  and replace it with a copy of  $\mathcal{T}_i$  which will be explained in Definition 8.5.

For any clause which is the disjunction of 3 distinct literals, Miklós, Kiss, and Tannier [6] defined a *unit subtree*, with 248 leaves. They also gave leaf labeling  $\hat{\varphi}$  for this unit subtree which assigns a binary string to each leaf with 3 coordinates corresponding to the variables in the clause and 148 coordinates for additional ones. This unit subtree has some useful properties which will be discussed after Definition 8.6.

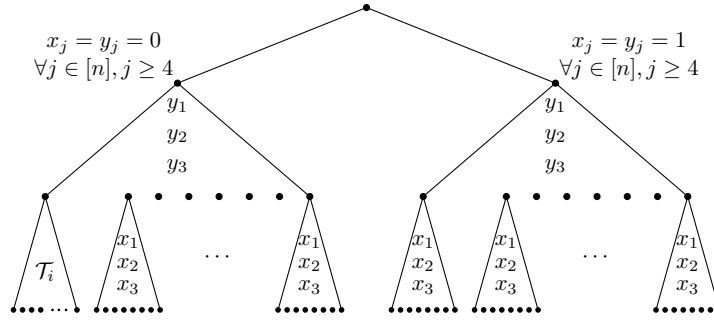


FIGURE 2. The binary tree  $\hat{\mathcal{B}}_i$ , for  $i \in [k]$  created for clause  $c_i = x_1 \vee x_2 \vee x_3$ .

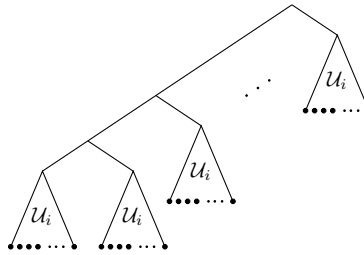


FIGURE 3. Comb connecting  $16n^2 + 8kn$  copies of  $\mathcal{U}_i$  to create  $\mathcal{T}_i$

For each  $i \in [k + 4n]$ , let  $\mathcal{U}_i$  be the unit subtree for clause  $c_i$ . If  $i \leq k$  and  $c_i$  relates  $v_\alpha, v_\beta, v_\gamma$ , then  $\mathcal{U}_i$  will have leaf labels with coordinates  $\{x_\alpha, x_\beta, x_\gamma\}$  and 148 coordinates for additional ones. If  $i > k$ ,  $\mathcal{U}_i$  will have leaf labels with coordinates  $\{x_i, y_i, x_{i+1}\}$  and 148 coordinates for additional ones.

**Definition 8.5.** The tree  $\mathcal{T}_i$  in Step 5 of Definition 8.3 is a comb joining  $16n^2 + 8kn$  copies of  $\mathcal{U}_i$ , as in Figure 3.

Recall  $\mathcal{B}$  is a comb connecting binary trees  $\mathcal{B}_i$ . The binary tree  $\mathcal{B}$  also has a leaf labeling  $\varphi : L(\mathcal{B}) \rightarrow \{0, 1\}^{2n+t}$  where  $t = 148(16n^2 + 8kn)(k + 4n)$ . Each leaf label will have coordinates  $(x_1, y_1, \dots, x_n, y_n, e_1, \dots, e_t)$ . In the next definition, we define  $\mathcal{B}_i$  and values of  $\varphi$  on the leaves of  $\mathcal{B}_i$ .

**Definition 8.6.** For each  $i \in [k + 4n]$ , the binary tree  $\mathcal{B}_i$  will have the same structure as  $\hat{\mathcal{B}}_i$ . We just need to explain the labeling  $\varphi : L(\mathcal{B}_i) \rightarrow \{0, 1\}^{2n+t}$

Partition  $[t]$  into classes  $E_{ij}$  with  $|E_{ij}| = 148$  for each  $i \in [k + 4n]$  and  $j \in [16n^2 + 8kn]$ . Identify the set  $E_{ij}$  with the 148 coordinates for additional ones in  $\hat{\varphi}$  in the  $j^{\text{th}}$  copy of  $\mathcal{U}_i$  in  $\mathcal{T}_i$ .

There are two cases:

- If leaf  $\ell$  is not in subtree  $\mathcal{T}_i$ , then  $\varphi(\ell)[e_s] = 0$  for all  $s \in [t]$ . The value of each  $\varphi(\ell)[x_w]$  and  $\varphi(\ell)[y_w]$  for  $w \in [n]$  is inherited from the labels of the ancestors of  $\ell$  as they appeared in  $\hat{\mathcal{B}}_i$ .
- If leaf  $\ell$  is in  $\mathcal{T}_i$ , then it is a leaf within the  $j^{\text{th}}$  copy of  $\mathcal{U}_i$  for some  $j \in [16n^2 + 8kn]$ . We require that  $\varphi(\ell)$  agree with  $\hat{\varphi}(\ell)$  on the 3 coordinates for the variables in  $c_i$  as well as the



148 coordinates  $e_s$  for  $s \in E_{ij}$ . Set  $\varphi(\ell)[e_s] = 0$  for  $s \notin E_{ij}$ . All other coordinates will take the value 0 because these are the values inherited from the ancestors of  $\ell$ .

Define  $\varphi_{x_j} : L(\mathcal{B}) \rightarrow \{0, 1\}$  so that  $\varphi_{x_j}(\ell) = \varphi(\ell)[x_j]$ . Define  $\varphi_{y_j}$  and  $\varphi_{e_j}$  similarly. We want to examine the Fitch solutions on  $\mathcal{B}$  for each  $\varphi_{x_j}$ ,  $\varphi_{y_j}$ , and  $\varphi_{e_j}$ . Ultimately, we will prove that Fitch's algorithm find all most parsimonious labelings for  $\varphi$  on  $\mathcal{B}$ .

For arbitrary  $j \in [t]$ , we first explore the Fitch solutions for  $\varphi_{e_j}$ .

**Fact 8.7.** *Fix  $j \in [t]$ . There is only one  $\ell \in L(\mathcal{B})$  with  $\varphi_{e_j}(\ell) = 1$ . After running Part 1 of Fitch's algorithm,  $B(\ell) = \{1\}$ ,  $B(v) = \{0, 1\}$  for  $v$  the parent of  $\ell$ , and  $B(u) = \{0\}$  for all other vertices. Consequently, Part 2 of Fitch's algorithm will output a most parsimonious labeling  $\varphi'_{e_j}$  such that  $\varphi'_{e_j}(\ell) = 1$  and for all other  $u \in V(\mathcal{B})$ ,  $\varphi'_{e_j}(u) = 0$ .*

*Proof.* This values of  $B$  follow directly from the description of  $\varphi(\ell)[e_j]$ , for leaf  $\ell$ , which was given in Definition 8.6. The conclusion follows from (24).  $\blacksquare$

**Fact 8.8.** *For  $j \in [t]$ , there is only one most parsimonious labeling  $\varphi'_{e_j}$  which extends leaf labeling  $\varphi_{e_j}$  of  $\mathcal{B}$ .*

*Proof.* The most parsimonious labeling obtained from Fitch's algorithm has

$$\sum_{uv \in E(\mathcal{B})} H(\varphi'_{e_j}(u), \varphi'_{e_j}(v)) = 1.$$

Because there is only one leaf  $\ell$  with  $\varphi_{e_j}(\ell) = 1$ ,  $\varphi'_{e_j}$  obtained from Fitch's algorithm is the only extension of  $\varphi_{e_j}$  with the sum of Hamming distances equal 1.  $\blacksquare$

Fix  $j \in [n]$ . Next we consider the most parsimonious labelings for  $\varphi_{x_j}$  on  $\mathcal{B}$ . The same arguments will hold for each  $\varphi_{y_j}$ .

Run Part 1 of Fitch's algorithm on  $\mathcal{B}$  with leaf labeling  $\varphi_{x_j}$  leaf labeling of  $\mathcal{B}$ . For those clauses  $c_i$  which contain variable  $v_j$ , we have the following result:

**Proposition 8.9** (Miklós, Kiss, and Tannier [6]). *Fix a clause  $c_i$ . Suppose variable  $v_j$  is in  $c_i$ . Let  $r^i$  be the root of unit subtree  $\mathcal{U}_i$  for  $c_i$ . Run Fitch's algorithm on  $\mathcal{U}_i$  with leaf labeling  $\varphi_{x_j}$ . The following hold:*

- (1)  $B(r^i) = \{0, 1\}$ .
- (2) If  $v$  is a child of  $u$ , then  $B(v) = \{0, 1\} \Rightarrow B(u) = \{0, 1\}$ .

With this result and the structure of each  $S(a, b, c)$ , it is straightforward to see that in  $\mathcal{B}$  with leaf labeling  $\varphi_{x_j}$ , if  $v$  is a child of  $u$ ,

$$B(v) = \{0, 1\} \Rightarrow B(u) = \{0, 1\}.$$

By Lemma 7.5, we can conclude that Fitch's algorithm finds all most parsimonious labelings of  $\mathcal{B}$  that extend  $\varphi_{x_j}$ . Further, there are exactly two such most parsimonious labelings because  $B(\rho) = \{0, 1\}$ .

As mentioned earlier, these results also hold for coordinate  $y_i$ . Fitch's algorithm finds the only two most parsimonious labelings that extend  $\varphi_{y_j}$  on  $\mathcal{B}$ .

Define  $\varphi'_{x_j}$  to be the restriction of  $\varphi'$  to coordinate  $x_j$ , similar to the definition of  $\varphi_{x_j}$ . Likewise, define  $\varphi'_{y_j}$  and  $\varphi'_{e_s}$ .

**Lemma 8.10.** *For leaf labeling  $\varphi$  of  $\mathcal{B}$ , Fitch's algorithm finds all most parsimonious labelings. There are exactly  $2^{2n}$  of them and each is characterized by the string it assigns to  $\rho$ . For each most parsimonious labeling  $\varphi'$ ,  $\varphi'(\rho) \in \{0, 1\}^{2n} \times \{0\}^t$ .*

*Proof.* Given a most parsimonious labeling  $\varphi'$  that extends  $\varphi$ , each  $\varphi'_{x_j}$ ,  $\varphi'_{y_j}$ , and  $\varphi'_{e_s}$  is a most parsimonious labeling for that coordinate. So it suffices to first find all most parsimonious scenarios for the leaf labelings  $\varphi_{x_j}, \varphi_{y_j}, \varphi_{e_s}$  for all  $j \in [n]$  and  $s \in [t]$  and take combinations of these labelings.

We have already seen that Fitch's algorithm will find all most parsimonious labelings for  $\varphi_{x_j}$  and  $\varphi_{y_j}$ , and there are exactly 2 of each. Fitch's algorithm will also find the one and only most parsimonious labeling for  $\varphi_{e_s}$ . Therefore, there are  $2^{2n}$  most parsimonious labelings of  $\mathcal{B}$  that extend  $\varphi$ . Part 2 of Fitch's algorithm shows that each most parsimonious labeling is characterized by the string it assigns to  $\rho$ . Since  $B(\rho) = \{0, 1\}$  for each  $\varphi_{x_j}$  and  $\varphi_{y_j}$  and  $B(\rho) = \{0\}$  for each  $\varphi_{e_s}$ , the possible strings for  $\varphi'(\rho)$  are  $\{0, 1\}^{2n} \times \{0\}^t$ .  $\blacksquare$

Let  $\mathcal{M} := \{0, 1\}^{2n} \times \{0\}^t$ .

**Definition 8.11.** Here we create a bijection between  $\mathcal{M}$  and the possible truth assignments for  $\Psi(\Gamma)$ . In particular, given any  $\mu \in \mathcal{M}$ , define a truth assignment for variables  $\{v_i\}_{i=1}^n \cup \{w_i\}_{i=1}^n$  as follows:

- for each  $i \in [n]$ , let  $v_i$  be assigned the value true if  $\mu[x_i] = 1$  and false otherwise
- for each  $i \in [n]$ , let  $w_i$  be assigned the value true if  $\mu[y_i] = 1$  and false otherwise

Define  $\mathcal{M}'_{\Psi(\Gamma)}$  to be the set of  $\mu \in \mathcal{M}$  which correspond to satisfying truth assignments for  $\Psi(\Gamma)$ . Likewise, for any  $\Theta$ , a clause or conjunction of clauses from  $\Psi(\Gamma)$ , define  $\mathcal{M}'_{\Theta}$  to be the set of  $\mu \in \mathcal{M}$  which correspond to satisfying truth assignments for  $\Theta$ .

Now we know the most parsimonious labelings of  $\mathcal{B}$  extending  $\varphi$  are found using Fitch's algorithm and are characterized the binary string they assigned to the root. From here, we are interested in the number of SCJ scenarios admitted by each of these most parsimonious scenarios.

Let  $\varphi'$  be a most parsimonious labeling for  $\mathcal{B}$ . The number of scenarios which are admitted by  $\varphi'$  is precisely

$$\mathcal{H}(\varphi'(\rho)) := \prod_{uv \in E(\mathcal{B})} H(\varphi'(u), \varphi'(v))!$$

To calculate this, we partition the edges of  $\mathcal{B}$  into 4 sets.

Consider the edges of the comb which connects  $\{\mathcal{B}_i\}_{i=1}^{k+4n}$ . Part 2 of Fitch's algorithm will set  $\varphi'(\rho) = \varphi'(\rho^i)$  where  $\rho^i$  is the root of  $\mathcal{B}_i$ . So the Hamming distance along each of these edges is 0.

Next we look within each  $\mathcal{B}_i$ . Fix  $i \in [k + 4n]$ . Let  $\rho^i$  be the root of  $\mathcal{B}_i$  with children  $\rho_\ell^i$  and  $\rho_r^i$ . Set  $\eta := \varphi'(\rho^i)$ . We are now interested in the Hamming distances  $H(\eta, \varphi'(\rho_\ell^i))$  and  $H(\eta, \varphi'(\rho_r^i))$ . Suppose  $\eta \in \mathcal{M}_{\wedge \Phi_i}$ . Then for each coordinate  $x$  considered in Step 2 of Definition 8.3 and 8.4, if  $\eta[x_i] = 0$  then  $\eta[y_i] = 1$  and  $\eta[x_i]$  will add contribute 1 to  $H(\eta, \varphi'(\rho_r^i))$  while  $\eta[y_i] = 1$  will contribute 1 to  $H(\eta, \varphi'(\rho_\ell^i))$ . On the other hand, if  $\eta[x_i] = 1$  then  $\eta[y_i] = 0$  and  $\eta[x_i]$  will add contribute 1 to  $H(\eta, \varphi'(\rho_\ell^i))$  while  $\eta[y_i] = 0$  will contribute 1 to  $H(\eta, \varphi'(\rho_r^i))$ . Thus

$$H(\eta, \varphi'(\rho_\ell^i)) = H(\eta, \varphi'(\rho_r^i)) = n - 3.$$

If  $\eta \notin \mathcal{M}_{\wedge \Phi_i}$ , then we only know that  $H(\eta, \varphi'(\rho_\ell^i)) + H(\eta, \varphi'(\rho_r^i)) = 2n - 6$  because each  $x_i$  and each  $y_i$  will contribute 1 to one of the Hamming distances. Therefore

$$(n - 3)!^2 \leq H(\eta, \varphi'(\rho_\ell^i))! \cdot H(\eta, \varphi'(\rho_r^i))! \leq (2n - 6)!0!$$

Based on the construction of  $S(a, b, c)$ , the Hamming distance  $H(\varphi'(u), \varphi'(v))$  for each edge  $uv$  in each copy of  $S(a, b, c)$  is exactly 1.

The only piece remaining is  $\mathcal{T}_i$ . Say variable  $v_j$  appears in clause  $c_i$ . Run Fitch's algorithm on leaf labeling  $\varphi_{x_j}$  of  $\mathcal{B}$ . By Proposition 8.9, if  $r^i$  is the root of  $\mathcal{U}_i$ , then  $B(r^i) = \{0, 1\}$ . Since all ancestors  $u$  of  $\rho_{\mathcal{U}_i}$  also have  $B(u) = \{0, 1\}$ , Part 2 of Fitch's algorithm assigns  $\varphi'_{x_j}(r^i) = \varphi'_{x_j}(\rho)$ . This is true for all variables that appear in clause  $c_i$ .

Miklós, Kiss, and Tannier [6] assigned leaf labeling  $\hat{\varphi}$  to the leaves of  $\mathcal{U}_i$ . Observe that for each leaf  $u$  in  $\mathcal{U}_i$ ,  $\varphi(u)$  and  $\hat{\varphi}(u)$  agree on the coordinates of  $\hat{\varphi}(u)$ . Let  $\varphi'$  be a most parsimonious labeling extending  $\varphi$ . There is a most parsimonious labeling  $\hat{\varphi}'$  such that for each vertex  $u$  in  $\mathcal{U}_i$  such that  $\varphi'(u)$  and  $\hat{\varphi}'(u)$  agree on the coordinates of  $\varphi'(u)$ . More specifically, for  $r^i$  being the root of  $\mathcal{U}_i$ , if  $\hat{\varphi}'(r^i)$  and  $\varphi'(r^i)$  agree on the coordinates of  $\hat{\varphi}'(r^i)$ , then Fitch's algorithm guarantees that they agree on all of the vertices of  $\mathcal{U}_i$ .

**Claim 8.12.** *Let  $r^i$  be the root of  $\mathcal{U}_i$ . If  $\varphi'(r^i) = \hat{\varphi}'(r^i)$ , then for each  $uv \in E(\mathcal{U}_i)$ ,*

$$H(\varphi'(u), \varphi'(v)) = H(\hat{\varphi}'(u), \hat{\varphi}'(v)).$$

*Proof.* If  $v_j$  is not in  $c_i$ , then Fitch's algorithm will set  $\varphi_{x_j}(u) = 0$  for every vertex  $u$  in  $\mathcal{T}_i$  and consequently each copy of  $\mathcal{U}_i$ . Likewise, if  $w_j$  is not in  $c_i$ , then  $\varphi'_{y_j}(u) = 0$  for every vertex in  $u$ . Within the  $j^{\text{th}}$  copy of  $\mathcal{U}_i$  in  $\mathcal{T}_i$ , for  $e_s \notin E_{ij}$ ,  $\varphi'_{e_s}(u) = 0$  for all vertices  $u$  in the  $j^{\text{th}}$  copy of  $\mathcal{U}_i$ . ■

As a result

$$\prod_{uv \in \mathcal{U}_i} H(\varphi'(u), \varphi'(v))! = \prod_{uv \in \mathcal{U}_i} H(\hat{\varphi}'(u), \hat{\varphi}'(v))!.$$

This is calculated as follows:

**Fact 8.13** (Miklós, Kiss, and Tannier [6]). *For  $i \in [k + 4n]$ , the binary tree  $\mathcal{U}_i$  with root  $r^i$  and leaf labeling  $\hat{\varphi}$ ,*

(1) *If  $\hat{\varphi}'(r^i)$  corresponds to a satisfying truth assignment for  $c_i$ , then*

$$\prod_{uv \in \mathcal{U}_i} H(\hat{\varphi}'(u), \hat{\varphi}'(v))! = 2^{156} \times 3^{64}.$$

(2) *If  $\hat{\varphi}'(r^i)$  corresponds to a truth assignment which does not satisfy  $c_i$ , then*

$$\prod_{uv \in \mathcal{U}_i} H(\hat{\varphi}'(u), \hat{\varphi}'(v))! = 2^{136} \times 3^{76}.$$

Since  $\varphi'(\rho) = \varphi'(r^i) = \hat{\varphi}'(r^i)$ ,  $\varphi'(\rho)$  corresponds to a satisfying truth assignment for  $c_i$  if and only if  $\hat{\varphi}'(r^i)$  also corresponds to a satisfying truth assignment for  $c_i$ .

As a result of the above discussion, we have proven the following claim.

**Claim 8.14.** *Fix  $i \in [k + 4n]$ . If  $\varphi'(\rho)$  corresponds to a satisfying truth assignment for clause  $c_i$  and  $\bigwedge_{i \in [n]} \Phi_i$ , then*

$$\prod_{uv \in E(\mathcal{B}_i)} H(\varphi'(u), \varphi'(v))! = (n - 3)!^2 (2^{156} \times 3^{64})^{16n^2 + 8kn}.$$

*If  $\varphi'(\rho)$  corresponds to a truth assignment which does not satisfy  $c_i$ , then*

$$(n - 3)!^2 (2^{136} \times 3^{76})^{16n^2 + 8kn} \leq \prod_{uv \in E(\mathcal{B}_i)} H(\varphi'(u), \varphi'(v))! \leq (2n - 6)! (2^{136} \times 3^{76})^{16n^2 + 8kn}.$$

*If  $\varphi'(\rho)$  corresponds to a truth assignment which satisfies  $c_i$  but does not satisfy  $\bigwedge_{i \in [n]} \Phi_i$ , then*

$$(n - 3)!^2 (2^{156} \times 3^{64})^{16n^2 + 8kn} < \prod_{uv \in E(\mathcal{B}_i)} H(\varphi'(u), \varphi'(v))! \leq (2n - 6)! (2^{156} \times 3^{64})^{16n^2 + 8kn}.$$

Observe,

$$\begin{aligned}
 \frac{(2n-6)! [2^{136} \times 3^{76}]^{16n^2+8kn}}{(n-3)!^2 [2^{156} \times 3^{64}]^{16n^2+8kn}} &= \left[ \frac{3^{12}}{2^{20}} \right]^{16n^2+8kn} \binom{2n-6}{n-3} \\
 &< \left[ \frac{3^{12}}{2^{20}} \right]^{16n^2+8kn} 2^{2n} \\
 &< \left[ \frac{3^{12}}{2^{20}} \right]^{16n^2+8kn} 2^{2n+k} \\
 &= \left[ \frac{3^{12}}{2^{20-1/(8n)}} \right]^{16n^2+8kn} \\
 &< \left[ \frac{3^{12}}{2^{19.5}} \right]^{16n^2+8kn} \\
 &< 1
 \end{aligned}$$

Consequently,

$$\begin{aligned}
 (2n-6)! (2^{136} \times 3^{76})^{16n^2+8kn} &\leq (n-3)!^2 (2^{156} \times 3^{64})^{16n^2+8kn} \\
 &\leq (2n-6)! (2^{156} \times 3^{64})^{16n^2+8kn}
 \end{aligned}$$

**Claim 8.15.** *If  $\varphi'(\rho)$  corresponds to a satisfying truth assignment for  $\Psi(\Gamma)$ , then*

$$\mathcal{H}(\varphi'(\rho)) = \left[ (n-3)!^2 (2^{136} \times 3^{76})^{16n^2+8kn} \right]^{k+4n} =: B_{good}.$$

*If  $\varphi'(\rho)$  corresponds to a truth assignment which does not satisfy  $\Psi(\Gamma)$ , then there must be a clause  $c_i$  for some  $i \in [k+4n]$  which is not satisfied. Therefore,*

$$\mathcal{H}(\varphi'(\rho)) \leq (2n-6)! (2^{136} \times 3^{76})^{16n^2+8kn} \left[ (2n-6)!^2 (2^{156} \times 3^{64})^{16n^2+8kn} \right]^{k+4n-1} =: B_{bad}.$$

Let  $B_{total}$  be the total number of most parsimonious SCJ scenarios with  $\mathcal{B}$  and leaf labeling  $\varphi$ , as in Definition 3.12. This is the sum of the number of scenarios admitted by each most parsimonious labeling  $\varphi'$ .

Given only  $B_{total}$ , we would like to determine the number of satisfying truth assignments for  $\Psi(\Gamma)$ .

$$\begin{aligned}
 B_{total} &= \sum_{\eta \in \mathcal{M}'_{\varphi}(\Gamma)} \mathcal{H}(\eta) + \sum_{\eta' \in \mathcal{M} \setminus \mathcal{M}'_{\varphi}(\Gamma)} \mathcal{H}(\eta') \\
 &= |S| B_{good} + \sum_{\eta' \in \mathcal{M} \setminus \mathcal{M}'_{\varphi}(\Gamma)} \mathcal{H}(\eta').
 \end{aligned}$$

As long as  $\sum_{\eta' \in \mathcal{M} \setminus \mathcal{M}'_{\varphi}(\Gamma)} \mathcal{H}(\eta') < B_{good}$ , we can conclude that the number of satisfying truth assignments for  $\Psi(\Gamma)$  (and for  $\Gamma$ ) is precisely

$$\left\lfloor \frac{B_{total}}{B_{good}} \right\rfloor.$$

Observe, for  $n \geq 2$ ,

$$\begin{aligned}
\frac{2^{2n} B_{bad}}{B_{good}} &= 2^{2n} \left[ \frac{3^{12}}{2^{20}} \right]^{16n^2+8kn} \binom{2n-6}{n-3}^{k+4n} \\
&< 2^{2n} \left[ \frac{3^{12}}{2^{20}} \right]^{16n^2+8kn} 2^{2n(k+4n)} \\
&< 2^{8n^2+2kn+2n} \left[ \frac{3^{12}}{2^{20}} \right]^{16n^2+8kn} \\
&< 2^{8n^2+4kn} \left[ \frac{3^{12}}{2^{20}} \right]^{16n^2+8kn} \\
&= \left[ \frac{3^{12}}{2^{20-1/2}} \right]^{16n^2+8kn} \\
&< 1.
\end{aligned}$$

Consequently, we obtain our desired result:

$$\sum_{\eta' \in U} \mathcal{H}(\eta') \leq \sum_{\eta' \in U} B_{bad} \leq 2^{2n} B_{bad} \leq B_{good}.$$

Therefore, if we could determine the total number of most parsimonious scenarios for this binary tree in polynomial time, then we could obtain the total number of satisfying assignments for  $\Psi(\Gamma)$  and for  $\Gamma$  in polynomial time. This completes the proof. ■

## 9. ACKNOWLEDGEMENTS

<sup>†</sup>This author acknowledges support from DARPA and AFOSR under contract #FA9550-12-1-0405, the NSF DMS contract 1300547, and a SPARC Graduate Research Grant from the Office of the Vice President for Research at the University of South Carolina.

## REFERENCES

- [1] E. Bach and J. Shallit. *Algorithmic Number Theory, Volume I: Efficient Algorithms*. MIT Press, 1996.
- [2] S. A. Cook. “The Complexity of Theorem-proving Procedures”. In: *Proceedings of the Third Annual ACM Symposium on Theory of Computing*. STOC ’71. New York, NY, USA: ACM, 1971, pp. 151–158.
- [3] P. L. Erdős and L. A. Székely. “On weighted multiways cuts in trees”. In: *Mathematical Programming* 65 (1-3 1994), pp. 93–105.
- [4] P. Feijão and J. Meidanis. “SCJ: A Breakpoint-Like Distance that Simplifies Several Rearrangement Problems”. In: *IEEE/ACM Transactions on Computational Biology and Bioinformatics* 8.5 (2011), pp. 1318–1329.
- [5] W. M. Fitch. “Toward Defining the Course of Evolution: Minimum Change for a Specific Tree Topology”. In: *Systematic Zoology* 20.4 (1971), pp. 406–416. ISSN: 00397989.
- [6] I. Miklós, Sándor Z. Kiss, and E. Tannier. “Counting and sampling SCJ small parsimony solutions”. In: *Theoretical Computer Science* 552 (2014), pp. 83–98.
- [7] B. Rosser. “Explicit bounds for some functions of prime numbers”. In: *American Journal of Mathematics* 63.1 (1941), pp. 211–232.
- [8] D. Sankoff and P. Rousseau. “Locating the vertices of a Steiner tree in an arbitrary metric space”. In: *Math. Prog.* 9.1 (1975), pp. 240–246. ISSN: 0025-5610. DOI: 10.1007/BF01681346.
- [9] L.G. Valiant. “The complexity of computing the permanent”. In: *Theo. Comp. Science* 8.2 (1979), pp. 189–201.