Abstract

In IPCO 2002, Letchford and Lodi describe an algorithm for separating simple comb inequalities that runs in $O(n^3m^3 \log n)$ time, where $n$ and $m$ are respectively the number of nodes and arcs in the support graph of the point to be separated.

In this extended abstract, we demonstrate that the above algorithm separates over a superclass of simple comb inequalities, which we call simple domino parity inequalities. We also prove several structural properties about the support graph of points in the subtour polytope. With these, we design a faster separation algorithm for simple domino parity inequalities. Our algorithm runs in $O(n^2m^2 \log \frac{n^2}{mn})$ time.
1 Introduction

The symmetric traveling salesman problem (STSP) is defined on an edge-weighted, undirected graph \( G = (V, E) \). Given a cost vector \( c : E \to \mathbb{R}^+ \) that satisfies \( c_{ij} = c_{ji} \), the object is to select a minimum cost subset \( F \subseteq E \) of edges such that \( (V, F) \) is connected and the degree of every vertex is two. In other words, \( F \) describes a tour: a simple cycle that passes through every node in \( V \).

The STSP can be formulated as the following linear integer program. A tour vector \( x \) corresponding to tour \( F \) satisfies \( x(e) = 1 \) if and only if \( e \) is included in \( F \). Let \( \delta(S) \) (respectively \( E(S) \)) be the subset of edges \( (v, w) \in E \) with \( |\{ v, w \} \cap S | = 1 \) (respectively \( |\{ v, w \} \cap S | = 2 \)). Given vector \( x \in \mathbb{R}^{|E|} \) and \( E' \subseteq E \), let \( x(E') \) be defined as \( \sum_{e \in E'} x_e \).

\[
\begin{align*}
\text{minimize} & \quad c^T x \\
\text{subject to} & \quad x(\delta(v)) = 2 \quad \forall v \in V \\
& \quad x(E(S)) \leq |S| - 1 \quad \forall S \subset V \\
& \quad x(e) \in \{0, 1\} \quad \forall e \in E
\end{align*}
\]

The constraints (1) are called degree constraints, and the inequalities (2) are called the subtour elimination constraints (SECs). The convex hull of tour vectors is called the STSP polytope. The polytope obtained by replacing the integrality constraints (3) by nonnegativity constraints is called the subtour polytope. The optimal solution \( x^* \) to the optimization problem over the subtour polytope may not be a tour vector. In this case it is desired to find an inequality \( ax \leq b \) that is satisfied by all tours, but not by \( x^* \). One type of especially strong inequalities is the set of comb inequalities. A comb is defined by subsets \( H \subset V \) and \( T_j \subset V \) for \( j = 1, \ldots, t \), with \( t \geq 3 \) and odd, such that \( T_j \cap H \neq \emptyset, T_j \setminus H \neq \emptyset, \) and \( T_i \cap T_j = \emptyset \) for all \( 1 \leq i, j \leq t \). \( H \) is the handle and the set \( \{ T_1, \ldots, T_t \} \) are the teeth. The corresponding comb inequality,

\[
x(\delta(v)) = 2 \quad \forall v \in V \\
x(E(S)) \leq |S| - 1 \quad \forall S \subset V \\
x(e) \in \{0, 1\} \quad \forall e \in E
\]

is a facet for the polytope that is the convex hull of all tours [8, 9].

There are an exponential number of comb inequalities, so it is not feasible to add all of them to the subtour polytope formulation. Thus, it would be useful to have a separation algorithm for comb inequalities. A separation algorithm for a class \( \mathcal{I} \) of inequalities takes as input a vector \( x \) and returns either an inequality in \( \mathcal{I} \) that is violated by \( x \), or a proof that \( x \) satisfies all inequalities in \( \mathcal{I} \). It is a long-standing open question to find a polynomial time separation algorithm for comb inequalities.

Comb inequalities are a subclass of the set of \( \{0, \frac{1}{2}\} \)-Chvátal-Gomory cuts of the STSP polytope. Given an integer polytope \( P_I := \text{conv}\{ x \in \mathbb{Z}^m : Ax \leq b \} \) where \( A \in \mathbb{Z}^{r \times m}, b \in \mathbb{Z}^r \), a \( \{0, \frac{1}{2}\} \)-Chvátal-Gomory cut is an inequality for \( P_I \) of the form \( \lambda A x \leq \lfloor \lambda b \rfloor \) where \( \lambda \in \{0, \frac{1}{2}\}^r \), \( \lambda A \) is integral, and \( \lambda b \) is fractional. All
\{0, \frac{1}{2}\}\)-Chvátal-Gomory cuts are valid for \(P_I\): Since \(\lambda A x \leq \lambda b\) is clearly valid for \(P_I\), and \(\lambda A\) is integral, we can strengthen the inequality so that it still holds for all \(x \in P_I\) by rounding \(\lambda b\) down to the nearest integer. For convenience, we will refer to these cuts as \(\{0, \frac{1}{2}\}\) inequalities or \(\{0, \frac{1}{2}\}\) cuts.

Since comb inequalities are important facets of the STSP, there has been considerable effort invested in designing algorithms to find violated ones. Effective heuristics are described in [16, 1, 13]. Algorithms with theoretical guarantees include, in chronological order: comb inequalities with \(t\) teeth can be separated using \(O(n^2 t)\) maximum flow computations [5]; comb inequalities that are violated by exactly \(\frac{1}{2}\), the maximum amount, can be separated in \(O(n^2 \log n)\) time in planar support graphs [7]; \(\{0, \frac{1}{2}\}\) cuts that are violated by exactly \(\frac{1}{2}\) can be separated in \(O(n^2 m)\) time [3]; a subclass of \(\{0, \frac{1}{2}\}\) cuts that includes general comb inequalities can be separated in \(O(n^3)\) time in planar support graphs [11]; \(\{0, \frac{1}{2}\}\) cuts with a fixed handle can be separated with a high-order, but polynomial, number of steps [4].

In this paper, we describe a polynomial time separation algorithm for simple domino parity inequalities. Simple domino parity are a subset of \(\{0, \frac{1}{2}\}\) cuts of the subtour polytope that include simple comb inequalities as defined in [12], but do not include all comb inequalities. Precise definitions and the relations of these inequalities are given in the next section.

### 1.1 Simple Domino Parity Inequalities

A tooth \((S, T)\) consists of two disjoint vertex sets \(S\) and \(T\) such that \(V - (S \cap T) \neq \emptyset\). A simple tooth \((i, S)\) consists of a root \(i \in V\) and a disjoint subset of vertices \(S \subset V - \{i\}\), called a body. The subset of \(E\) with one end point in \(S\) and the other in \(T \subset V - S\) is denoted \(E(S : T)\). Given a vector \(x \in \mathbb{R}^{|E|}\), the tooth inequality for \((S, T)\) is

\[
2x(E(S)) + 2x(E(T)) + x(E(S : T)) \leq 2|S \cup T| - 3.
\]

For the simple tooth \((i, S)\), this can be expressed as the simple tooth inequality

\[
2x(E(S)) + x(E(i : S)) \leq 2|S| - 1.
\]

The simple tooth inequality for \((i, S)\) is obtained by summing together the subtour constraints on \(S\) and \(S \cup \{i\}\). Thus it is valid and implied by the subtour polytope.

A domino parity (DP) inequality is a \(\{0, \frac{1}{2}\}\) inequality derived from the tooth inequalities, degree constraints and nonnegativity constraints. A simple DP inequality is a DP inequality for which all the teeth are simple. In order that the (simple) DP inequalities are not implied by the subtour polytope, it is necessary that their derivation include an odd number of tooth inequalities.

A comb inequality is a DP inequality for which the teeth corresponding to the tooth inequalities used in the derivation are disjoint [12]. A simple comb inequality is a comb inequality in which all teeth are simple.
Hence, the simple DP inequalities include all simple comb inequalities. A special subclass of simple combs are the two-matching inequalities [6], which have an efficient separation algorithm [15]. The simple DP inequalities do not include general comb inequalities.

In [12], Letchford and Lodi claim a separation algorithm for simple comb inequalities that runs in $O(n^3 m^3 \log n)$ time, where $m$ and $n$ are respectively the number of edges and vertices of the support graph on which their algorithm is run. This claim depends on the following conjecture (stated implicitly as a fact in [12]):

**Conjecture 1.1** [12] All simple DP inequalities are dominated by simple comb inequalities, subtour constraints, degree constraints, and nonnegativity inequalities.

The algorithm in [12] depends on partitioning the candidate tooth inequalities by the slack of the inequality. The *slack* of an inequality $a^T x \leq b$ is the quantity $b - ax$.

**Lemma 1.2** The sum of the slacks of the tooth inequalities and nonnegativity inequalities used to derive a violated simple comb inequality is $< 1$.

**Corollary 1.3** At most one tooth inequality with slack $\geq \frac{1}{2}$ is used to derive a violated simple comb.

A simple tooth whose inequality has slack less than $\frac{1}{2}$ is called light; a simple tooth whose inequality has slack $\geq \frac{1}{2}$ but $< 1$ is called heavy. Letchford and Lodi [12] show that the number of light teeth is $O(n^2)$ and the number of heavy teeth is $O(n^3)$. They also show that for separation, it suffices to consider $O(m)$ light teeth. The run time of their algorithm depends on the number of light and heavy teeth considered.

**Conjecture 1.4** [12] The number of light teeth is $O(n)$.

**Conjecture 1.5** [12] The number of heavy teeth is $O(mn)$.

1.2 Our results

We give a counterexample to Conjecture 1.1 (Section 2) by displaying a violated simple DP inequality that is not dominated by simple combs, subtours, degree or nonnegativity constraints. In Section 3, we describe a separation algorithm for simple DP inequalities that runs in $O(n^2 m^2 \log(n^2/m))$ time. We do this by proving that the number of light teeth considered by a separation algorithm can be restricted to a set of size $O(n)$, by proving that the number of heavy teeth considered by a separation algorithm can be restricted to a set of size $O(mn)$ with special structure, and finally by modifying the algorithm in [12] to find these sets efficiently and use them appropriately. What we prove is not exactly Conjecture 1.4 and Conjecture 1.5,
but suffices to obtain the improved run time that motivates that conjecture. The outline of the proofs are contained in Section 4 and Section 5. Due to space limitations, proofs of more straightforward statements are omitted and appear in the full paper. All other proofs are contained in the Appendix.

2 Not all simple DP cuts are simple combs

In this section, we describe a counterexample to Conjecture 1.1. Figure 1 contains a point inside the subtour polytope that does not violate any comb inequality, but is violated by a simple DP inequality. The violated simple DP inequality is derived using 5 teeth: \{\{a, g\}, \{c, e\}, \{d, f\}, \{b, i\}, \{h, c, e, d, f\}\}, and the degree equations for \{e, f, g, h, i\}. This structure differs from simple comb inequalities in two respects: some teeth are nested inside another teeth, and the body of one tooth crosses the handle, which is defined as the set of vertices for which the degree constraints are included in the derivation.

\[
\begin{array}{cccc}
 a & g \\
 0.33 & 0.33 & 0.67 \\
 0.33 & 0.67 \\
 c & e \\
 0.67 & 0.33 & 0.67 & h \\
 d & f \\
 & 0.33 & 0.67 \\
 & 0.33 & 0.33 \\
 b & i \\
\end{array}
\]

Figure 1: A point inside the subtour polytope on 9 vertices for which there is a violated simple DP inequality but no violated comb inequalities. The simple DP inequality is derived using tooth inequalities of 5 teeth: \{\{a, g\}, \{c, e\}, \{d, f\}, \{b, i\}, \{h, c, e, d, f\}\}, and degree constraints on the dark vertices. The unmarked, dark lines have weight 1.

The resulting simple DP inequality is

\[x(E(H)) + \sum_i x(E(T_i)) - x_{eh} - x_{fh} \leq 10.\]

The point depicted in Figure 1 has left-hand-side value of 10.33. It is a vertex of the polytope that is described by \(x_{ag} = x_{bi} = x_{ce} = x_{df} = 1, x_e = 0\) if \(e\) is not in support of the point in Figure 1, all
degree constraints, the subtour constraint on \( \{c, d, e, f, h\} \), and the two comb inequalities implied by the following sets of handles and teeth: \( H1 = \{g, h, i\} \), \( T1a = \{a, g\} \), \( T1b = \{b, i\} \), \( T1c = \{c, d, e, f, h\} \), and \( H2 = \{c, d, h\} \), \( T2a = \{c, e\} \), \( T2b = \{d, f\} \), \( T2c = \{a, b, g, h, i\} \).

3 Separation of simple DP Inequalities

We describe a new separation algorithm for simple DP inequalities that improves upon [12] by reducing the run time complexity by a factor of \( O(mn) \). Our improvement relies on procedures that reduce both the number of light teeth and the number of heavy teeth that are considered by the algorithm.

We build on a result of Caprara and Fischetti that relates polynomial time separable cases of \( \{0, \frac{1}{2}\} \) cuts with EPT matrices. The first step in this relation associates the slack of inequalities used in the derivation to the violation of the derived inequality. The following is a straightforward extension of Lemma 1.2.

Proposition 3.1 Let \( x^* \) be a point in the subtour polytope. Then a \( \{0, \frac{1}{2}\} \)-cut is violated by \( x^* \) if and only if the sum of the slacks of the inequalities used, computed with respect to \( x^* \), is less than 1.

3.1 EPT Matrices

For general polytopes, the separation problem for the class \( \{0, \frac{1}{2}\} \) inequalities is NP-complete [2]. However, for some polytopes \( P_I \) with special structure, polynomial time separation algorithms exist. One such class of such polytopes is defined using edge-path incidence matrices of trees.

A \( p \times q \) \( \{0, 1\} \)-matrix \( M \) is the edge-path incidence matrix of a tree (EPT matrix) if there is a tree \( T \) on \( p + 1 \) vertices such that each row of \( M \) corresponds to an edge of \( T \) and each column of \( M \) is the incidence vector of the edges in a path in \( T \). Let \( M \) be an EPT matrix; and define the graph \( G \) on the same vertex set to contain an edge \( e_c \) for every column \( c \) of \( M \). The end points of edge \( e_c \) are the end points of the path \( P_c \) represented by \( c \). Then \( M \) is the matrix whose rows correspond to the incidence vectors of the fundamental cuts of \( G \) with respect to \( T \). We call the pair \((G, T)\) the witness for \( M \).

If the matrix representing a set of constraints can be represented as the vertical concatenation of an EPT matrix and an identity matrix, then Caprara and Fischetti [2] observe that it is possible to separate over \( \{0, \frac{1}{2}\} \)-cuts derived from such constraints in polynomial time via an odd minimum cut calculation in \( G \).

Lemma 3.2 ([2]) If \( M \) is an EPT matrix then it is possible to separate over \( \{0, \frac{1}{2}\} \)-cuts derived from constraints with matrix \( \begin{bmatrix} M & I \end{bmatrix} \).
Our algorithm, described below, separates simple DP inequalities by separating the \( \{0, \frac{1}{2}\} \) cuts of a sequence of systems that can each be represented as a concatenation of an EPT matrix and an identity matrix. This is the model proposed by Letchford and Lodi [12]. We show how to do it more efficiently.

The general outline of the algorithm is to first find all possible light and heavy teeth; reduce this set as much as possible while not reducing the set of \( \{0, \frac{1}{2}\} \) cuts that may be obtained using the inequalities implied by this set along with the degree constraints and nonnegativity inequalities; solve the separation problem for the set of inequalities implied by just the light teeth; arrange the heavy teeth into a small number of subsets so that it is possible to separate over the \( \{0, \frac{1}{2}\} \) cuts implied by the inequalities implied by the teeth in each subset together with the set of light teeth; and solve the resulting sequence of separation problems.

### 3.2 Structure of Simple Teeth

In this section, we outline some of the structure of light and heavy teeth, and what this implies about bounding the number of light and heavy teeth considered by our algorithm. The first two lemmas are due to Letchford and Lodi [12]. The first lemma is not stated explicitly in their paper, but is implicit in and central to their work. Theorem 3.6 and Theorem 3.7 are the two main theorems of this paper. First, a definition of a term used below: laminar sets are a collection of subsets of a groundset \( V \) such that if \( S \) and \( T \) are in the collection, then at least one of the following three sets are empty: \( S \cap T, S \setminus T, T \setminus S \).

**Lemma 3.3** If \( T \) is a set of simple teeth such that for all roots \( i \) the set of bodies of teeth with root \( i \) form a laminar set, then the matrix corresponding to the tooth inequalities for \( T \) and the degree constraints is an EPT matrix.

**Lemma 3.4** ([12]) The set of bodies that form light teeth with a given root \( i \) is laminar.

**Corollary 3.5** The \( \{0, \frac{1}{2}\} \) inequalities derived from light teeth, degree constraints, and nonnegativity constraints can be separated in polynomial time.

Another implication of Lemma 3.4 is that the total number of light teeth is \( O(n^2) \). Theorem 3.6 below implies that it is sufficient to consider a set of size \( O(n) \). The proof is contained in Section 4.

**Theorem 3.6** There exists a set of light teeth \( \mathcal{L} \) of size \( O(n) \) such that if there exists a violated simple DP inequality, then there exists one with light teeth from the set \( \mathcal{L} \) only. The set \( \mathcal{L} \) can be found in \( O(n^3m) \) time.

Our second main theorem implies that it is sufficient to consider a set of heavy teeth of size \( O(mn) \) that has a special structure. The proof of this theorem is contained in Section 5.
Theorem 3.7 There is a set of heavy teeth of size $O(mn)$, such that the heavy teeth with root $i$ can be partitioned into $|\delta(i)|$ laminar subsets; and if there exists a violated simple DP inequality derived using a heavy tooth, then there exists one with its heavy tooth in this set. This set has $O(mn)$ size and can be found in $O(n^3 m)$ time.

3.3 The Separation Algorithm

Our separation algorithm for simple DP inequalities relies on a subroutine $\text{buildT}(S)$ that takes as input a set of teeth $S$ such that tooth inequalities for teeth in $S$ and the degree constraints form an EPT matrix $M$, and gives as output the tree $T_S$ and corresponding graph $G_S = (V_S, E_S)$ such that $(G_S, T_S)$ is a witness for $M$. The set $S$ is determined by our current solution $x$ and the set of light and heavy teeth inequalities it induces. Details of $\text{buildT}$ are contained in the Appendix. The next lemma explains how $G_S$ is useful.

Lemma 3.8 A simple DP inequality derived from tooth inequalities in $S$, degree constraints, and nonnegativity inequalities is violated by a point in the subtour polytope if and only if the corresponding edges form a cutset in $G_S$ that contains an odd number of odd edges and has weight less than 1.

SimpleDPSep($G, x$)

1. Find all simple tooth inequalities with $x$-slack less than 1.
2. Reduce the subset of light teeth to a set $L$ of size $O(n)$ by uncrossing. (Theorem 3.6)
3. $(G_L, T_L) \leftarrow \text{buildT}(L)$.
4. Find a minimum odd cut in $G_L$.
5. If weight of cut is $< 1$, output candidate inequality.
6. Reduce the subset of heavy teeth to $O(m)$ laminar sets. (Theorem 3.7)
7. For each root $i$,
8. $L_i \leftarrow$ the set of light teeth with root $i$.
9. Partition the set of heavy teeth with root $i$ into $|\delta(i)|$ laminar subsets, $\{H^i_1, \ldots, H^i_{|\delta(i)|}\}$. (Theorem 3.7)
10. For each such subset $H^i_j$,
11. $K \leftarrow (L - L_i) \cup H^i_j$.
12. $(G_K, T_K) \leftarrow \text{buildT}(K)$.
13. Find the minimum odd cut in $G_K$.
14. If weight of cut is $< 1$, output candidate inequality.

Figure 2: Separation algorithm for simple domino parity inequalities.

The algorithm, outlined in Figure 3.3, first looks for simple DP inequalities that are derived using light teeth only. Then, it looks for simple DP inequalities that use one heavy tooth. By Theorem 3.7, Lemma 3.8, and...
and the following Lemma 3.9, this can be done as follows: for each $i \in V$, consider at one time all light teeth with roots in $V - \{i\}$ and a subset of heavy teeth with root $i$, and check the corresponding graph $G_S$ for minimum odd cut.

**Lemma 3.9** If a point in the subtour polytope violates a simple DP inequality, then it violates a simple DP inequality with the property that no two teeth corresponding to tooth inequalities used in the derivation share the same root.

Our main algorithmic result is summarized in the next theorem.

**Theorem 3.10** SimpleDPSep is a separation algorithm for simple domino parity inequalities that runs in $O(n^2m^2 \log n m^2)$ time.

4 Proof of Theorem 3.6

The proof of Theorem 3.6 works in two steps. In the first step, we show that the number of roots $i$ that form a light tooth with any fixed body $S$ is at most 3. In the second step, we show that it is possible to obtain a laminar set of bodies such that all light teeth we consider have a body in this set. This implies that the number of bodies we consider is $O(n)$.

**Lemma 4.1** At most 3 distinct light teeth share the same body.

Let $(i, X)$ and $(i, Y)$ be two teeth with the property that $X \subseteq Y$. If the slack of the tooth inequality on $(i, Y)$ is at least the slack of the tooth inequality on $(i, X)$ plus the slack of the nonnegativity constraints for $E(i : Y - X)$, then we say that $(i, X)$ improves $(i, Y)$.

Sets $S, T \subseteq V$ are said to cross if $S \cap T, S - T, T - S, V - (S \cup T)$ are all nonempty. For tooth $(i, S)$ define $S'_i = V - S - \{i\}$.

**Lemma 4.2** The tooth inequality for $(i, S)$ is equivalent to the tooth inequality for $(i, S'_i)$.

Instead of multiplying inequalities in the derivation of a $\{0, \frac{1}{2}\}$ inequality by $\frac{1}{2}$, we can simply add inequalities together and consider the derived inequality modulo 2. The $\{0, \frac{1}{2}\}$ inequalities are then inequalities with odd right hand side and even coefficients on the left hand side. To obtain even coefficients on the left, for a fixed set of tooth inequalities and degree constraints, it may be necessary to add nonnegativity inequalities.

If the tooth inequality for $(i, S)$ is used in the derivation of some $\{0, \frac{1}{2}\}$ inequality, it contributes an odd amount to the right hand side, and an odd amount to the coefficients of all edges in $E(i : S)$. Thus, removing...
(i, S) and replacing it with (i, S − T) changes the parity only of coefficients of edges in E(i : S ∩ T). If we then add (or remove) the nonnegativity constraints for these edges, no parities are changed, and the inequality remains valid. The next lemma describes how this can be useful to uncross bodies of teeth. The proof is straightforward but technical, and is contained in the full paper.

**Lemma 4.3** If (i, S) and (j, T) are two teeth and S and T cross, then either one of the following four conditions holds strictly, or two conditions hold exactly:

i) the slack on tooth inequality (i, S − T) plus the slack for the nonnegativity constraints for E(i : S ∩ T) is at most the slack of tooth inequality (i, S),

ii) the slack on tooth inequality (i, S ∪ T) plus the slack for the nonnegativity constraints for E(i : T − S) is at most the slack of tooth inequality (i, S),

iii) the slack on tooth inequality (j, T − S) plus the slack of the nonnegativity constraints for E(j : S ∩ T) is at most the slack of tooth inequality (j, T),

iv) the slack on tooth inequality (j, S ∪ T) plus the slack of the nonnegativity constraints for E(j : S − T) is at most the slack of tooth inequality (j, T).

The next lemma describes why uncrossing teeth is useful in bounding L.

**Lemma 4.4** Let L be a set of light teeth that satisfies the following property. For all pairs (i, S) and (j, T) in L, at most one of the following pairs of bodies cross: (S, T), (S, T′ j), (S′ i, T), (S′ i, T′ j). Then, the size of L is O(n).

We say that tooth (i, S) t-crosses tooth (j, T) if either S or S′ i crosses both T and T′ j or either T or T′ j crosses both S and S′ i. Since i can be in at most one of T and T′, this implies that (i, S) and (j, T) do not cross if and only if at most one following pairs of bodies cross: (S, T), (S, T′ j), (S′ i, T), and (S′ i, T′ j). If two teeth t-cross, we can apply Lemma 4.3 to uncross them.

**Lemma 4.5** There is a set of light teeth L with |L| = O(n) such that if there is a violated simple DP inequality derived using tooth inequalities of light teeth only, then there is a simple DP inequality derived using tooth inequalities from the set L only. Given laminar sets of i-teeth for all i, the set L can be found in O(n^3 m) time.

The final piece of the proof involves establishing the time it takes to go from the list of light teeth sorted by root obtained in line (1) of the algorithm to an organized laminar set of bodies for each root. For the proof of Lemma 4.5, all that is needed is that the bodies be sorted according to size. Naively, this takes at most O(n^2) time per root, or O(n^3) overall.
5 Proof of Theorem 3.7

We begin the proof with a theorem about the structure of heavy teeth. Although the \( i \)-heavy sets need not be nested, they satisfy a certain ‘circular’ property.

**Theorem 5.1** Let \( i \in V \) be a given root. There is a cyclic ordering of the vertices in \( V \setminus \{i\} \) such that each \( i \)-heavy set is the union of consecutive vertices in the ordering.

The proof is based on the following: Let \( j \in V \setminus \{i\} \) be an arbitrary vertex, and let \( M \) be a 0-1 matrix whose columns correspond to the vertices in \( V \setminus \{i, j\} \), and whose rows are the incidence vectors of the \( i \)-heavy sets which do not include \( j \). Due to Lemma 4.2, the theorem is true if and only if the columns of \( M \) can be permuted so that, in every row of the resulting matrix, the 1s occur consecutively. Then, from Theorem 9 of Tucker [17] on matrices with the consecutive 1s property, it suffices to prove five claims that disallow certain arrangements of \( i \)-heavy teeth. The statement and proof of the claims appear in the full paper. An important corollary of this is the following.

**Corollary 5.2** For a fixed root \( i \), the \( i \)-heavy sets can be partitioned into \( O(n) \) nested families.

The following lemma, from [12], enables us to eliminate teeth from consideration. In conjunction with Theorem 5.1, this will yield a new, reduced bound on the number of heavy teeth to be considered.

**Lemma 5.3 ([12])** Suppose a violated \( \{0, \frac{1}{2}\} \)-cut can be derived using the tooth inequality with root \( i \) and body \( S \). If there exists a set \( S' \subset V \setminus \{i\} \) such that

\[
\begin{align*}
\bullet & \quad E(i : S) \cap E^* = E(i : S') \cap E^*, \\
\bullet & \quad 2|S'| - 2x^*(E(S')) - x^*(E(i : S')) \leq 2|S| - 2x^*(E(S)) - x^*(E(i : S))
\end{align*}
\]

then we can obtain a \( \{0, \frac{1}{2}\} \)-cut violated by at least as much by replacing the body \( S \) with the body \( S' \) (and adjusting the set of used non-negativity inequalities accordingly).

The next theorem shows that, after Lemma 5.3 is applied, relatively few heavy teeth remain.

**Theorem 5.4** After applying the elimination criterion of Lemma 5.3, only \( O(nm) \) heavy tooth inequalities remain, and these can be partitioned into \( O(m) \) nested families.

To complete the proof of Theorem 3.7, it remains to show that the reduction and reorganization of heavy teeth from \( O(n^3) \) teeth in lists sorted by root to \( O(m) \) nested families can be accomplished in \( O(n^3m) \) time. We sketch the basics in the Appendix.
References


Appendix

Details of \texttt{buildT}: Given a set of teeth \(S\) such that the bodies of all teeth with root \(i\) form a laminar set, Lemma 3.3 says the corresponding matrix \(M\) of degree constraints and tooth inequalities is EPT. The routine \texttt{buildT}(\(S\)) builds the witness \((G_S, T_S)\) as follows. The tree \(T_S\) contains a central vertex \(r\) and for each \(i \in V\), a subtree \(T_i\) rooted at a node \(v_i\) connected to \(r\) with an edge \((r, v_i)\). The subtree \(T_i\) describes the containment relation of the laminar sets that are bodies of teeth with root \(i\). For \(S\) and \(S'\) that are both bodies of teeth with root \(i\) in \(S\), if \(S' \subset S\) then \(v_S\) is the parent of \(v_{S'}\) in \(T_i\). For all maximal bodies \(S\) with
root \( i \), the parent of \( v_S \) is \( v_i \). The edge \((r, v_i)\) corresponds to the degree constraint for \( i \), has weight 0, and is labeled even. The last edge on the path from \( v_i \) to \( v_S \) corresponds to the tooth inequality for tooth \((i, S)\), has weight equal to the \( x \)-slack of this inequality, and is labeled odd. Note that if \( S \) is a body of a tooth with root \( i \) and \( j \in S \) then \((i, j)\) has odd coefficient in all inequalities corresponding to edges on the path from \( r \) to \( S \) through \( v_i \).

The graph \( G_S \) contains \( T_S \) plus a set of edges with positive \( x \)-value. These correspond to nonnegativity constraints with positive slack. For edge \((i, j)\) in the support of \( x \), let \( S^i_j \) be the body of the smallest tooth in \( S \) with root \( i \) that contains \( j \). If there is no such tooth, then \( S^i_j = \{i\} \). An edge is added to \( G' \) from \( S^i_j \) to \( S^j_i \). This edge has weight equal to its \( x \)-value, and is labeled even.

\( \text{buildT}(S) \) runs easily in time \( O(nm) \) on a set \( S \) of size \( n \)

**Proof of Theorem 3.10:** If there is a violated simple DP inequality using only teeth in \( L \), then by Lemma 3.8 and Theorem 3.6, \( \text{SimpleDPSep} \) finds it in line 4. Otherwise, using Lemmas 3.9 and 3.8 and Theorems 3.7 and 3.6, if there is a violated simple DP inequality using a heavy tooth with root \( i \), \( \text{SimpleDPSep} \) finds it in line 13. This establishes correctness. Now we focus on run time.

The algorithm of Nagamochi, Nishimura, and Ibaraki [14] finds all sets \( S \) with \( x(\delta(S)) < 3 \) in \( O(m^2n + mn^2 \log n) \) time [10]. The number of such sets is at most \( O(n^2) \) [10]. Each of the sets can be the body or union of body and root of at most \( n \) light or heavy teeth. These \( O(n^3) \) teeth can then be found and sorted according to root in \( O(n^4) \) additional time. Thus the time required for step 1 is \( O(n^4 + m^2n + mn^2 \log n) \).

Theorem 3.6 implies that time required for Step 2 is \( O(n^3m) \). Details of \( \text{buildT} \) imply that the total time spent in this subroutine over the course of the algorithm (lines 3 and 12) is at most \( O(nm^2) \).

For the light teeth, this results in a graph \( G_L \) containing a tree \( T_L \) with a root with \( n \) branches, and a total number of nodes of \( O(n) \). There is an edge for every edge in the support graph, so the number of edges is \( O(m) \). On this graph, it takes \( O(n^2m \log \frac{n^2}{m}) \) time to find a minimum odd cut using a Gomory-Hu tree. (lines 4)

By Theorem 3.7, lines 7-9 take total time at most \( O(n^3m) \). Lines 10-11 are not a bottleneck. To find violated inequalities that use a heavy tooth with root \( i \), the graph \( G_K \) still contains \( O(n) \) nodes (nodes from the original \( O(n) \) light teeth plus \( O(n) \) nodes for the laminar teeth with root \( i \)) and \( O(m) \) edges.

By Theorem 3.7, it suffices to consider \( O(m) \) of these graphs. Thus the time spent on these graphs is \( O(n^2m^2 \log \frac{n^2}{m}) \) (line 13).

Given an odd cut in \( G_S \), the corresponding inequality can be recovered in time \( O(n) \) times the size of the cut, i.e. in \( O(nm) \) time (lines 5, 14).

**Proof of Lemma 4.1**: Consider a body \( S \). The subtour constraint for \( S \) imposes that \( x^*(E(S)) \leq |S| - 1 \).
The total weight leaving $S$, $x^*(\delta(S))$, equals $2|S| - 2x^*(E(S))$, which can be seen by summing degree constraints on nodes in $S$. The tooth inequality with body $S$ and root $i$ imposes that $x^*(E(i : S)) \leq 2|S| - 1 - 2x^*(E(S))$. Thus if $(i, S)$ is a light tooth, then $x^*(E(i : S)) > 2|S| - \frac{3}{2} - 2x^*(E(S))$. Let $\ell(S)$ be the number of light teeth with body $S$. Summing over all roots of light teeth, we have that

$$\ell(S)(2|S| - \frac{3}{2} - 2x^*(E(S))) < \sum_{i:(i, S)} x^*(E(i : S)) \leq x^*(\delta(S)) = 2|S| - 2x^*(E(S)),$$

so that

$$\ell(S) < \frac{2|S| - 2x^*(E(S))}{2|S| - \frac{3}{2} - 2x^*(E(S))}.$$

The right hand side of this expression is maximized when $x^*(E(S))$ is at its upper bound of $|S| - 1$, and hence equals $\frac{3}{2} = 4$. Since $\ell(S) < 4$ and integral, we have that $\ell(S) \leq 3$. \hfill \blacksquare

**Proof of Lemma 4.4:** We first remove all teeth in $L$ of the form $(i, j)$ or $(i, V - \{i, j\})$. By complementing with respect to some $i \in V$, this set of teeth has bodies in $V$. Let $L'$ be the set of teeth in $L$ that are not of this form.

For the teeth in $L'$, we construct the following graph: There are $|L'|$ pairs of vertices. Each pair corresponds to a tooth in $L'$. For tooth $(i, S)$ the first vertex corresponds to $S$, the second to $S'_j$. There is an edge joining vertices for $S$ and $S'_j$. There is also an edge joining each pair of vertices that correspond to bodies that cross. A maximal independent set $I$ in this graph corresponds to a set of laminar bodies $K$. Thus, this set $K$ has size $O(n)$. Associate each $S \in K$ with the light teeth in $\{(i, S)|i \in V\}$. Let $L'' = L' \cap \cup_{S \in K}\{(i, S)|i \in V\}$. By Lemma 4.1, the size of $L''$ is $O(n)$. Define $K'$ to be the set $\{S' \subset V | S' = V - S - \{i\} \text{ for } (i, S) \in L''\}$. Since $|K'| \leq |L''|$ we have that $|K'| = O(n)$.

Suppose that for tooth $(i, S)$, neither the vertex for $S$ nor the vertex for $S'$ are in $I$. Then both $S$ and $S'$ each cross some set in $K$. Suppose $S$ crosses $T$ for a tooth $(j, T) \in L''$, and $S'$ crosses $U$ for some tooth $(k, U) \in L''$. We claim that either $T' = S'$ or $U' = S$.

If this claim is true, then every tooth in $L$ has a body or complement body in $K$ or $K'$ or $V$. Since $|K \cup K' \cup V| = O(n)$, Lemma 4.1 implies that $|L| = O(n)$.

We now establish the claim. Since $S$ crosses $T$ it cannot cross $T'$, thus either $S \cap T' = \emptyset$ or $T' \subset S$. In the former case, $S - T = \{j\}$; in the latter, $T' \cap S' = \emptyset$. Since also $S'$ does not cross $T$, either $S' \cap T = \emptyset$ or $S' \subset T$. The former implies that $T - S = \{i\}$; the latter that $S' \cap T' = \emptyset$.

If both $S - T = \{j\}$ and $T - S = \{i\}$, then $S' = T'$, and we are done. Otherwise $S' \cap T' = \emptyset$. If in addition, either $S \cap T' = \emptyset$ or $S' \cap T = \emptyset$, then either $S' = \{j\}$ or $T' = \{i\}$. But we removed all such teeth from $L'$ at the start, so this case cannot occur. If neither $T' = S'$ nor $U' = S$, then it must be that $T' \subset S$, $S' \subset T$, $T' \subset S'$, and $S \subset U$. In this case, since $T$ crosses $S$ and contains $S'$, which contains $U'$, we then have that $T$ crosses $U$. But this contradicts $T, U \in K$. Thus either $T' = S'$ or $U' = S$, and the claim is
Proof of Lemma 4.5: We begin with the set of teeth $\mathcal{L}'$ given by Lemma 3.4. We will create a new set $\mathcal{L}$ such that each tooth in $\mathcal{L}'$ is improved by some tooth in $\mathcal{L}$. This is done on an incremental basis, until all teeth in $\mathcal{L}'$ are improved by some tooth in $\mathcal{L}$. We start with $\mathcal{L} = \emptyset$. Throughout, we maintain that no two teeth in $\mathcal{L}$ t-cross. Then, by Lemma 4.4 the size of $\mathcal{L}$ is $O(n)$ throughout the algorithm.

We begin with $\mathcal{L} = \{(1,S)|(1,S) \in \mathcal{L}'\}$. By Lemma 3.4, no two teeth in $\mathcal{L}$ t-cross. Then, for roots $i = 2$ through $n$, we orient all teeth $(j,T) \in \mathcal{L}$ (by perhaps replacing $(j,T)$ with $(j,T'_j)$) so that $T$ does not contain $i$. We consider one by one teeth in $\{(i,S) | S \subset V \} \cap \mathcal{L}'$; and for one such $(i,S)$ we apply Lemma 4.3 to $(i,S)$ and the teeth in $\mathcal{L}$ starting with the teeth with the smallest bodies. With such a procedure, each uncrossing produces a new tooth that does not t-cross any previously uncrossed tooth: If the bodies of $(j,T) \in \mathcal{L}$ and $(k,U) \in \mathcal{L}$ are nested before uncrossing one of them with $(i,S)$, then after uncrossing, their respective bodies are either still nested, or completely disjoint. Thus we end up with a final set of teeth that do not t-cross.

The time to uncross one pair of teeth is $O(m)$. Since the size of $\mathcal{L}$ is at most $O(n)$, the number of uncrossings per new tooth in $\mathcal{L}'$ is at most $O(n)$. The initial size of $\mathcal{L}'$ may be $O(n^2)$. Thus the total time taken by this routine is $O(n^3m)$.

Proof of Theorem 5.4: Let $d_i = |\delta(i)|$ By the circular property of $i$-heavy teeth (Theorem 5.1), the sets $E(i : S)$ also have a circular property. After applying the elimination criterion, there can be at most $d_i^2$ $i$-heavy teeth. So the total number of heavy tooth inequalities is at most $\sum_{i \in V} (d_i)^2 \leq n \sum_{i \in V} d_i = 2nm$. Moreover the $i$-heavy sets partition naturally into $d_i$ nested families, giving $\sum_{i \in V} d_i = 2m$ nested families in total.

Completion of proof of Theorem 3.7: From the sorted lists, we first apply Lemma 5.3. There are at most $O(n^2)$ $i$-heavy teeth in $i$’s list. For each tooth $(i : S)$, we create an ordered list of size at most $d_i$ of the edges in $E(i : S)$ in $O(d_i)$ time per set, and compute the value on the right of the expression of the second criteria of Lemma 5.3. We can sort these $O(n^2)$ lists with $O(n^2 \log n)$ comparisons, each comparison taking $O(d_i)$ time. The value computation takes at most $O(n^2m)$ time. After sorting, elimination of sets is not a bottleneck. Thus, over all roots $i$, the time spent in elimination is $O(n^3m)$.

For each $i$, it then takes $O(n)$ time per tooth to place it in a sorted family. To find the nesting order of each nested family for teeth with root $i$ then takes $O(nd_i^2 \log n)$ time, or $O(n^2m \log n)$ overall. This completes the proof.