A Study of Domino-Parity and k-Parity Constraints for the TSP

William Cook Daniel Espinoza Marcos Goycoolea

ISYE, Georgia Institute of Technology

Abstract. Letchford (2000) introduced the domino-parity inequalities for the symmetric traveling salesman problem as a superclass of the comb inequalities proposed by Chvatal (1973) and Grötschel and Padberg (1979). A key result of Letchford is that if the support graph of an LP solution is planar, then the separation problem for domino-parity inequalities can be solved in polynomial time. We generalize domino-parity inequalities to multi-handled configurations, introducing a superclass of bipartition and star inequalities. Also, we generalize Letchford's algorithm, proving that for a fixed integer k, one can separate a superclass of k-handled clique-tree inequalities satisfying certain connectivity characteristics with respect to the planar support graph. We describe an implementation of this algorithm which is exact for a single handle (that is, Letchford's Algorithm) and a heuristic for the case of two handles. This implementation includes pruning methods to restrict the search for dominoes, a parallelization of the main domino-building step, heuristics to obtain planar-support graphs, a safe-shrinking routine, a random-walk heuristic to extract additional violated constraints, and a tightening procedure to allow us to modify existing inequalities as the LP solution changes. We report computational results showing the strength of the new routines, including the optimal solution of the TSPLIB instance pla33810.

1 Introduction

Let $G = (V, E)$ be a complete graph with edge costs $(c_e : e \in E)$. The symmetric traveling salesman problem, or TSP, is to find a minimum-cost tour in G, that is, a Hamiltonian cycle of minimum total edge cost. A tour can be represented as a 0-1 vector $x = (x_e : e \in E)$, where $x_e = 1$ if edge e is used in the tour and $x_e = 0$ otherwise. In the Dantzig, Fulkerson, and Johnson [7] cutting-plane method for the TSP, a linear programming (LP) relaxation is created by iteratively finding linear inequalities that are satisfied by all tour vectors. This approach has been the most successful exact solution procedure proposed to date for the TSP; surveys of the large body of literature on the approach can be found in Jünger, Reinelt, and Rinaldi [12] and Naddef [14].

For any $S \subseteq V$, let $\delta(S)$ denote the set of edges with exactly one end in S and let $E(S)$ denote the set of edges having both ends in S. For disjoint sets $S, T \subseteq V$, let $E(S : T)$ denote the set of edges having one end in S and one end in T. For any set $F \subseteq E$, define $x(F) := \sum(x_e : e \in F)$.

Every tour of G satisfies the *subtour constraints*

$$x(\delta(S)) \geq 2 \qquad \forall\, \emptyset \neq S \subsetneq V. \tag{1}$$

An important property of these constraints is that the corresponding separation problem can be solved efficiently, that is, given a non-negative vector x^* a violated constraint can be found in polynomial time, provided one exists.

Much of the TSP literature is devoted to the study of classes of inequalities that are valid for the TSP, extending the subtour constraints in different ways. Many properties of these classes of inequalities are known, but for the most part polynomial-time separation algorithms have proven to be elusive. A notable exception is the separation algorithm for blossom-inequalities by Padberg and Rao [17]; variations of the Padberg-Rao algorithm are included in most current codes for the TSP. The absence of other efficient separation algorithms has lead to the use of various heuristic methods for handling TSP inequalities within cutting-plane algorithms. The heuristics are effective in many cases (see Padberg and Rinaldi [18], Applegate et al. [1], and Naddef and Thienel [16]), but additional exact methods could be critical in pushing TSP codes on to larger test instances.

An interesting new approach to TSP separation problems was adopted by Letchford [13], building on earlier work of Fleischer and Tardos [8]. Given an LP solution vector x^*, the support graph G^* is the subgraph of G induced by the edge-set $E^* = \{e \in E : x_e^* > 0\}$. Letchford [13] introduced a new class of TSP inequalities, called domino-parity constraints, and provided a separation algorithm in the case where G^* is a planar graph. An initial computational study of this algorithm by Boyd et al. [4], combining a computer implementation with by-hand computations, showed that the method can produce strong cutting planes for instances with up to 1,000 nodes.

In this paper we present a further study of Letchford's algorithm. We begin by describing a generalization of domino-parity inequalities and Letchford's algorithm to include certain multi-handled configurations. We also include a range of procedures for improving the practical performance of the separation routines, together with computational testing of large TSPLIB instances.

2 The k-Parity Inequalities

Definition 1. *Consider a family of sets $(T_1, T_2, \ldots, T_k; T)$ satisfying $\emptyset \neq T_i \subsetneq T \subsetneq V$, $\forall i \in I_k \equiv \{1, \ldots, k\}$. We call this family a* regular k-domino *if for any set $\emptyset \neq K \subseteq I_k$, the edges $\bigcup \{E(T_i : T \setminus T_i) : i \in K\}$ define a $|K| + 1$ (or greater) cut in the subgraph of G induced by T. The family is called a* degenerate k-domino *if $(T_1, \ldots, T_k)$ defines a partition of T. We refer to the $k-$domino $(T_1, \ldots, T_k; T)$ as T and define β_T as 1 if T is regular and as $\frac{k}{k-1}$ if T is degenerate. In general, we say that the sets $T_1, \ldots, T_k$ are the* halves *of T. Finally, if T is a k-domino, we say that $\kappa(T) = k$.*

Lemma 1. *Let $T = (T_1, T_2, \ldots, T_k; T)$ be a $k-$domino. If x satisfies all subtour constraints, then*

$$\frac{\beta_T}{2}(x(\delta(T)) - 2) + \sum_{i=1}^{k} x(E(T_i : T \setminus T_i)) \geq k.$$

Proof. Assume x satisfies all subtour constraints. Let $B_1, B_2, \ldots, B_r$ correspond to the partition of T obtained by removing the edge sets $E(T_1 : T \setminus T_1), E(T_2 : T \setminus T_2), \ldots, E(T_k : T \setminus T_k)$. Then

$$\sum_{i=1}^{r} x(\delta(B_i)) = x(\delta(T)) + \sum_{i=1}^{r} x(E(B_i : T \setminus B_i)).$$

It follows that

$$\frac{\beta_T}{2}\left(x(\delta(T)) - 2\right) = \frac{\beta_T}{2}\left(\sum_{i=1}^{r}\left(x(\delta(B_i)) - x(E(B_i : T \setminus B_i))\right) - 2\right). \tag{2}$$

However, note that if T is regular, then $\beta_T = 1$ and

$$\sum_{i=1}^{r} x(E(B_i : T \setminus B_i)) \leq 2\sum_{i=1}^{k} x(E(T_i : T \setminus T_i)).$$

On the other hand, if T is degenerate, then $\beta_T \leq 2$ and each T_i can be assumed equal to B_i. Thus, in either case we have

$$\frac{\beta_T}{2}\sum_{i=1}^{r} x(E(B_i : T \setminus B_i)) \leq \sum_{i=1}^{k} x(E(T_i : T \setminus T_i)). \tag{3}$$

Finally, note that if T is regular, then $r > k$ and $\beta_T = 1$. Likewise, if T is degenerate then $r = k$ and $\beta_T = k/(k-1)$. Thus, in either case, $\beta_T(2r-2)/2 \geq k$, and

$$\frac{\beta_T}{2}\left(\sum_{i=1}^{r} x(\delta(B_i)) - 2\right) \geq \frac{\beta_T}{2}(2r - 2) \geq k \tag{4}$$

Putting together (2), (3), and (4) we get the desired result. $\square$

Definition 2. *Consider a family of teeth $\mathcal{T}$ and a family of handles $\mathcal{H}$, where each $T \in \mathcal{T}$ is a $\kappa(T)$-domino (with $\kappa(T) \leq |\mathcal{H}|$)and each $H \in \mathcal{H}$ is a proper subset of V. We say that Λ defines a* proper tooth-handle relationship *on $\mathcal{T}$ and $\mathcal{H}$ if we have the following (symmetric) associations. Each tooth $T \in \mathcal{T}$ is associated with exactly $\kappa(T)$ handles $H \in \mathcal{H}$, call this set $\Lambda(T)$, and each handle $H \in \mathcal{H}$ is associated with an odd number of dominoes $T \in \mathcal{T}$, call this set $\Lambda(H)$. For ease of notation, we index the halves of T according to the handle to which they are associated, that is, the halves of T are labeled $\{T_H\}_{H\in\Lambda(T)}$.*

Definition 3. *Let $\mathcal{F}= \{E_1, E_2, \dots, E_k\}$, where $E_i \subseteq E$ for all $i \in I_k$, and define $\mu_e := |\{F \in \mathcal{F} : e \in F\}|$ for each $e \in E$. Following Letchford [13], the family $\mathcal{F}$ is said to* support *the cut $\delta(H)$ if $\delta(H) = \{e \in E : \mu_e$ is odd$\}$.*

Theorem 1. *Suppose that Λ defines a proper tooth-handle relationship on $\mathcal{T}$ and $\mathcal{H}$. For each $H \in \mathcal{H}$ let $F_H \subseteq E$ be such that $\{F_H, \{E(T_H : T \setminus T_H)\}_{T\in\Lambda(H)}\}$ supports the cut $\delta(H)$ in G and define μ^H accordingly. Then the inequality*

$$\sum_{H\in\mathcal{H}} \mu^H x + \sum_{T\in\mathcal{T}} \beta_T x(\delta(T)) \geq \sum_{H\in\mathcal{H}} |\Lambda(H)| + 2\sum_{T\in\mathcal{T}} \beta_T + |\mathcal{H}| \tag{5}$$

is satisfied by all tours.

Proof. We use induction on $|\mathcal{H}|$, the case $|\mathcal{H}| = 0$ following from the validity of the subtour constraints. Let x^c be the incidence vector of a tour. If there exists $H_o \in \mathcal{H}$ such that $\mu^{H_o}x^c > |\Lambda(H_o)| - 1$, then, since $\mu^{H_o}x^c$ is even valued (see Letchford [13]), we have $\mu^{H_o}x^c \geq |\Lambda(H_o)| + 1$. Note also that for each $T \in \Lambda(H_o)$ the family $\{T_H : H \in \Lambda(T) \setminus H_o; T\}$ defines a regular $(|\Lambda(T)| - 1)-$domino. Thus, by induction, the inequality obtained by removing H_o and redefining $\beta'_T = \beta_T$ for $T \in \mathcal{T} \setminus \Lambda(H_o)$ and $\beta'_T = 1$ for $T \in \Lambda(H_o)$

$$\sum_{H\in\mathcal{H}\setminus H_o} \mu^H x + \sum_{T\in\mathcal{T}} \beta'_T x(\delta(T)) \geq \sum_{H\in\mathcal{H}\setminus H_o} |\Lambda(H)| + (|\mathcal{H}| - 1) + 2\sum_{T\in\mathcal{T}} \beta'_T$$

is valid. Then (5) follows since $(\beta_T - \beta'_T)x^c(\delta(T)) \geq (\beta_T - \beta'_T)2$, and $\mu^{H_o}x^c \geq |\Lambda(H_o)| + 1$.

So we can now assume that $\mu^H x^c \leq |\Lambda(H)| - 1$ for each $H \in \mathcal{H}$. From Lemma 1 we have for each $T \in \mathcal{T}$

$$\beta_T(x^c(\delta(T)) - 2) \geq 2|\Lambda(T)| - 2\sum_{H\in\Lambda(T)} x^c(E(T_H : T \setminus T_H)).$$

Hence,

$$\begin{aligned}
\sum_{T\in\mathcal{T}} \beta_T(x^c(\delta(T)) - 2) &\geq 2\sum_{T\in\mathcal{T}} |\Lambda(T)| - 2\sum_{T\in\mathcal{T}}\sum_{H\in\Lambda(T)} x^c(E(T_H : T \setminus T_H)) \\
&\geq 2\sum_{H\in\mathcal{H}} |\Lambda(H)| - 2\sum_{H\in\mathcal{H}} \mu^H x^c \\
&= \sum_{H\in\mathcal{H}} |\Lambda(H)| + \sum_{H\in\mathcal{H}} \big(|\Lambda(H)| - \mu^H x^c\big) - \sum_{H\in\mathcal{H}} \mu^H x^c \\
&\geq \sum_{H\in\mathcal{H}} |\Lambda(H)| + |\mathcal{H}| - \sum_{H\in\mathcal{H}} \mu^H x^c.
\end{aligned}$$

□

We refer to the constraints (5) as *k-parity inequalities*, when $|\mathcal{H}| = k$. When $k = 1$ this class is precisely the *domino-parity inequalities* of Letchford [13]. It is easy to see that not all k-parity inequalities define facets of the TSP polytope, but the class does provide a common framework for possibly extending Letchford's algorithm to superclasses of other inequalities that have proven to be effective in TSP codes. In particular, k-parity inequalities generalize clique-tree inequalities (Grötschel and Pulleyblank [10]) in the same way as domino-parity inequalities generalize combs.

Definition 4. *Families $\mathcal{H}$ and $\mathcal{T}$ are said to define a* clique-tree *if:*
(i) $\mathcal{H}$ is a family of pairwise disjoint proper subsets of V.
(ii) $\mathcal{T}$ is a family of pairwise disjoint proper subsets of V.
(iii) No $T \in \mathcal{T}$ is contained in $\bigcup(H : H \in \mathcal{H})$.
(iv) For each $H \in \mathcal{H}$ let $\Lambda(H) = \{T : T \cap H \neq \emptyset\}$. $|\Lambda(H)|$ must be odd.
(v) The intersection graph defined by the families $\mathcal{H}$ and $\mathcal{T}$ is a tree.

In this context, the sets $H \in \mathcal{H}$ are called handles and the sets $T \in \mathcal{T}$ are called teeth. If the intersection graph defined by the families $\mathcal{H}$ and $\mathcal{T}$ is a forest, we say that $\mathcal{H}$ and $\mathcal{T}$ define a clique-forest. Note that if $\mathcal{H}$ and $\mathcal{T}$ define a clique-forest, then it is possible to define a $|\mathcal{H}|$-parity constraint as follows. For each $T \in \mathcal{T}$ define $\Lambda(T) = \{H \in \mathcal{H} : T \in \Lambda(H)\}$, and $T_H = T \cap H$, $\forall H \in \Lambda(T)$. Clearly $(T_H : H \in \Lambda(T); T)$ defines a $|\Lambda(T)|$-domino and $(\mathcal{H}, \mathcal{T}, \Lambda)$ defines a proper tooth-handle relationship. Thus, Theorem 1 implies that the well-known clique-tree (forest) constraint is valid,

$$\sum_{H \in \mathcal{H}} x(\delta(H)) + \sum_{T \in \mathcal{T}} x(\delta(T)) \geq 2|\mathcal{T}| + |\mathcal{H}| + \sum_{H \in \mathcal{H}} |\Lambda(H)|$$

where in the case of clique-trees, $\sum(|\Lambda(H)| : H \in \mathcal{H})$ is commonly written as $|\mathcal{T}| + |\mathcal{H}| - 1$. Clique-tree inequalities generalize combs inequalities, which are clique trees having a single handle.

We will focus on special cases of clique trees in the next section, but we would like to point out that k-parity inequalities also generalize several other well-known classes of TSP constraints.

Proposition 1. *The family of $k-$parity inequalities generalizes the family of bipartition inequalities and the family of star inequalities.*

3 Planar separation with multiple handles

Throughout this section we assume that the LP solution x^* satisfies all subtour constraints. Also, for any set $F \subseteq E$, we define $F^* = \{e \in F : x^*_e > 0\}$.

Definition 5. *For a given $x^* \in SEP(n)$, We say that a k-domino $(T_1, \ldots, T_k; T)$ is* super-connected *if:*
(i) T and $V \setminus T$ are connected in G^*.
(ii) T_i and $T \setminus T_i$ are connected in G^* for all $i \in I_k$.
(iii) $x^*(E(T_i : V \setminus T)) > 0$ and $x^*(E(T \setminus T_i : V \setminus T)) > 0$ for all $i \in I_k$.
We say that a k-parity constraint having teeth $\mathcal{T}$ is super-connected, if every tooth $T \in \mathcal{T}$ is super-connected.

While as of yet it is an open problem whether or not the class of k-parity inequalities can be separated in polynomial time, we extend the ideas of Letchford [13] so as to separate, for fixed k, a subclass of k-parity inequalities which contains all super-connected clique-trees with k handles or less, under the assumption that the support graph G^* is planar.

For this we proceed in three steps. First, we characterize violated k-parity inequalities. Second, we characterize violated k-parity inequalities under the additional assumptions that the support graph G^* is planar, and that teeth are super-connected. Finally, we outline an algorithm for separating a subclass of k-parity inequalities when G^* is planar; this subclass (defined with respect to an LP solution x^*) contains all super-connected clique-tree inequalities which have k handles or less.

The following two Propositions (the proofs of which are to be included in a future paper) are not used throughout the following sections. However, they serve as a motivation for separating classes of super-connected constraints.

Proposition 2. *Let x^* be an LP solution and consider a violated clique-tree constraint on k handles, having teeth $\mathcal{T}$. Let $(T_1, T_2, \ldots, T_q; T) \in \mathcal{T}$. If all clique-tree constraints having less than k handles are satisfied by x^*, then (a) T is connected, (b) T_i is connected for all $i \in I_q$, (c) $x^*(E(T_i : V \setminus T)) > 0$ and $x^*(E(T \setminus T_i : V \setminus T)) > 0$ for all $i \in I_q$.*

Proposition 3. *If all subtour inequalities are satisfied, then there exists a maximally violated (if any) comb inequality which is super-connected. If all subtour and comb inequalities are satisfied, then there exists a maximally violated (if any) clique-tree inequality on two handles which is super-connected.*

Proposition 2 indicates that when clique-tree inequalities on k handles are satisfied, then all violated clique-trees on $k+1$ handles are *almost* super-connected. Proposition 3 shows that once comb inequalities are effectively separated, we may assume for exact separation purposes that two handled clique-trees are super-connected.

3.1 Characterizations of Violated k-Parity Constraints

Definition 6. *Define the* weight *of k-domino $(T_1, T_2, \ldots, T_k; T)$ to be*

$$w(T) := \beta_T(x(\delta(T)) - 2) + \sum_{i=1}^{k} x(E(T_i : T \setminus T_i)) - k. \tag{6}$$

Lemma 2. *The slack of a k-parity inequality is $\sum_{T \in \mathcal{T}} w(T) + \sum_{H \in \mathcal{H}} x(F_H) - |\mathcal{H}|$.*

Proof. Consider a k-parity inequality defined by $\mathcal{H}$, $\mathcal{T}$, and Λ. The slack is,

$$\begin{aligned}
&\sum_{H \in \mathcal{H}} \mu^H x + \sum_{T \in \mathcal{T}} \beta_T x(\delta(T)) - \sum_{H \in \mathcal{H}} |\Lambda(H)| - 2 \sum_{T \in \mathcal{T}} \beta_T - |\mathcal{H}| \\
&= \sum_{H \in \mathcal{H}} \left(x(F_H) + \sum_{T \in \Lambda(H)} x(E(T_H : T \setminus T_H)) \right) + \sum_{T \in \mathcal{T}} \left(\beta_T \left(x(\delta(T)) - 2 \right) - |\Lambda(T)| \right) - |\mathcal{H}| \\
&= \sum_{H \in \mathcal{H}} x(F_H) + \sum_{T \in \mathcal{T}} \left(\beta_T \left(x(\delta(T)) - 2 \right) + \sum_{H \in \Lambda(T)} x(E(T_H : T \setminus T_H)) - |\Lambda(T)| \right) - |\mathcal{H}| \\
&= \sum_{H \in \mathcal{H}} x(F_H) + \sum_{T \in \mathcal{T}} w(T) - |\mathcal{H}|
\end{aligned}$$

□

Note that Lemma 1 and Lemma 2 together imply that a violated k-parity constraint must satisfy

$$0 \leq \frac{\beta_T}{2}(x(\delta(T)) - 2) \leq w(T) \leq |\mathcal{H}| \qquad \forall T \in \mathcal{T}. \tag{7}$$

Definition 7. *Consider a family of teeth $\mathcal{T}$, where each $T \in \mathcal{T}$ satisfy $\kappa(T) \leq k$. We say that Φ defines an* abstract tooth-handle *relationship over $\mathcal{T}$ and I_k if (i) $\Phi(T) \subseteq I_k$ and $|\Phi(T)| = \kappa(T)$ for all $T \in \mathcal{T}$, (ii) $\Phi(i) \subseteq \mathcal{T}$ and $|\Phi(i)|$ is odd, for all $i \in I_k$, and (iii) $T \in \Phi(i)$ iff $i \in \Phi(T)$ for all $i \in I_k, T \in \mathcal{T}$.*

Lemma 3. *There exists a violated k-parity inequality iff there exist $(\mathcal{T}, \Phi)$ defining an abstract tooth-handle relationship, and sets $R_i \subseteq E^*$ for all $i \in I_k$ such that:*
(i) $\{E^*(T_i : T \setminus T_i)\}_{T \in \Phi(i)}$ and $\{R_i\}$ support a cut in G^* for all $i \in I_k$.
(ii) $\sum_{i \in I_k} x^*(R_i) + \sum_{T \in \mathcal{T}} w(T) - k < 0$.

Proof. From Theorem 1 and Lemma 2, a k-parity inequality is violated iff there exist $(\mathcal{H}, \mathcal{T}, \Lambda)$ defining a proper tooth-handle relationship, and sets $F_H \subseteq E$ for $H \in \mathcal{H}$ such that:
(a) $\{E(T_H : T \setminus T_H)\}_{T \in \Lambda(H)}$ and $\{F_H\}$ support the cut $\delta(H)$ in G for all $H \in \mathcal{H}$
(b) $\sum_{H \in \mathcal{H}} x^*(F_H) + \sum_{T \in \mathcal{T}} w(T) - |\mathcal{H}| < 0$

We first prove necessity. Assume that $(\mathcal{H}, \mathcal{T}, \Lambda)$ defines a violated k-parity inequality. We know that there exists $F_H \subseteq E$ for $H \in \mathcal{H}$ satisfying (a)-(b). Assume $\mathcal{H} = \{H_i : i \in I_k\}$. For each $i \in I_k$ define $\Phi(i) = \Lambda(H_i)$, $\Phi(T) = \{i : H_i \in \Lambda(T)\}$ and $R_i = F_{H_i} \cap E^*$. Note that Φ and $\mathcal{T}$ define an abstract tooth-handle relationship, and $|\mathcal{H}| = k$. Hence, conditions (a)-(b) imply (i)-(ii).

We next prove sufficiency. Assume that $\mathcal{T}, \Phi$ define an abstract tooth-handle relationship, and sets $R_i \subseteq E^*$, $i \in I_k$ are such that (i) and (ii) hold. For each $i \in I_k$ let $H_i \subseteq V$ be one shore of the cut supported by $\{E^*(T_i : T \setminus T_i)\}_{T \in \Phi(i)}$ and R_i, and let $\Lambda(H_i) = \Phi(i)$. Likewise, for $T \in \mathcal{T}$ define $\Lambda(T) = \{H_i : i \in \Phi(T)\}$. Note that Λ define a proper tooth-handle relationship on $\mathcal{T}, \mathcal{H}$. Define $F_{H_i} \subseteq R_i \cup \{e \in \delta(H_i) : x_e = 0\}$ such that (a) holds. Thus (b) must also hold. □

For the remainder of this section, assume that G^* is a planar graph and let $\bar{G^*}$ denote the planar dual of G^*. For any subset $F \subseteq E(G^*)$, denote by $\bar{F}$ the corresponding edges in $\bar{G^*}$. For each $\bar{e} \in \bar{G^*}$ let $x^*_{\bar{e}} = x^*_e$.

Definition 8. *A graph H is called* Eulerian *if every node has even degree. (As in Letchford [13], we do not require that H be connected.)*

Definition 9. *Let r be a positive integer and suppose that $E_1, \ldots, E_r$ are edge-sets satisfying $E_i \subseteq E^*$, $i \in I_r$. The collection $\{\bar{E}_i : i \in I_r\}$ is said to* support an Eulerian subgraph *in $\bar{G^*}$ if the edges $\bar{e}$ for which μ_e is odd form an Eulerian subgraph in $\bar{G^*}$.*

This definition implies that $\{\bar{E}_i : i \in I_r\}$ supports an Eulerian subgraph in $\bar{G^*}$ iff $\{E_i : i \in I_r\}$ supports a cut in G^*. Hence we have the the following dual version of Lemma 3.

Lemma 4. *A k-parity inequality is violated iff there exist $\mathcal{T}, \Phi$ defining an abstract tooth-handle relationship, and sets $\bar{R}_i \subseteq \bar{E}^*$ for $i \in I_k$ such that:*
(i) $\{\overline{E^*(T_i : T \setminus T_i)}\}_{T \in \Phi(i)}$ and $\{\bar{R}_i\}$ support an Eulerian subgraph in $\bar{G^*}$ for all $i \in I_k$.
(ii) $\sum_{i=1}^{k} x^*(\bar{R}_i) + \sum_{T \in \mathcal{T}} w(T) - k < 0$

Proof. Follows from the definitions. □

Lemma 5. *A k-domino $(T_i : i \in I_k; T)$ is super-connected iff (a) $C(T) = \overline{\delta^*(T)}$ is a simple cycle in $\bar{G^*}$, (b) for each $i \in I_k$ the edges $P_i(T) = \overline{E^*(T_i : T \setminus T_i)}$ define a simple path in $\bar{G^*}$ with end-points $\{s_i^T, t_i^T\}$ in $C(T)$ (where $s_i^T \neq t_i^T$) and all other nodes not in $C(T)$, and (c) all of the paths $P_i(T)$ are in the same side of the cycle C with respect to the planar embedding.*

Proof. First note that for any set $A \subsetneq V$ both A and $V \setminus A$ are connected if and only if $\overline{\delta^*(A)}$ is a simple cycle in $\bar{G^*}$. Thus condition (i) in Definition 5 is equivalent to condition (a) in the Lemma.

Now we prove that conditions (i)-(iii) of Definition 5 imply conditions (b)-(c) of the Lemma.

Assume that for $p, q \in I_k$ we have that some edge $\bar{e}_p \in P_p(T)$ and some edge $\bar{e}_q \in P_q(T)$ are such that $\bar{e}_p, \bar{e}_q$ are on different sides of C with regard to the embedding. Then, the end-points of e_p and e_q are separated by $\delta^*(T)$. Since this can't happen, because T is connected, it follows that the sets $P_i(T)$ for $i \in I_k$ are all on the same side of C. Thus we obtain condition (c) on Lemma 5.

From condition (ii) on Definition 5 it follows that T_i is connected. From (i), (ii), and (iii) on Definition 5 it follows that $T \setminus T_i$ and $V \setminus T$ are connected, and $E^*(T \setminus T_i : V \setminus T) \neq \emptyset$, hence $V \setminus T_i$ is connected. Thus $\overline{\delta^*(T_i)}$ is a simple cycle. Since $x^*(E(T_i : V \setminus T)) > 0$ it follows that $\overline{\delta^*(T_i)}$ intersects $\overline{\delta^*(T)}$ in at least one edge. Since $P_i(T)$ is $\overline{\delta^*(T_i)} \setminus \overline{\delta^*(T)}$, it must be a node-disjoint union of simple paths with end-points in C. If it is the union of two or more such paths then note that these paths must be on the same side of C, and must divide this side into at least three parts - meaning that either T_i or $T \setminus T_i$ is disconnected in G^*. Thus P_i must be a single path, and (b) follows.

Next we prove that conditions (a)-(c) imply conditions (ii)-(iii) of Definition 5.

Since $\overline{\delta^*(T_i)} = E^*(T_i : T \setminus T_i) \cup E^*(T_i : V \setminus T)$ it follows that $\delta^*(T_i) \subseteq (\delta^*(T) \cup \overline{E^*(T_i : T \setminus T_i)})$. However, $\overline{\delta^*(T_i)}$ must be an Eulerian subgraph in $\bar{G^*}$, and the only Eulerian subgraphs contained in $\overline{\delta^*(T)} \cup \overline{E^*(T_i : T \setminus T_i)}$ are simple cycles. Thus, $\overline{\delta^*(T_i)}$ is a simple cycle and T_i is connected. To prove that $T \setminus T_i$ is connected is analogous, and so (ii) follows. If $x^*(E(T_i : V \setminus T)) = 0$ then $E^*(T_i : T \setminus T_i) = \delta(T_i)$. Hence, $\overline{E^*(T_i : T \setminus T_i)}$ would be an Eulerian subgraph and not a simple path. If $x^*(E(T \setminus T_i : V \setminus T)) = 0$ then $E^*(T_i : T \setminus T_i) = \delta(T \setminus T_i)$. Hence, $\overline{E^*(T_i : T \setminus T_i)}$ would be an Eulerian subgraph and not a simple path. Thus (iii) follows. □

Definition 10. *Consider two distinct super-connected k-dominoes T and L. If the end-points of $P_i(T)$ and $P_i(L)$ are the same for $i \in I_k$ and $w(T) < w(L)$ we say that T* dominates *L.*

Lemma 6. *Consider two distinct super-connected k-dominoes T and L. If T dominates L, and if L is used in some violated k-parity constraint, then L may be replaced by T to obtain another violated k-parity constraint has less slack.*

Proof. By removing each path $P_i(L)$ and replacing it with $P_i(T)$ for $i \in I_k$, condition (i) of Lemma 4 is not changed, and $w(T) < w(L)$ implies condition (ii) is not changed - in fact, the violation, given by (ii), will improve from the substitution.

From Lemma 6 it follows that a maximally violated super-connected k-parity constraint will only have non-dominated teeth.

3.2 Separating Super-Connected Clique Tree Constraints

Given a fixed $k \in \mathbb{Z}_+$ and a fractional LP solution x^* satisfying all subtour constraints, the algorithm proceeds in two steps. First, a minimal family of non-dominated teeth is generated. Next, a violated super-connected k-handle constraint is generated (if such exists) by solving an odd Eulerian subgraph problem in an appropriate graph.

In order to describe the tooth generation procedure, it is important to establish two results.

Lemma 7. *Every tooth T in a violated k-handle clique-tree constraint must satisfy*

$$2 \leq x^*(\delta(T)) < 2(k+1) \tag{8}$$

Proof. Follows from (7), the subtour constraints, and the fact that clique-trees have no degenerate teeth.

Lemma 8. *Consider a violated super-connected k-handled clique-tree constraint with tooth set $\mathcal{T}$. Let $(T_i : i \in I_{\kappa(T)}; T)$ be a $\kappa(T)$-domino in $\mathcal{T}$, and define P_i to be the path $\overline{E(T_i : T \setminus T_i)}$ for all $i \in I_{\kappa(T)}$. Then,*
(i) Paths P_i and P_j don't cross with regards to the dual embedding, for $i \neq j \in I_{\kappa(T)}$.
(ii) Paths P_i and P_j can't have the same end-points, unless $\kappa(T) = 2$ for $i \neq j \in I_{\kappa(T)}$.
(iii) If paths P_i and P_j have the same end-points, then $P_i \neq P_j$, for $i \neq j \in I_{\kappa(T)}$.

Proof. For (i) If paths P_i and P_j cross, then halves T_i and T_j must intersect. For (ii) assume that P_i and P_j have the same end-points. If there exists a path P_k with $k \neq i, j$, given that it can't intersect paths P_i or P_j, it must either run between P_i and P_j, or must run the side of either P_i or P_j. In either case, this implies that T_k intersects T_i or T_j. For (iii) if P_i and P_j have the same end-points and the paths coincide, then the tooth must be degenerate. However, this is contradictory with the definition of clique-trees. □

Lemma 7 and Lemma 8 suggest a natural algorithm by which to enumerate a minimal set of teeth for a violated super-connected k handle clique-tree constraint. First, enumerate all connected sets $T \subseteq V$ which satisfy condition (8) using an algorithm such as that of Nagamochi et. al [15]. Keep those sets T for which $V \setminus T$ is connected. Let $C = \overline{\delta(T)}$. Choose a side of C with regard to the planar embedding, and let W represent the nodes of that side minus the nodes in C. Next, for each pair of nodes $u, v \in C$ compute the shortest path and second-shortest path from u to v in W. Choose $q \in I_k$ and a set of end-points $\{(s_i, t_i) : i \in I_q\}$ in C. Check that that no two pairs of end-points are crossing (that is, such that it is impossible to take a path from s_i to t_i without crossing a path from s_j to t_j). If $q = 2$ and $s_1 = s_2 = s$, $t_1 = t_2 = t$, let P_1 be the shortest s to t path, and let P_2 be the second-shortest s to t path. Otherwise, define $\hat{P}_i$ as the shortest s_i to

t_i path for $i \in I_q$. If the paths $\hat{P}_1, \ldots, \hat{P}_q$ cross each other, un-cross them so as to define paths $P_i,\ i \in I_q$. At this stage, C and the paths $P_i,\ i \in I_q$, define a q-domino. If the weight is larger than k, or, if there is another q-domino which dominates it, discard the tooth. Keep iterating until all possible combinations of end-points, sides of the cycle, and sets T have been exhausted. It is not difficult to see that this algorithm is polynomial, and that it enumerates a minimal set of non-dominated teeth (which is polynomially sized).

For the specific case in which $k = 1$, a faster tooth generation procedure is presented in Letchford [13]. First, if $k = 1$, it is shown that a tooth T is super-connected iff $\overline{\delta(T)}$ and $\overline{E(T_1 : T \setminus T_1)}$ define three node-disjoint paths in $\bar{G}^*$. In order to construct the teeth, a network $\mathcal{N}$ is constructed from the graph $\bar{G}^*$ so that the nodes of $\mathcal{N}$ and $\bar{G}^*$ coincide. Then, for each edge in $\bar{G}^*$, two arcs (one in each direction) of capacity one are added to $\mathcal{N}$. By solving the min-cost three-unit flow problem between each pair of nodes in $\mathcal{N}$, it is possible to generate a minimal set of non-dominated teeth. The fact that paths in a solution may possibly cross is not a problem, for it is shown that if an optimal solution is crossing, then for the given pair of nodes there can be no tooth satisfying condition (7). In our implementation of Letchford's algorithm we use this idea, which will be further discussed in Section 4. This brings us to our main result.

Theorem 2. *Suppose G^* is planar and x^* satisfies all subtour constraints. Consider a fixed integer $k \geq 1$. It is possible to separate in polynomial time a subclass of k-parity constraints which contains all violated super-connected clique-tree inequalities on k handles.*

The proof of this theorem consists of two parts. First we outline a two-stage algorithm which runs in polynomial time, and then we prove that the algorithm separates a subclass of k-parity constraints which contains all violated super-connected clique-tree inequalities.

The two steps of the algorithm are as follows:

(i) Construct a minimal non-dominated family of teeth $\mathcal{L}$.

(ii) Construct a graph $M[k]$ using $\mathcal{L}$ and $\bar{G}^*$. Solve the min-weight 1^k-Eulerian Subgraph problem in $M[k]$.

For a definition of the min-weight 1^k-Eulerian Subgraph problem, see Appendix A. If we obtain a solution in step (ii) having weight less than k, then we have found a violated k-parity constraint. The intuition of the algorithm is as follows: From Lemma 4 we know that a violated k-parity inequality can be characterized by a set of Eulerian-subgraphs of G^*, one for each handle, and each utilizing an odd number of teeth. For every path $P_i(T) = \overline{E^*(T_i : T \setminus T_i)}$ define an *odd* edge whose end-points coincide with the end-points of $P_i(T)$, and whose weight coincides with the weight of the tooth. Thus, the problem can be modeled as that of searching for a set of odd Eulerian subgraphs (those that use an odd number of odd edges), one for each handle, whose combined weight is minimized, subject to side constraints (defined by the teeth) which link these subgraphs to each other. The side constraints would impose that either all of the paths associated to a tooth are used (in different handles), or none at all. The proposed algorithm works by defining a graph $M[k]$ which contains the Cartesian product of k copies of $\bar{G}^*$; the idea being that any path in $M[k]$ corresponds to k individual paths in $\bar{G}^*$, one in each of the components (or layers) which make up the Cartesian product. By defining special edges in $M[k]$ associated to teeth in $\mathcal{L}$, it is possible to associate certain Eulerian subgraphs in $M[k]$ to k-parity inequalities defined in $\bar{G}^*$. Note that $M[1]$ coincides with the graph M^* as defined in Letchford [13]; in this case Letchford proved that the condition of being Eulerian can be replaced by the condition of being a simple cycle.

As we have already discussed, Step 1 can be performed in polynomial time. thus, we concentrate on the second step of the algorithm.

Definition 11. *Let $x^* \in SEP(n)$, and let $\mathcal{L}$ be a family of super-connected teeth, each with at most k halves. We define $M[k]$ in terms of G^* and $\mathcal{L}$, as an undirected graph, and functions $p : E(M[k]) \rightarrow \{0,1\}^k$, $w : E(M[k]) \rightarrow \mathbb{R}_+$, where:*

(i) $V(M[k]) = (\bar{V}^*)^k$.

(ii) For each $\hat{e} \in \bar{E}^*$, $i \in i_k$, and $a \in (\bar{V}^*)^k$, define $e = e[\hat{e}, i, a] = (u, v) \in E(M[k])$ such that $e_j := (u_j, v_j) = (a_j, a_j)$ if $j \neq i$, and $e_i := (u_i, v_i) = \hat{e}$. Define $w(e[\hat{e}, i, a]) = x^*_{\hat{e}}$ and $p(e[\hat{e}, i, a]) = 0^k$.

(iii) For each $T \in \mathcal{L}$, $a \in (\bar{V}^*)^k$, and $J = \{j_i : i \in i_{\kappa(T)}\}$ an ordered subset of I_k, define $e = e[T, J, a] = (u, v) \in e(M[k])$ such that $e_j := (u_j, v_j) = (a_j, a_j)$ if $j \notin J$, and $e_{j_i} := (u_{j_i}, v_{j_i}) = (s_i^T, t_i^T)$ if $j_i \in J$. Define $w(e[T, J, a]) = w(T)$ and $p(e[T, J, a])_j = 1$ if $j \in J$ and $p(e[T, J, a])_j = 0$ if $j \notin J$.

Lemma 9. *The size of $M[k]$ is polynomially bounded in the size of G^* and $\mathcal{L}$.*

Proof. (i) $|V(M[k])| = |\bar{V}^*|^k$.
(ii) $|E(M[k])| = k\,|\bar{E}^*|\,|\bar{V}^*|^k + \sum\left(\binom{k}{i}i!|\bar{V}^*|^k 2^{i-1}|\{T \in \mathcal{L} : \kappa(T) = i\}| : i \in i_k\right)$.
(iii) $|\{T \in \mathcal{L} : \kappa(T) = i\}| \leq \binom{|\bar{V}^*|}{2}^i$.

$\square$

Note that the problem of finding a minimum weight Eulerian subgraph of $M[k]$ having parity $p \in \{0,1\}^k$ when $w \geq 0$ can be solved in polynomial time on the size of $M[k]$ (see appendix A). Now we will prove that an optimal solution to the 1^k-parity problem in $M[k]$ represents a k-parity constraint with the appropriate violation.

Lemma 10. *Let M^* be an optimal solution to the minimum weight 1^k-parity Eulerian subgraph problem in $M[k]$. If $w(M^*) < k$ then there exists a violated $k-$parity inequality for x^* with violation at least $w(M^*) - k$.*

Proof. We proceed by constructing a set $\mathcal{T} \subseteq \mathcal{L}$ of teeth and an abstract tooth-handle relationship Φ on $\mathcal{T}$ which satisfy the conditions of Lemma 3.

Let $\mathcal{T} := \{T \in \mathcal{L} : e[T, s, a] \in M^*\}$. Observe that $\mathcal{T}$ is a collection, and as such may have repeated teeth. Despite this, note then that for all $T \in \mathcal{T}$ there exist associated $S_T \subseteq I_k$ and $a_T \in (\bar{V}^*)^k$ such that $e[T, S_T, a_T] \in M^*$ and $|S_T| = \kappa(T)$. For each $T \in \mathcal{T}$ define $\Phi(T) = S_T$. Likewise, define $\Phi(i) = \{T \in \mathcal{T} : i \in S_T\}$. Since $p(M^*)_i = 1$ we have that $|\Phi(i)|$ is odd for all $i \in I_k$. Thus Φ defines an abstract-tooth handle relationship on $\mathcal{T}$.

Let $\bar{R}_i := \{\hat{e} \in \bar{E}^* :$ such that there are an odd number of edges $e[\hat{e}, i, a] \in M^*\}$. For $T \in \Phi(i)$ let $\hat{\imath}$ be such that $i = j_{\hat{\imath}} \in S_T$. Then, by definition of $M[k]$ and M^*, we have that $\{P_{\hat{\imath}}(T)\}_{T \in \phi(i)}$ and $\bar{R}_i$ support an Eulerian subgraph in $\bar{G}^*$.

Finally, note that $\sum(x^*(\bar{R}_i) : i \in i_k) + \sum(w(T) : T \in \mathcal{T}) \leq w(M^*) < k$, Thus, the conditions of Lemma 3 hold and we conclude the proof. $\square$

To finish the proof of Theorem 2 the only missing step is to prove that each maximally violated clique-tree inequality $(\mathcal{H}, \mathcal{T})$ on k handles can be represented in $M[k]$ as an Eulerian subgraph M^* of parity 1^k and weight $w(M^*) = \sum(x(R_H) : H \in \mathcal{H}) + \sum(w(T) : T \in \mathcal{T})$. The proof given will be inductive, and makes use of the following Lemma.

Lemma 11. *Given a clique-tree constraint on k handles $(\mathcal{H}, \mathcal{T}, \Lambda)$, there exists $H_o \in \mathcal{H}$ such that it shares at most one tooth with the other handles, i.e. $|\Lambda(H_o) \cap (\bigcup(\Lambda(H) : H \in \mathcal{H} \setminus H_o))| <= 1$.*

Proof. Follows directly from the tree-structure of the intersection graph. $\square$

If $k = 1$, then consider the subgraph of $M[1]$ with edges $\overline{\bigcup(E(T_H : T \setminus T_H)} : T \in \Lambda(H))$ and F_H. Note that by Lemma 3 this graph is Eulerian. Replace each path $\overline{E(T_H : T \setminus T_H)}$ with $e[T, \{H\}, a]$, where $a \in \bar{V}^*$ is arbitrary, and fixed. Call this subgraph M^*. Clearly M^* is also Eulerian. Moreover $p(M^*) = |\mathcal{T}| \mod 2 = 1$ and $w(M^*) = x(F_H) + \sum(w(T) : T \in \mathcal{T})$. This concludes the proof for the case $k = 1$.

Now, assume that the theorem holds on super-connected clique-trees with up to k handles, and let $(\mathcal{H}, \mathcal{T}, \Lambda)$ be a super-connected clique-tree with $k+1$ handles. By Lemma 11 there exists $H_o \in \mathcal{H}$ such that $|\Lambda(H_o) \cap (\bigcup(\Lambda(H) : H \in \mathcal{H} \setminus H_o))| <= 1$. Consider now $(\hat{\mathcal{H}}, \hat{\mathcal{T}}, \hat{\Lambda})$ to be the super-connected clique-tree inequality we obtain after removing handle H_o from $(\mathcal{H}, \mathcal{T}, \Lambda)$. That is, remove H_o, all the 1-dominoes in $\Lambda(H_o)$, and eliminate the half T_{H_o} from the shared tooth, if any. Let M^k be the Eulerian subgraph representing $(\hat{\mathcal{H}}, \hat{\mathcal{T}}, \hat{\Lambda})$ in $M[k]$ and let M^1 be the Eulerian subgraph representing $(H_o, \Lambda(H_o), \Lambda_{H_o})$ in $M[1]$.

If H_o doesn't share any dominoes with the rest of the original inequality, it is enough to take any $a \in \bar{V}^*$ and define M^* in $M[k+1]$ as $M^* = \hat{M}^1 \cup \hat{M}^k$ where $\hat{M}^1$ and $\hat{M}^k$ are obtained from M^1 and M^k by extending their dimension so that they fit in $M[k+1]$. To do this, define $(\hat{M}^k)_j = M^k_j$ for $j \in I_k$ and $(\hat{M}^k)_{k+1} = \{a, \dots, a\}$; also define $(\hat{M}^1)_{k+1} = M^1$ and $(\hat{M}^1)_j = \{a, \dots, a\}$ for $j \in I_k$. It follows that

$w(M^*) = w(M^1) + w(M^k) = \sum(x(F_H) : H \in \mathcal{H}) + \sum(w(T) : T \in \mathcal{T})$ and $p(M^*)_j = p(M^1)_j + p(M^k)_j \mod 2 = 1$ for all $j \in I_{k+1}$. Thus we conclude this case.

Assume now that H_o does share a domino $\hat{T}$ with the rest of the inequality. Assume also that both M^1 and M^k are connected. Let $e_1 = (u_1, v_1) = e[\hat{T}, S_1, a_1]$ be the edge representing $\hat{T}$ in M^k and let $e_2 = (u_2, v_2) = e[\hat{T}, H_o, a_2]$ be the edge representing $\hat{T}$ in M^1. Since both graphs are connected and Eulerian we may assume that both M^k and M^1 are closed walks and can be written as $M^k = \{v_1 \to u_1 \to e_1 \to v_1\}$ and $M^1 = \{v_2 \to u_2 \to e_2 \to v_2\}$. Thus we can define $e = ((u_1, u_2), (v_1, v_2)) = e[\hat{T}, S_1 \cup \{k+1\}, (a_1, a_2)] \in E(M[k+1])$ and define the Eulerian subgraph

$$M^* = \left\{ \begin{array}{l} v_1 \overset{M^k}{\to} u_1 \to e_1 \to v_1 \;\to\; v_1 \\ u_2 \;\to\; u_2 \to e_2 \to v_2 \overset{M^1}{\to} u_2 \end{array} \right\}.$$

Note that by the same arguments given above, this subgraph has the correct weight and parity. If any of the Eulerian graphs M^1, M^k are not connected, then for each of these graphs, select the connected component which has the shared tooth. For those components, repeat the procedure used in the previous case. For the components not sharing a tooth, repeat the first procedure of appending a constant node for the missing components in M^*. The parity for M^* is just the sum of the parities of M^1 and M^k, and the weight is the desired value. This completes the proof of Theorem 2.

4 Implementation and Computational Results

In this section we briefly describe our computational tests. First we discuss the implementation of Letchford's algorithm for separating domino-parity constraints, emphasizing the techniques we adopted to improve its practical performance, and presenting some computational results. Next, we discuss the implementation of a simple heuristic for separating 2-parity constraints.

4.1 Domino-Parity Constraints

Domino Searching Teeth were generated by using the network flow approach described in Section 3.2 and Letchford [13]. To find the min-weight node-disjoint paths between pairs $s, t \in V(\bar{G}^*)$ we used the augmenting-shortest-path network-flow algorithm (See Ahuja et al [3] for details). For this, we build a network $\mathcal{N}$ for the graph $\bar{G}^*$ defining two arcs for each edge (one in each direction), assigning a capacity of one to each. This algorithm computes the $s-t$ flow by solving three successive $s-t$ shortest path problems on reduced capacity networks successively derived from $\mathcal{N}$. Using this algorithm several speed-ups were possible. Firstly, for fixed $s \in V(\bar{G}^*)$ the first $s-t$ flow for all nodes t can be obtained by solving a single Dijkstra algorithm in $\mathcal{N}$ rooted at s. The additional shortest-path computations need only be computed for nodes t at distance not greater than 4/3 from s. Finally, when computing the $s-t$ three-flow in $\mathcal{N}$, one only need consider intermediary nodes at distance not greater than 2 from s and t. This follows from the fact that every cycle in $\bar{G}^*$ corresponds to a cut in G^*, and hence, has weight at least 2 (due to subtour constraints). Thus, if a node is used which has distance at least 2, since the other two paths will define a cycle, the bound of 4 would be exceeded. A useful heuristic idea is to further restrict the set of intermediary nodes to those of distance no greater than 2α for some $\alpha < 1$; this restriction can cause the algorithm to miss violated DP-cuts, but it greatly improves the speed and appears to work well in practice (we have set $\alpha = .55$ in our tests).

Parallelization Dominoes may be computed in parallel. In fact, one may divide the nodes $s \in V(\bar{G}^*)$ among different machines so that each one computes all of the (s, t) three-disjoint paths. We found the domino-computation stage to be (by far) the most time consuming part of the algorithm, making this parallelization crucial for obtaining acceptable running times on large instances when using $\alpha = 1$. Our parallel implementation is a master–worker system based on message passing.

Random Walk The algorithm as formally defined in Letchford [13] computes exactly one constraint. In practice, one would like the algorithm to compute as many violated constraints as possible. To achieve this, instead of just solving the shortest odd-cycle problem in M^* we additionally run a random walk algorithm that attempts to find small-weight odd cycles. This algorithm is fast, easy to implement, and in our tests generally produced a large number of additional cuts, only the best of which were kept.

Safe Shrinking The size of the graph G^* has a dramatic impact on the running time of our implementation. Following the work of Padberg and Rinaldi [18], we attempt to reduce the size of G^* by contracting edges in G^*, redefining the vector x^*, and solving the separation problem in the new, smaller, graph. In this *shrinking* process, a contraction is called *safe* if we know that the existence of a violated DP-inequality implies the existence of one in the graph we obtain after the contraction. Although it is not always the case that shrinking is safe, it is possible to give conditions under which it will be.

Theorem 3. *Consider x^* satisfying all subtour constraints, a DP-inequality $ax \leq b$ satisfying $ax^* > b$, and nodes $u, v, t \in V(G^*)$ such that $x^*_{uv} = 1$, and $x^*_{ut} + x^*_{vt} = 1$. If $a_{u,v} \neq 0$ there exists another DP-inequality $a'x \leq b'$ such that $a'_{u,v} = 0$ and $(a'x^* - b') \geq (ax^* - b)$. Thus, we can contract edge $\{u, v\}$ and ensure the existence of a maximally violated DP inequality with zero $\{u, v\}$ coefficient.*

As a pre-processor to our implementation, we repeatedly contract edges $\{u, v\}$ while there exist nodes u, v, t satisfying the conditions of Theorem 3.

Planarity The safe-shrinking procedure can greatly reduce the size of the graph over which we work, but if the original graph G^* is non-planar then the shrunk may too be non-planar. If this is the case, our implementation does non-safe shrinks until a planar graph is obtained, as in Boyd et al. [4]. If G^* is not planar, we identify a forbidden $K_{3,3}$ or K_5 minor $M \subseteq E(G^*)$. We then take two nodes in the minor with degree at least 3 and contract them (and thus eliminating the minor), iterating until a planar graph is obtained. An alternative is to eliminate an edge $e \in M$ from G^*, iterating until a planar graph is obtained. There are several ways in which M and $e \in M$ may be selected, and we found that the way in which the selection is made can make an important difference in the performance of the algorithm.

Tightening After adding a cutting plane to an LP and re-solving, it is possible that we may obtain another fractional solution that differs very little from the one just separated. In this case, rather than generating new cuts all over again, it may be desirable to attempt to "fix up" some tight constraints currently in the LP or in the cut-pool by slightly modifying them in such a way as to make the new fractional point infeasible (or make an already violated constraint more violated). This is certainly much faster than separating from scratch, and also does not require G^* to be planar. This type of approach has been very successful on other classes of inequalities (see Applegate et al. [2]) and it had a great impact in our computational results.

To formalize this notion of *simple modifications* for DP-inequalities, recall that every DP-inequality is completely defined by a family of dominoes $\{A_i, B_i\}_{i=1}^k$ and a handle H. Thus, adding and/or deleting a node from any of those sets will result in slight changes of the constraint which potentially could result in a new, violated cut.

In our implementation we consider the following set of simple modifications. Given a node in G^*, we can (i) add it/remove it from a domino; (ii) have it switch sides in a domino; (iii) add it/remove it from the handle; (iv) do some combinations of the previous modifications. We implemented a greedy heuristic which computes the best move for every relevant node[1], and while the best move (among all nodes) reduces the slack of the constraint, perform the move, and update the best move for the relevant nodes in the graph. If all remaining best moves are zero-valued (that is, they do not change the slack), we first do moves that enlarge either the handle or a domino, then do moves that flip elements within a domino and then do moves that shrink a domino or a handle. We repeat this until some improving move is found or until we cannot make any more moves.

[1] A node u is relevant in the heuristic if $\exists e \in \delta(u)$ such that it has a non-zero coefficient in the DP-inequality.

TSPLIB Tests In Tables 1, 2, and 3 we report on a set of tests on all instances from the TSPLIB having at least 3,000 cities. The computations were performed on a single processor of a dual 2.66 GHz Intel Xeon Linux workstation. The LP solver used was ILOG CPLEX 6.5. The algorithm used for planarity testing was Boyer and Myrvold[2] [5].

In the tests in Tables 1 and 2, we used the Concorde command line option -mC48 to allow Concorde to repeatedly call the local-cuts routine up to size 48 (see Applegate et al. [2]); this setting requires additional CPU time over the default version of Concorde, but it allows Concorde to obtain substantially better lower bounds. For each instance in Table 2 we also ran Concorde together with the DP-cut code starting from Concorde's final LP (cutting off the runs after 72 hours), while for each instance in Table 1 we ran Concorde with the DP-cut code starting from scratch.

Table 1. DP-Cuts on TSPLIB Instances

Name	Optimal	Concorde	DP	Gap Δ	Concorde Hours	DP Hours
pcb3038	137694	137660	137687	79%	24.9	8.6
fl3795	28772	28697	28772	100%	21.2	8.2
fnl4461	182566	182555	182559	36%	7.9	3.4
rl5915	565530	565384	565482	67%	103.7	46.1
rl5934	556045	555929	556007	67%	17.5	48.3
pla7396	23260728	23255280	23259532	78%	133.7	106.9

Table 2. DP-Cuts on Larger TSPLIB Instances

Name	Optimal	Concorde	Concorde+DP	Gap Δ	Concorde Hours	DP Hours
usa13509	19982859	19979209	19981173	54%	81.2	72.0
brd14051	469385	469321	469352	48%	53.2	72.0
d15112	1573084	1572853	1572956	45%	114.0	72.0
d18512	(645238)	645166	645193	38%	74.0	72.0

In Table 3 we consider the two largest examples in the TSPLIB. Rather than working from scratch on these instances, we study the effectiveness of DP-cuts in improving the best available LP relaxations. In each instance, we begin with an LP found by Applegate et al. by gathering cuts into a pool during a sequence of 3 branch-and-cut runs (stopping each run after it reached 1,000 active subproblems). The LP was then improved by applying DP-cuts, with new cut pools gathered using 2 branch-and-cut runs for pla33810 and a single short run (to 75 active subproblems) for pla85900.

Table 3. DP-Cuts on Largest TSPLIB Instances

Name	Optimal	Concorde (with pool)	Concorde+DP (with pool)	Gap Δ
pla33810	66048945	66018619	66037858	63%
pla85900	(142382671)	142336550	142354693	39%

As a case-study, starting from the 66,037,858 LP, we have established the optimal value for pla33810. The optimal tour is a slight improvement on the best reported tour, found by by Helsgaun [11] with a variant of

[2] We give special thanks for J.M. Boyer for allowing us to use his implementation of the planarity testing algorithm.

his LKH heuristic. The previously best lower bound for this instance was established by Applegate et al. [1] using Concorde and a branch-and-cut run having a total of 2,821 subproblems.

Concorde Bound	LKH Tour	Optimal
66,032,419	66,050,499	66,048,945

The branch-and-cut that solved the instance used 577 search nodes (given the upper bound of 1 larger than Helsgaun's LKH tour). We also solved the instance a second time starting with a 66,037,858 LP (obtained using the cuts from the earlier run) and an upper bound of 1 greater than the optimal value; the branch-and-cut run in this case used 135 search nodes.

Our solution of pla33810 should be viewed only as evidence of the potential strength of the new procedures; the computational study was made as we were developing our code and the runs were subject to arbitrary decisions to terminate tests as the code improved. The total CPU time used in the solution of pla33810 was approximately 15.7 CPU years (the additional branch-and-cut run of 135 nodes took 86.6 days).

The two remaining open problems in the TSPLIB are d18512 and pla85900; the tour values reported in our tables for these instances were obtained by Tamaki [19] and Helsgaun [11], respectively.

4.2 2-Parity Constraints

To test the efficacy of 2-parity constraints, we developed a heuristic that works by taking tight (or almost tight) domino-parity constraints, and attempts to grow a second handle. For this, consider a super-connected domino-parity constraint with teeth $\mathcal{T}$. The heuristic works in two stages. First, for every tooth $T \in \mathcal{T}$ shortest paths are computed between pairs of nodes in $\overline{\delta(T)}$. Then, in a second stage, the algorithm attempts to connect an odd number of these paths into a simple cycle with edges in $\bar{G}^*$, by using a random-walk which gives preference to edges having small weight (with regard to the values given by the fractional vector x) and which forbids taking two different paths associated to a same tooth $T \in \mathcal{T}$. It is not difficult to see from Lemma 3 that such a structure in $\bar{G}^*$ corresponds to a 2-parity cut.

For this heuristic we also employed many of the techniques described for domino-parity constraints, such as tightening and contraction of edges to ensure planarity.

In Table 4 we report results using an implementation of a 2-parity separation heuristic on a selection of small TSPLIB instances that are not easily solvable at the root node.

Table 4. 2-Parity Cuts on TSPLIB Instances

Name	Optimal	DP	DP+2P	Gap Δ
pcb442	50778	50765	50765	0%
att532	27686	27685	27686	100%
dsj1000	18660188	18659299	18660093	89%
u1060	224094	224044	224054	20%
vm1084	239297	239294	239297	100%

The "DP" column gives the LP value using Concorde with DP-cuts and with local cuts of size 32; the "DP+2P" column reports the LP value obtained by starting with the "DP" LP and running Concorde with the 2P-cut separator. The improvement in the LP gap varied widely, but it is promising that 2P-cuts can often strengthen these already very good LP bounds.

A Appendix: The k-Parity Eulerian Subgraph Problem

We say that a vector $p \in \{0,1\}^k$ is a *k-parity vector*, since it may be interpreted as the product of k even-odd values. In particular we say that $p_o = \{0\}^k$ is the *null* parity.

Consider a graph $G = (V, E)$, possibly having parallel edges. Assume that each edge $e \in E$ has associated a non-negative weight w_e and a k-parity vector p_e. Also consider a collection (ie with possible repetitions) of edges F in G. Define the weight of F to be $w(F) = \sum(w_e : e \in F)$, and the parity of F to be $p(F) = \sum(p_e : e \in F) \mod 2$. For every node $v \in V$ define the collection $F(v) = \{e \in F : e \text{ is incident to } v\}$. We say that F defines an *Eulerian walk* in G if $|F(v)|$ is even for all $v \in V$. If F defines an Euler walk and in addition has no repeated edges, we say that F defines an *Euler subgraph* of G. If in addition F is connected, we say that F is a *cycle*. Finally, if F also satisfies $|F(v)| = 2$, for all $v \in V$ we say that F is a *simple cycle*.

If F is an Eulerian walk of parity p we say that F is a *p-Eulerian walk*. Likewise, we will use the terms *p-Eulerian subgraph*, *p-cycle*, and *p-simple cycle*.

Theorem 4. *Consider $p \in \{0,1\}^k$. The problem of finding a minimum weight p-Eulerian subgraph is polynomially solvable.*

The proof of Theorem 4 consists of three main strands. First we establish some basic results concerning Eulerian subgraphs. Next we show how to build in polynomial time a family of Eulerian subgraphs $\{E_q\}_{q \in \{0,1\}^k}$ all having relatively low weight. Finally, we present an algorithm which takes symmetric differences of these Eulerian subgraphs so as to obtain a minimum weight p-Eulerian subgraph.

Lemma 12. *Basic results concerning p-Eulerian subgraphs.*

1. *Every Eulerian subgraph can be decomposed into an edge-disjoint union of simple cycles.*
2. *Let F_1 be a p_1-Eulerian subgraph, and let F_2 be a p_2-Eulerian subgraph. Then, $F_1 \Delta F_2$ is an Eulerian subgraph of parity $(p_1 + p_2) \mod 2$. Furthermore, $w(F_1 \Delta F_2) \leq w(F_1) + w(F_2)$.*
3. *Consider $p \in \{0,1\}^k$ and let F be a minimum weight p-Eulerian subgraph. If F strictly contains two edge disjoint Eulerian subgraphs F_1, F_2 such that $p(F_1) = p(F_2)$ then $w(F_1) = w(F_2) = 0$.*

Proof. (1) is a well known result, and (2) is straight-forward. To prove (3), assume $0 < w(F_1) + w(F_2)$. Note that $F_1 \cup F_2$ is a p_o-Eulerian subgraph. Hence, $\overline{F} = F \setminus (F_1 \cup F_2)$ is a p-Eulerian subgraph satisfying $w(\overline{F}) < w(F)$ - thus contradicting the minimality. ☐

Lemma 13. *For every $p \in \{0,1\}^k$ there exists a minimum-weight p-Eulerian subgraph which can be decomposed into an edge-disjoint union of simple cycles, each having a different parity.*

Proof. Consider $p \in \{0,1\}^k$, and let F be a minimum-weight p-Eulerian subgraph. From Lemma 12 we know that F can be decomposed into the edge disjoint union of simple cycles. From Lemma 12 it also follows that if F strictly contains two such simple cycles of the same parity, they can be removed from F so as to obtain another minimum-weight p-Eulerian subgraph. By repeating this procedure we will be left with a set of edge disjoint simple cycles of different parities, or the empty set, which is an optimal solution for $p = p_o$. ☐

Lemma 14. *Let C_p^* correspond to a minimum-weight simple cycle of parity p. It is possible to find in polynomial time a p-Eulerian subgraph E_p of weight less than or equal to that of C_p^*.*

Proof. For this purpose construct an auxiliary graph G'.

For each node v in $V(G)$ define 2^k copies in $V(G')$ and label them $v_q,\ q \in \{0,1\}^k$.

For each edge e in $E(G)$ define 2^k copies in $E(G')$ and label them $e_q,\ q \in \{0,1\}^k$. Assume that the end nodes of e in G are $u, w \in V(G)$. Define the end nodes of the edges e_q in the following way: If $q_1, q_2 \in \{0,1\}^k$ are such that $q_1 + q = q_2 \mod 2$, let e_q connect u_{q_1} and w_{q_2}. For every edge e_q define its parity to be $p_{e_q} = p_e$, its weight to be $w_{e_q} = w_e$, and define $\pi(e_q) = e$.

Consider a path $P = \{e'_1, e'_2, \ldots, e'_l\}$ in G' with end nodes u_o and u_q. From the construction of G' it follows that:

- P has parity q.
- The collection of edges $W = \{\pi(e'_1), \pi(e'_2), \ldots, \pi(e'_l)\}$ defines a q-Eulerian walk in G.

On the other hand, note that for every simple cycle $C \subseteq E(G)$ of parity q which passes through node $u \in V$ there exists a path $P \subseteq E(G')$ with end nodes u_o and u_q such that $\pi(P) = C$. This implies that the value of the min-weight u_o-u_q path in G' is not greater than the value of the min-weight simple cycle in G of parity q passing through u.

Finally, note that every q-Eulerian walk W in $E(G)$ contains a q-Eulerian subgraph F in $E(G)$ such that $w(F) \leq w(W)$. Furthermore, such a set F can be obtained from W by iteratively removing pairs of repeated edges until no more exist.

In fact, after removing any pair of repeated edges the number of incident edges to each node will remain even. When we can no longer remove such pairs of edges, every node will be incident to an even number of edges, and there will be no edge repetitions - thus, we will have obtained an Eulerian subgraph. Additionally, note that the parity of the set never changes as we remove such pairs of edges, since for each $e \in E,\ p_e + p_e = p_o \mod 2$. Finally, note that since for each $e \in E,\ w_e \geq 0$ after removing edges the weight can only decrease.

To conclude, let C_p^* denote a minimum-weight simple cycle of parity p in graph G, and for each $v \in V$ let P_v denote a shortest v_o-v_p path in G'. Let P be the minimizer of $w(P_v), v \in V$. From what we have said, the parity of P is p, and $w(P) \leq w(C_p^*)$. Additionally, from $W(P)$ we obtain an p-Eulerian-walk in G whose weight is also not greater than $w(C_p^*)$. Finally, by edge-elimination we find a p-Eulerian subgraph E_p contained in $W(P)$ such that $w(E_p) \leq w(P) \leq w(C_p^*)$ And so we conclude the lemma. □

Proof (Theorem 4). We assume $p \neq p_o$, since otherwise the solution is trivially the empty set.

For each $q \in \{0,1\}^k$, let C_q^* be a minimum-weight simple cycle of parity q, and let E_q be a q-Eulerian subgraph such that $w(E_q) \leq w(C_q^*)$.

For each $q \in \{0,1\}^k$ define a 0-1 variable x_q and consider the following problem:

$$\begin{aligned} &\min \sum_{q \in \{0,1\}^k} x_q w(E_q) \\ &\sum_{q \in \{0,1\}^k} q \cdot x_q = p \mod 2 \\ &x_q \in \{0,1\}\ \forall q \in \{0,1\}^k \end{aligned} \tag{9}$$

Note that x is a feasible solution to (9) if and only if $\{E_q : x_q = 1\}$ is a family of Eulerian subgraphs whose symmetric difference is a p-Eulerian subgraph (See Lemma 12).

Note that the optimal solution to problem (9) gives us, after taking symmetric differences, a minimum-weight p-Eulerian subgraph.

In fact, from Lemma 13 we know that there exists an edge disjoint family of simple cycles $\{C_i\}_{i=1}^r$ whose union is a minimum-weight p-Eulerian subgraph. Assume that for each $i \in 1, \ldots, r$ the parity of cycle C_i is p_i. Consider the family of Eulerian subgraphs $\{E_{p_i}\}_{i=1}^r$. Recall that $w(E_{p_i}) \leq w(C_i)$. Furthermore, by Lemma 12 we know that the symmetric difference of these Eulerian subgraphs is a p-Eulerian subgraph whose total weight is not greater than the sum of the weights of the individual Eulerian subgraphs. Hence, by taking the symmetric difference of Eulerian subgraphs $\{E_{p_i}\}_{i=1}^r$ we obtain a minimum-weight p-Eulerian subgraph. Since the vector $x' \in \{0,1\}^k$ defined by:

$$x'_q = \begin{cases} 1 & \text{if } q = p_i, \text{ for } i = 1, \ldots, r \\ 0 & \text{otherwise} \end{cases}$$

is a feasible solution to the problem (9) we conclude.

Finally, note that for k fixed, problem (9) does not depend in the size of the multi-graph G, but only on k. Hence, it can be solved in polynomial time. □

References

1. Applegate, D., R. Bixby, V. Chvátal, W. Cook. 1998. On the solution of traveling salesman problems. Documenta Mathematica Journal der Deutschen Mathematiker-Vereinigung, International Congress of Mathematicians. 645–656.

2. Applegate, D., R. Bixby, V. Chvátal, W. Cook. 2003. Implementing the Dantzig-Fulkerson-Johnson algorithm for large traveling salesman problems. Mathematical Programming **97**, 91–153.
3. Ahuja R.K., T.L. Magnanti, J.B. Orlin. 1993. *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall, New Jersey.
4. Boyd, S., S. Cockburn, D. Vella. 2001. On the domino-parity inequalities for the STSP. Computer Science Technical Report TR-2001-10. University of Ottawa.
5. Boyer, J. M., W. Myrvold. 2004. On the cutting edge: simplified O(n) planarity by edge addition. Journal of Graph Algorithms and Applications. To appear.
6. Chvátal, V. 1973. Edmonds polytopes and weakly hamiltonian graphs. Mathematical Programming **5**, 29–40.
7. Dantzig, G., R. Fulkerson, S. Johnson. 1954. Solution of a large-scale traveling salesman problem. Operations Research **2**, 393–410.
8. Fleischer, L., É. Tardos. 1999. Separating maximally violated comb inequalities in planar graphs. Mathematics of Operations Research **24**, 130–148.
9. Grötschel, M., O. Holland. 1991. Solution of large-scale symmetric travelling salesman problems. Mathematical Programming **51**, 141–202.
10. Grötschel, M., W. R. Pulleyblank. 1986. Clique tree inequalities and the symmetric travelling salesman problem. Mathematics of Operations Research **11**, 537–569.
11. Helsgaun, K. 2000. An effective implementation of the Lin-Kernighan traveling salesman heuristic. European Journal of Operational Research **126**, 106–130.
12. Jünger, M., G. Reinelt, G. Rinaldi. 1995. The traveling salesman problem. M. Ball, T. Magnanti, C. L. Monma, G. Nemhauser, eds. *Handbooks on Operations Research and Management Sciences: Networks*. North Holland, Amsterdam, The Netherlands. 225–330.
13. Letchford, A. N. 2000. Separating a superclass of comb inequalities in planar graphs. Mathematics of Operations Research **25**, 443–454.
14. Naddef, D. 2002. Polyhedral theory and branch-and-cut algorithms for the symmetric traveling salesman problem. G. Gutin, A. Punnen, eds. *The Traveling Salesman Problem and Its Variations*. Kluwer, Dordrecht, pp. 29–116.
15. Nagamochi, H., K. Nishimura, T. Ibaraki. 1997. Computing all small cuts in undirected networks. SIAM Journal on Discrete Mathematics **10**, 469–481.
16. Naddef, D., S. Thienel. 2002. Efficient separation routines for the symmetric traveling salesman problem II: separating multi handle inequalities. Mathematical Programming **92**, 257–283.
17. Padberg, M. W., M. R. Rao. 1982. Odd minimum cut-sets and *b*-matchings. Mathematics of Operations Research **7**, 67–80.
18. Padberg, M., G. Rinaldi. 1991. A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems. SIAM Review **33**, 60–100.
19. Tamaki, H. 2003. Alternating cycle contribution: a tour-merging strategy for the travelling salesman problem. Max-Planck Institute Research Report MPI-I-2003-1-007. Saarbrücken, Germany.