On the domino-parity inequalities for the $STSP$*

Sylvia Boyd
University of Ottawa, Ottawa, Canada

Sally Cockburn
Hamilton College, Clinton, N.Y., USA

Danielle Vella
University of Ottawa, Ottawa, Canada

October 8, 2001

Abstract

One method which has been used very successfully for finding optimal and provably good solutions for large instances of the Symmetric Travelling Salesman Problem ($STSP$) is the branch and cut method. This method requires knowledge of classes of useful valid inequalities for the polytope associated with the $STSP$, as well as efficient separation routines for these classes of inequalities.

Recently a new class of valid inequalities called the domino-parity inequalites were introduced for the $STSP$. An efficient separation routine is known for these constraints if certain conditions are satisfied by the point to be separated. This separation routine has never been implemented or tested. We present several performance enhancements for this separation routine, and discuss our implementation of this improved algorithm. We test our implementation and provide results which we believe demonstrate the practical usefulness of these constraints and the separation routine for the $STSP$ within a branch and cut framework.

It was also recently shown that a subset of the domino-parity constraints, called the twisted comb constraints, are facet-inducing for the $STSP$. This class is a subset of the domino-parity constraints

*This research was partially supported by grants from the Natural Sciences and Engineering Research Council of Canada.

which have one "non-regular" domino. In this paper we show that these inequalities are not equivalent to other well-known classes of facet-inducing inequalities for the $STSP$, and thus form a new class of previously unknown facets for the $STSP$. We also give a characterization of the facet-inducing domino-parity constraints with exactly one non-regular domino.

1 Introduction

Given the complete graph $K_n = (V, E)$ on n nodes with edge costs $c \in \mathbf{R}^E$, the *Symmetric Travelling Salesman Problem* (henceforth $STSP$) is to find a Hamiltonian cycle (or *tour*) in K_n of minimum cost. This problem has practical applications in areas such as printed circuit board production, X-ray crstallography and vehicle routing (see [15]).

For any edge set $F \subseteq E$ and $x \in \mathbf{R}^E$, let $x(F)$ denote the sum $\sum_{e \in F} x_e$. For any node set $W \subset V$, let $\delta(W)$ denote $\{uv \in E | u \in W, v \notin W\}$. An integer linear programming (ILP) formulation for the $STSP$ is as follows:

$$\begin{aligned}
\text{minimize } cx & & (1)\\
\text{subject to } x(\delta(v)) = 2 & \quad \text{for all } v \in V, & (2)\\
x(\delta(S)) \geq 2 & \quad \text{for all } S \subset V, 3 \leq |S| \leq n-3, & (3)\\
0 \leq x_e & \quad \text{for all } e \in E, & (4)\\
x_e \leq 1 & \quad \text{for all } e \in E, & (5)\\
x & \quad \text{integer.} & (6)
\end{aligned}$$

Equations (2) are called *degree constraints*, constraints (3) are called *subtour elimination constraints*, constraints (4) are called *non-negativity constraints*, and constraints (5) are called *upper bound constraints*. By taking the convex hull in $\mathbf{R}^E$ of all vectors x satisfying (2) to (6) we obtain the *Symmetric Travelling Salesman Polytope*, denoted by $STSP(n)$.

If we drop the integer requirement for (1), we obtain a linear programming (LP) relaxation of the $STSP$ called the *Subtour Elimination Problem* (SEP), whose associated polytope is denoted by $SEP(n)$.

The $STSP$ is known to be NP-hard, even in the Euclidean case (see [14]). One method which has been used very successfully to find optimal and provably good solutions for large instances of the $STSP$ is the *branch and cut method* (see [1], [23] and [24]). This method starts with the optimal solution x^* for some LP relaxation of the $STSP$. Then, while x^* does not represent a tour, the method iteratively searches for inequalities that are

valid for $STSP(n)$ and are violated by, or "cut off" the current LP solution x^*. If any such inequalities, or "cuts", are found, they are added to the LP which is reoptimized. If a point is reached where no such violated inequalities can be found, a branch operation is applied, as within a standard branch and bound framework.

The success of the branch and cut method depends on several things. It requires the knowledge and use of "good" cuts (i.e. one ideally would like to identify violated inequalities that allow the iterative procedure to reach the desired integer LP solution using as few iterations as possible). Typically, classes of inequalities which are known to be *facet-inducing* for $STSP(n)$ are used for this purpose. These are inequalities which are necessary in any LP description of $STSP(n)$, and imply all other valid inequalities for the problem. Note that all of the constraints (2)-(5) are known to be facet-inducing for $STSP(n)$ for $n \geq 5$ (see [15]).

The success of the branch and cut method also depends on the existence of efficient methods for finding violated inequalities from our targeted classes of constraints, if such violations exist. Given a class of valid inequalities, an *exact separation algorithm* is one which given a point x^*, either finds one or more violated inequalities from the class or concludes that none exist.

For $STSP(n)$, polynomial-time exact separation algorithms exist for the subtour elimination constraints (see [21]) and the 2-matching constraints (see [22]). Recently, much time and effort has been invested in finding a polynomial-time exact separation for the well-known comb inequalities, which are used extensively in branch and cut methods for the $STSP$. In [5], Carr provides such an algorithm for combs with a fixed number of teeth. In [1], heuristics for comb inequality separation are discussed. Note that for points $x^* \in SEP(n)$, a comb constraint can be violated by at most 1. In [9], Fleischer and Tardos give a polynomial-time exact separation algorithm for the class of such maximally violated comb constraints (i.e. those violated by 1) for points $x^* \in SEP(n)$ such that the support graph G^* for x^* is planar (given a point $x^* \in \mathbf{R}^E$, the *support graph* $G^* = (V^*, E^*)$ of x^* is the subgraph of K_n induced by the edge set $E^* = \{e \in E | x^*_e > 0\}$). In [4], Caprara, Fischetti and Letchford give a polynomial-time separation algorithm for the class of maximally violated mod-2 cuts, which is a class of valid inequalities for $STSP(n)$ that contain the comb inequalities.

More recently, a new class of valid inequalities for $STSP(n)$ called the domino-parity (DP) constraints was introduced by Letchford in [16]. This class of inequalities is a subclass of the mod 2 cuts for $STSP(n)$ and is a superclass of the comb inequalities. Moreover, in his paper, Letchford also describes an $O(n^3)$ exact separation algorithm for these DP-constraints under the conditions that the point x^* we wish to cut off lies in $SEP(n)$

and has a planar support graph. His separation algorithm has the advantage over the algorithms in [9] and [4] that it does not restrict the separation to only maximally violated inequalities from the class. However, it is at a disadvantage over the separation algorithm from [4] for mod-2 cuts in that his algorithm requires that the current solution x^* being separated must have a planar support graph, and this will not always be the case.

There has been no previous testing of any of these three exact separation algorithms, and in particular, of the Letchford separation algorithm for DP-constraints. Because many inequalities in this class of constraints are not facet-inducing, some people have questioned their usefulness as cutting planes. Furthermore, some wonder about the practicality of Letchford's algorithm, given that many of the solutions x^* encountered during branch and cut have non-planar support graphs.

The main objective of this paper is to investigate the practical usefulness of the DP-constraints and the Letchford separation algorithm within a branch and cut framework for the $STSP$. Essentially we want our research to provide information and results that would help someone decide whether or not it would be worthwhile to include these constraints as cutting planes for the $STSP$. To this end, we implemented the DP-constraint separation algorithm and tested it on a number of Euclidean $STSP$s from the $STSP$ test problem library TSPLIB [25]. We believe our results demonstrate the practical usefulness of both the DP-constraints and Letchford's separation routine for solving $STSP$s within a branch and cut framework. Note that our implementation contains a number of enhancements to Letchford's original algorithm, including an important one that allows the algorithm to often be used for points that have non-planar support graphs. These enhancements and our test results are discussed in detail in Section 3.

The second question we investigate in the present paper has to do with which DP-constraints induce facets for $STSP(n)$. More specifically, do the DP-constraints contain other classes of facet-inducing inequalities besides the comb constraints which are different from other known classes of facet-inducing constraints for $STSP(n)$? It turns out that the answer to this question is yes. Recently in [2], a subclass of the DP-constraints called the twisted comb constraints were introduced and shown to be facet-inducing for $STSP(n)$. In Section 4 we show that these twisted comb constraints, along with a family of constraints equivalent to comb constraints, characterize the facet-inducing DP-constraints with exactly one "non-regular" domino. This is an important first step in characterizing exactly which DP-constraints represent facets for $STSP(n)$. In Section 5 we go on to show that these twisted comb constraints are not equivalent to other well-known classes of facet-inducing constraints for $STSP(n)$, thus showing that they provide a

new class of facet-inducing inequalities which can be used as cuts in a branch and cut framework for $STSP(n)$.

In the next section we describe the DP-constraints. The remainder of this section is devoted to some notation.

Given a graph $G = (V, E)$, we sometimes use the notation $E(G)$ for the edgeset E of G and $V(G)$ for the nodeset V of G. For any $A \subset V$, let $\delta(A)$ represent the set of edges in E with exactly one end in A, and let $\gamma(A)$ represent the set of edges in E with both ends in A. Given two subsets A and B of V, we let $E(A : B) := \{uv \in E | u \in A, v \in B\}$.

A graph is called *complete* if every pair of nodes is joined by an edge. The complete graph on n nodes is denoted by K_n. For the remainder of this paper, we use E and V to denote the edge set and node set of the complete graph K_n unless otherwise stated.

2 The DP-constraints

Definition A *domino* is a node subset $D = A \cup B \neq V$, where A and B are non-empty, disjoint node subsets; that is, $\emptyset \neq A, B \subset V$ and $A \cap B = \emptyset$. The edge set $E(A : B)$ is called the *semicut* of the domino.

Definition A *domino-comb* $\{F; D_1, D_2, \ldots, D_p\}$ consists of an edge subset F, and an odd number of dominoes $D_i = A_i \cup B_i, i = 1, 2, \ldots, p$, such that

$$\begin{aligned}\delta(H) = \{e \in E | e \text{ is in an odd number of the edge sets} \\ F, E(A_1 : B_1), E(A_2 : B_2), \ldots, E(A_p : B_p)\}\end{aligned}$$

for some node subset $H \subset V$. This set H is known as the *handle* of the domino-comb.

Note that given a domino-comb $\{F; D_1, D_2, \ldots, D_p\}$, we can equivalently represent it as $\{H; D_1, D_2, \ldots, D_p\}$ where H is the handle of the domino-comb. In this case, F can be obtained as follows: $F = F_1 \cup F_2$, where

$$\begin{aligned}F_1 = \quad & \{e \in E | e \in \delta(H) \text{ and e appears in an even number of the semicuts} \\ & E(A_i : B_i), i = 1, 2, \ldots, p\}; \\ F_2 = \quad & \{e \in E | e \notin \delta(H) \text{ and e appears in an odd number of the semicuts} \\ & E(A_i : B_i), i = 1, 2, \ldots, p\}.\end{aligned}$$

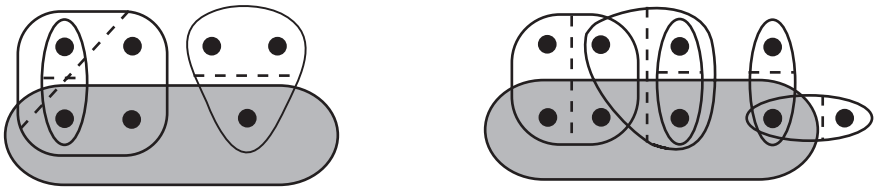

Figure 1: Examples of domino-combs

A domino-comb for which the dominoes are disjoint, $p \geq 3$ and $F = \delta(H) \backslash \{ \bigcup_{i=1}^{p} E(A_i : B_i) \}$ is known as a *comb*. In this case, each edge $e \in \delta(H)$ is in exactly one of the sets $F, E(A_1 : B_1), E(A_2 : B_2), \ldots, E(A_p : B_p)$. In general domino-combs however, the dominoes need not be disjoint. Moreover, the edges in the semicut of a domino $E(A_i : B_i)$ need not be contained in $\delta(H)$; that is, it may not be the case that $A_i = D_i \cap H$ and $B_i = D_i \backslash H$, as is true with combs. This can happen either when a semicut edge e occurs in an even number of semicuts (since dominoes need not be disjoint, their semicuts can intersect nontrivially), or when a copy of e exists in F. Some examples of domino-combs are illustrated in Figure 1. Rather than depict dominoes and the edges of F, we show dominoes and the implicitly defined handle H (which is shaded). In the figure, a dotted line in a domino D_i is used to separate A_i and B_i. We use this convention for drawing domino-combs throughout the paper.

Definition For a domino-comb $\{F; D_1, \ldots, D_p\}$, the corresponding *DP-constraint* is

$$\sum_{i=1}^{p} x(\delta(D_i)) + \sum_{i=1}^{p} x(E(A_i : B_i)) + x(F) \geq 3p + 1. \tag{7}$$

In the case of combs, (7) gives the well-known comb constraints for the *STSP*:

$$\sum_{i=1}^{p} x(\delta(D_i)) + x(\delta(H)) \geq 3p + 1 \tag{8}$$

The comb constraints were shown to be facet-inducing for $STSP(n)$ in [11] and [12].

Note that (7) can also be written as

$$\sum_{i=1}^{p} x(E(A_i : B_i)) + \sum_{i=1}^{p} x(E(A_i : C_i)) + \sum_{i=1}^{p} x(E(B_i : C_i)) + x(F) \geq 3p + 1 \quad (9)$$

where $C_i = V \backslash D_i$ for $i = 1, 2, \ldots, p$. This form of the DP-constraint has the advantage that we can view each domino as a tripartition $A_i \cup B_i \cup C_i$ of the nodes of V without specifying which pair of these sets form the domino.

3 Testing the Usefulness of DP-constraints and the Separation Algorithm

As mentioned earlier, one of the main objectives of this paper is to investigate the usefulness of the DP-constraints as cuts in a branch and cut setting, as well as the practicality of Letchford's separation algorithm for finding these cuts. We would like to emphasize here that our goal was not to design a tool to solve $STSPs$ faster and better than everyone else, but rather to simply help someone decide whether or not it is worthwhile to include DP-constraints as one of the classes of inequalities considered for cuts when implementing a branch and cut method for the $STSP$. Thus in our testing we concentrated on the cut phase of the branch and cut method, iteratively looking for violated DP-constraints using only Letchford's separation algorithm, adding these cuts to the current LP and reoptimizing. To this end, we developed an implementation of Letchford's separation algorithm in which we included a number of important enhancements and changes to the overall process. These are discussed in Section 3.1.

The details and results of our testing are discussed in Section 3.2. This testing process was greatly improved and simplified for us by making use of CONCORDE, which is a very successful and powerful software package developed by Applegate, Bixby, Cook and Chvátal, designed specifically for applying the branch and cut method to the $STSP$. Among other things, CONCORDE allows the user to iteratively add his or her own set of violated constraints (contained in a file) at each stage of the cutting process.

3.1 Enhancements Made in the Implementation

In order to facilitate the practical usefulness of Letchford's DP-constraint separation routine and enhance our ability to successfully use CONCORDE in the testing process, we added some enhancements to our implementation of this separation algorithm; they are described next.

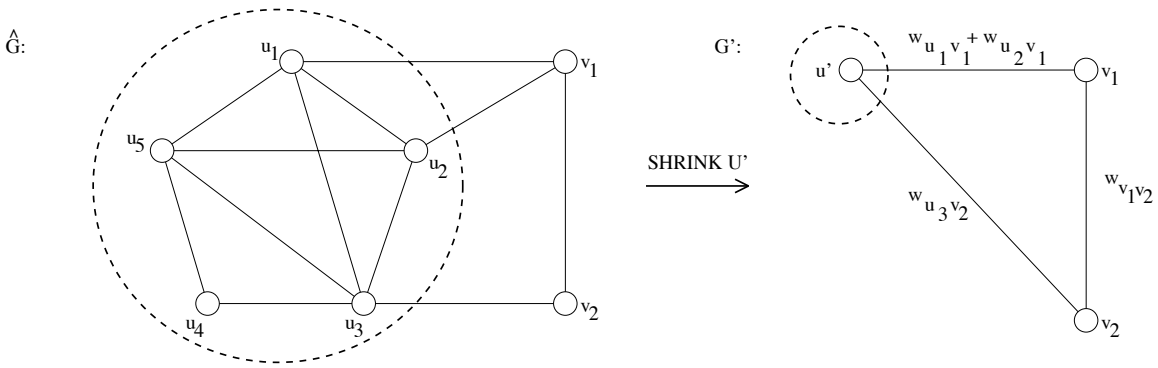

Figure 2: A shrinking example

3.1.1 Shrinking to Obtain Planarity

As discussed earlier, in order to use the DP-constraint separation algorithm on the current LP solution x^*, we must have that the corresponding support graph $G^* = (V^*, E^*)$ is planar. Although this will automatically be the case when x^* is the optimal solution for the subtour problem for Euclidean $STSP$s (see [10]), it is certainly not the case in general, and we frequently encountered non-planar support graphs during our testing. Fortunately, we found that it is often possible to make use of the DP-constraint separation algorithm even if the support graph G^* for our current solution x^* is non-planar. The method we used is based on the idea of shrinking portions of G^* in order to obtain a planar graph G' such that the violated DP-constraints found using the new graph G' correspond to violated DP-constraints for our original point x^*. Although this method is not guaranteed to find a violated DP-constraint which cuts x^* off if one exists, we found that we were often able to use it to successfully generate violated DP-constraints in our testing. We now explain the details of this process.

Definition Let $\hat{G} = (\hat{V}, \hat{E})$ be a weighted graph with edge weights $w \in \mathbf{R}^{\hat{E}}$ and let $U \subset \hat{V}$ be a node subset of $\hat{V}$. Let G' be obtained from $\hat{G}$ by replacing U and all edges in $\gamma(U)$ by a single node u' and for each $v \in \hat{V} \setminus U$, replacing all edges $\hat{E}(U : \{v\})$ by a single edge $u'v$ whose weight is the sum of the weights $(w_e : e \in \hat{E}(U : \{v\}))$. Then we say that G' was obtained from $\hat{G}$ by *shrinking the node set* U and we call u' the *pseudonode* obtained by shrinking node set U. Figure 2 shows an example of this shrinking process, where $U = \{u_1, u_2, u_3, u_4, u_5\}$.

If we let $\hat{G}$ be the support graph $G^* = (V^*, E^*)$ of a point x^*, we can use $(x^*{}_e : e \in E^*)$ as the weights for G^*, and when we shrink a node set U to obtain $G' = (V', E')$, we can associate a point $x' \in \mathbf{R}^{E(K_{|V'|})}$ (i.e. x' is indexed by the edge set of the complete graph induced by V') with G', where

$$x'_e = \begin{cases} w_e, & \text{for } e \in E' \\ 0, & \text{otherwise.} \end{cases}$$

Clearly G' is the support graph of x', and we say that x' *is the point obtained from* x^* *by shrinking* U. The following very useful proposition shows that we can generate violated DP-constraints for our original point x^* by looking instead for DP-constraints which are violated by x'.

Proposition 3.1 *Given a point* $x^* \in \mathbf{R}^{E(K_n)}$ *and its support graph* $G^* = (V^*, E^*)$ *and given* $U \subset V^*$ *, let the graph* $G' = (V', E')$ *and point* x' *be obtained by shrinking node set* U *in* G^**. Let* $\{F'; D'_1, D'_2, \ldots, D'_p\}$ *be a domino-comb in* $K_{|V'|}$ *for which the corresponding* DP*-constraint is violated by* x'*. Then, this domino-comb can be used to construct a corresponding domino-comb in* K_n *for which the related* DP*-constraint is violated by* x^**.*

Proof Let u' be the pseudonode in G' corresponding to U, and let H' be the handle of the domino-comb $\{F'; D'_1, D'_2, \ldots, D'_p\}$. Construct sets $F^* \subset E(K_n)$, $D^*_i \subset V(K_n)$, $i = 1, 2, \ldots, p$ as follows: Let F^* be the edge set obtained from F' by replacing each edge $u'v \in F'$ by the edge set $E(U : \{v\})$ in K_n, and for $i = 1, 2, \ldots, p$, let D^*_i be the node set in K_n obtained from D'_i by replacing any occurrence of u' by U.

Let $\mathcal{K}$ represent the set of all edges $e \in E(K_n)$ such that e is in an odd number of the edge sets $F^*, E(A^*_1 : B^*_1), E(A^*_2 : B^*_2), \ldots, E(A^*_p : B^*_p)\}$. We first show that $\{F^*; D^*_1, D^*_2, \ldots, D^*_p\}$ is a domino-comb by showing that $\mathcal{K} = \delta(H^*)$ for some $H^* \subset V^*$.

Consider the relationship between the two domino-combs. Clearly $\mathcal{K} = \{wv \in \delta(H') | w \neq u'\} \cup \{uv \in E(K_n) | u \in U, u'v \in \delta(H')\}$. Thus, $\mathcal{K} = \delta(H^*)$ for H^* obtained from H' by replacing any occurrence of u' by U.

Next we wish to show that the DP-constraint corresponding to the domino-comb $\{F^*; D^*_1, D^*_2, \ldots, D^*_p\}$ is violated by x^*. To see this, recall that

$$x'_{yv} = \begin{cases} x^*(E(U : \{v\})), & \text{if } y = u', v \in V' \setminus \{u'\} \\ x^*_{yv}, & y, v \neq u', w \end{cases}$$

and thus the right-hand side of the DP-constraints for the domino-combs $\{F^*; D^*_1, D^*_2, \ldots, D^*_p\}$ and $\{F'; D'_1, D'_2, \ldots, D'_p\}$ are identical. The result follows. $\square$

Note that if we start with some point x^* in $SEP(n)$ and shrink some node set U to obtain point x', this new point may no longer satisfy the degree constraints (2) or the upper bound constraints (5) for the SEP. However, although it is not first presented as such in [16], Letchford does point out in this same paper that the separation algorithm will also work when the solution vector x' only satisfies the non-negativity (4) and subtour elimination constraints (3) for all $S \in V$, as long as the support graph of x' is planar. Thus it follows that if G' is planar, we can use the separation algorithm to look for DP-constraints violated by x', and then use these constraints to construct violated DP-constraints for x^*. This leads to the following method for handling situations where our current LP solution x^* has a support graph $G^* = (V^*, E^*)$ which is non-planar:

Step 1 Look for disjoint node subsets $U_1, U_2, \ldots, U_k$, where $U_i \subset V^*$ for $i = 1, 2, \ldots k$, such that the graph $G' = (V', E')$ obtained by shrinking each of the sets U_i is planar. Let x' be the corresponding point obtained by this shrinking.

Step 2 Use the separation algorithm to look for DP-constraints violated by x'.

Step 3 Use the method described in Proposition 3.1 to transform each of the violated constraints found in Step 2 into violated DP-constraints for x^*.

It is important to note that this method does not provide an exact separation routine for x^*. That is, it is possible to have a violated DP-constraint for x^* with no corresponding violated DP-constraint for x' (an example of this is given in [26]). Thus the above method is not guaranteed to find a violated DP-constraint for x^* if one exists. However, we can clearly increase the method's effectiveness as a heuristic by finding the sets $U_1, U_2, \ldots, U_k$ for Step 1 which shrink as little of the graph G^* as possible. So ideally we would like to solve the following problem for Step 1: Given a graph $G^* = (V^*, E^*)$, find disjoint node subsets $U_1, U_2, \ldots, U_k$, where $U_i \subset V^*$ for $i = 1, 2, \ldots k$, such that the graph $G' = (V', E')$ obtained by shrinking these sets is planar, and $|V^*| - |V'|$ is minimized. Currently, we do not know of an efficient method for solving this problem in general, but there are several good heuristics which can be used (see [26]).

3.1.2 Finding a "CONCORDE FRIENDLY" Equivalent Form

One of the requirements of CONCORDE is that any constraint added as a cut must be in a reasonable *closed-set form*, i.e. must be in the form

$$\sum_{i=1}^{k} x(\delta(S_i)) \geq \alpha$$

for node subsets $S_1, S_2, \ldots, S_k$, where k is not too large. Thus in order to make use of CONCORDE for our testing, it was necessary to first find the closed-set form for general DP-constraints, and below we describe how this can be done. Unfortunately, this closed-set form can sometimes involve a very large number of node subsets, which again is unacceptable for CONCORDE. In such situations we found that it was often possible to "flip compartments" in dominoes to obtain a much nicer closed-set form of the inequality involving drastically fewer node subsets. This process is also described below.

Definition Given a domino-comb $\{F; D_1, D_2, \ldots, D_p\}$ with handle H, we call one of its dominoes D_k *regular* if

(i) $E(A_k : B_k) \subset \delta(H)$;

(ii) $E(A_k : B_k) \cap E(A_j : B_j) = \emptyset$ for all $1 \leq j \leq p$, $j \neq k$;

(iii) $E(A_k : B_k) \cap \delta(D_j) = \emptyset$ for all $j = 1, 2, \ldots, p$.

A domino that is not regular is called *non-regular*.

It is useful to understand the properties satisfied by regular dominoes, which are described in Proposition 3.2.

Proposition 3.2 *Let $\{F; D_1, D_2, \ldots, D_p\}$ be a domino-comb, where $D_i = A_i \cup B_i$, $i = 1, 2, \ldots, D_p$. Then, the following properties hold:*

(a) If a regular domino D_k intersects another domino D_i, then D_k must be entirely inside either A_i or B_i.

(b) All regular dominoes are disjoint from each other.

(c) All regular dominoes are minimal (i.e. they contain no other dominoes).

Proof Consider property (a). If regular D_k has some nodes both inside and outside another domino D_i, then D_k would not satisfy condition (iii) for regular dominoes. If D_k has some some nodes in A_i and some nodes in B_i, then D_k does not satisfy condition (ii) for regular dominoes. The result follows.

Consider property (b). By (a), if a regular domino D_k intersects a regular domino D_r, then D_k must lie entirely inside A_r or B_r. but if that is the case, one of either D_k or D_r must not satisfy condition (i) for regular dominoes. Hence property (b) holds.

Finally, property (c) follows from property (a). $\square$

Note that if we have a domino-comb for which all dominoes are regular, then we have a comb, and the corresponding DP-constraint (8) is already in the required closed-set form.

To obtain a closed-set form for general domino-combs, we begin by partitioning the dominoes into regular and non-regular. Without loss of generality, let $D_1, D_2, \ldots, D_r$ be the regular dominoes and let $D_{r+1}, D_{r+2}, \ldots, D_p$ be the non-regular ones for the domino-comb. Then by using the fact that

$$x(\delta(D_j)) + x(E(A_j : B_j)) = x(\delta(A_j)) + x(\delta(B_j)) - x(E(A_j : B_j))$$

for each non-regular domino D_j, we can rewrite the DP-constraint (7) as

$$\begin{aligned} \sum_{i=1}^{r} x(\delta(D_i)) + \sum_{i=r+1}^{p} x(\delta(A_i)) + \sum_{i=r+1}^{p} x(\delta(B_i)) + \sum_{i=1}^{r} x(E(A_i : B_i)) \\ + x(F) - \sum_{i=r+1}^{p} x(E(A_i : B_i)) \geq 3p + 1. \end{aligned} \tag{10}$$

Now consider the corresponding handle H. By definition, the edges of $\delta(H)$ can be partitioned into the following three sets:

(i) $\bigcup_{i=1}^{r} E(A_i : B_i)$ (i.e. the semicut edges for regular dominoes);

(ii) F_1;

(iii) $K := \{e \in E | e \in \delta(H)$ and e is in an odd number of the non-regular semicuts $E(A_i : B_i), i = r+1, \ldots, p\}$.

Applying this to (10) gives

$$\begin{aligned} \sum_{i=1}^{r} x(\delta(D_i)) + x(\delta(H)) + \sum_{i=r+1}^{p} x(\delta(A_i)) + \sum_{i=r+1}^{p} x(\delta(B_i)) \\ - \left(\sum_{i=r+1}^{p} x(E(A_i : B_i)) - x(F_2) + x(K) \right) \geq 3p + 1. \end{aligned} \tag{11}$$

We now concentrate on the term

$$\sum_{i=r+1}^{p} x(E(A_i : B_i)) - x(F_2) + x(K) \tag{12}$$

on the left-hand side of (11), as this is the only part of (11) which does not adhere to the closed-set form. By definition, because all regular dominoes D_i have $E(A_i : B_i) \subset \delta(H)$, we have

$$F_2 = \{e \in E | e \notin \delta(H) \text{ and } e \text{ is in an odd number of non-regular domino semicuts}\}. \tag{13}$$

For all $e \in E$, let ϕ_e represent the number of non-regular domino semicuts containing edge e. Using (13), we then have that (12) can be written as

$$\begin{aligned} \sum(\phi_e x_e : e \in E, \phi_e \text{ even }) + \quad & \sum((\phi_e - 1)x_e : e \in E, \phi_e \text{ odd }, e \notin \delta(H)) \\ + \quad & \sum((\phi_e + 1)x_e : e \in E, \phi_e \text{ odd }, e \in \delta(H)). \end{aligned} \tag{14}$$

Clearly, for all $e \in E$, the coefficient of x_e in (14) is a non-negative, even integer, so we can represent (12) as

$$\sum_{e \in E} 2a_e x_e$$

where, for all $e \in E$, a_e is a non-negative integer.
Substituting this into (11) gives

$$\begin{aligned} \sum_{i=1}^{r} x(\delta(D_i)) + \quad & x(\delta(H)) + \quad \sum_{i=r+1}^{p} x(\delta(A_i)) \\ + \quad & \sum_{i=r+1}^{p} x(\delta(B_i)) - \quad \sum_{e \in E} 2a_e x_e \geq 3p + 1. \end{aligned} \tag{15}$$

Now consider any edge $e = uv$. By adding the degree constraints (2) for u and v, we obtain

$$x\delta(\{u, v\}) + 2x_e = 4.$$

Thus, if we add a_e times the degree constraint for nodes u and v for every edge $e = uv \in E$ to the DP-constraint (15), we obtain the equivalent constraint

$$\sum_{i=1}^{r} x(\delta(D_i)) + x(\delta(H)) + \sum_{i=r+1}^{p} x(\delta(A_i)) + \sum_{i=r+1}^{p} x(\delta(B_i)) + \sum_{uv \in E} a_{uv} x(\delta(\{u,v\})) \geq 3p + 1 + 4 \sum_{e \in E} a_e \tag{16}$$

which is the required closed-set form.

Although constraint (16) is in closed-set form, it has one drawback. In the case that there are non-regular dominoes, this closed-set form may contain a very large number of node subsets of size two. In some cases this meant that the DP-constraints we generated in our testing were still not accepted by CONCORDE, even after being converted into form (16).

Fortunately we were able to overcome this problem. It turns out that it is possible to efficiently maximize the number of dominoes which can be considered regular for a given DP-constraint, thereby minimizing the number of node subsets in the closed-set form. This process is based on the idea of switching the compartments of a domino, as defined below.

Definition Let $D = A \cup B$ be a domino, and let $C = V \backslash D$. Sets A, B and C form a tripartition of the nodes V of K_n, which we call the *compartments* of D. We call A and B the *inside compartments* of D, and C the *outside compartment*.

Definition Given a domino $D = A \cup B$, we say dominoes $D' = A \cup C$ and $D'' = B \cup C$ are obtained by *switching the compartments* of D.

Definition Given sets A and B, the *symmetric difference of A and B*, denoted by $A \Delta B$, is the set $(A \backslash B) \cup (B \backslash A)$.

Theorem 3.3 below (which also appears in [2]) shows that we can essentially regard any two compartments of each domino as the inside compartments, i.e. switching compartments in any domino of a domino-comb results in another domino-comb which generates the same DP-constraint, albeit one with a somewhat different handle.

Theorem 3.3 *Let $\{F; D_1, D_2, \ldots, D_p\}$, $D_i = A_i \cup B_i$, be a domino-comb and let H be the associated handle. Let $C_1 = V \backslash D_1$ and let $D'_1 = A_1 \cup C_1$. Then, $\{F; D'_1, D_2, \ldots, D_p\}$ is also a domino-comb and the associated handle is $H' = H \Delta A_1$. Furthermore, the DP-constraints corresponding to $\{F; D_1, D_2, \ldots, D_p\}$ and $\{F; D'_1, D_2, \ldots, D_p\}$ are the same constraint.*

Proof For each $e \in E$, let μ_e represent the number of the sets $F, E(A_1 : B_1), E(A_2 : B_2), \ldots, E(A_p : B_p)$ which contain e and let μ'_e represent the number of the sets $F, E(A_1 : C_1), E(A_2 : B_2), \ldots, E(A_p : B_p)$ which contain e. Thus $\delta(H) = \{e|\mu_e \text{ is odd}\}$, and we wish to show that $\delta(H') = \{e|\mu'_e \text{ is odd}\}$, where $H' = H\Delta A_1$.

Since the only difference in the sets for μ and μ' is the replacement of $E(A_1 : B_1)$ by $E(A_1 : C_1)$ and these two sets are disjoint, we have that

$$\mu'_e = \begin{cases} \mu_e + 1, & e \in E(A_1 : C_1) \\ \mu_e - 1, & e \in E(A_1 : B_1) \\ \mu_e, & \text{otherwise.} \end{cases}$$

Thus, the parities of μ_e and μ'_e are different only for edges e in $E(A_1 : B_1) \cup E(A_1 : C_1) = \delta(A_1)$. It follows that μ'_e is odd for edges in $\delta(H)$ but not in $\delta(A_1)$, and edges in $\delta(A_1)$ but not in $\delta(H)$. Thus, $\{e \in E : \mu'_e \text{ is odd}\} = \delta(H\Delta A_1)$, as required.

Note that if we consider our DP-constraint in form (9), it is clear that both the domino-combs $\{F; D_1, D_2, \ldots, D_p\}$ and $\{F; D'_1, D_2, \ldots, D_p\}$ generate the same DP-constraint. □

To illustrate Theorem 3.3 and the impact it can have on the number of regular dominoes for a domino-comb, consider the comb $\{H; D_1, D_2, D_3\}$ in K_8 with handle $H = \{2, 4, 6, 7\}$ and dominoes $D_1 = \{1\} \cup \{2\}$, $D_2 = \{3\} \cup \{4\}$ and $D_3 = \{5\} \cup \{6\}$, which is shown in Figure 3(a). For D_1, let $A_1 = \{1\}$, $B_1 = \{2\}$ and $C_1 = \{3, 4, \ldots, 8\}$. Then by switching the compartments for domino D_1, we obtain the domino-combs shown in Figure 3(b) and (c), where respectively we replace D_1 by $D'_1 = A_1 \cup C_1$ and $D''_1 = B_1 \cup C_1$. These all generate the same DP-constraint. However, the domino-comb in (a) has three regular dominoes, whereas the domino-combs in (b) and (c) each have two, since D'_1 and D''_1 are both non-regular.

To improve our closed-set form (16) for CONCORDE, we would like to decide which two compartments should be the inside compartments for each domino in order to maximize the number of dominoes which are regular. Given that there are 3^p possible domino-combs obtainable through compartment switching for a particular DP-constraint with p dominoes, a brute-force approach to finding the one with the maximum number of regular dominoes would in general not be efficient. However, Theorem 3.4 and Corollary 3.5 provide us with an efficient method for solving this problem.

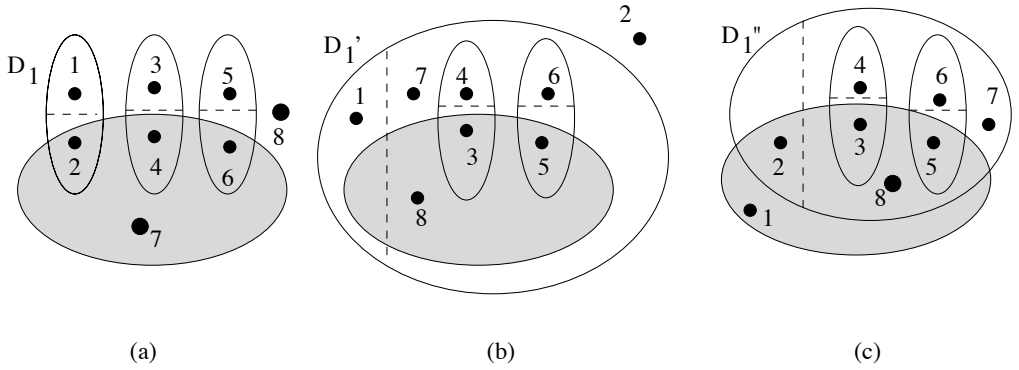

Figure 3: Three equivalent domino-comb constraints

Theorem 3.4 *Let $\{F; D_1, D_2, \ldots, D_p\}$ be a domino-comb with handle H, where $D_i = A_i \cup B_i$ and $C_i = V \backslash D_i$ for $i = 1, 2, \ldots, p$. For each edge $e \in E$, let $\overline{\mu}_e$ be the number of the sets $F, E(A_i : B_i), E(A_i : C_i), E(B_i : C_i)$, $i = 1, 2, \ldots, p$ which contain edge e. Then domino D_k is regular if and only if $\overline{\mu}_e = 1$ for all $e \in E(A_k : B_k)$.*

Proof Suppose D_k is regular. Then recall that, by definition we have

(i) $E(A_k : B_k) \subset \delta(H)$;

(ii) $E(A_k : B_k) \cap E(A_j : B_j) = \emptyset$ for all $1 \leq j \leq p$, $j \neq k$;

(iii) $E(A_k : B_k) \cap \delta(D_j) = \emptyset$ for all $j = 1, 2, \ldots, p$.

Properties (ii) and (iii) imply that any edge $e \in E(A_k : B_k)$ is in only one of the edge sets $E(A_i : B_i), E(A_i : C_i), E(B_i : C_i), i = 1, 2, \ldots, p$, namely the set $E(A_k : B_k)$. Thus, property (i) implies $E(A_k : B_k) \cap F = \emptyset$. It follows that $\overline{\mu}_e = 1$ for all $e \in E(A_k : B_k)$.

Now, suppose $\overline{\mu}_e = 1$ for all $e \in E(A_k : B_k)$. Then clearly properties (ii) and (iii) hold for D_k. Also, we must have that $E(A_k : B_k) \cap F = \emptyset$. It follows that property (i) holds for D_k as well, and thus D_k is regular. □

Note that if we switch the compartments for some domino D_j in a domino-comb $\{F; D_1, D_2, \ldots, D_p\}$, the value of $\overline{\mu}_e, e \in E$, in Theorem 3.4 does not change. Thus, we have the following corollary to Theorem 3.4.

Corollary 3.5 *Let $\{F; D_1, D_2, \ldots, D_p\}$ be a domino-comb and let D_k be a regular domino. Then, D_k is also regular for the domino-comb obtained by switching compartments in domino $D_j, j \neq k$.*

Theorem 3.4 and Corollary 3.5 provide us with an efficient method for maximizing the number of regular dominoes in a domino-comb, thus making the closed-set form (16) much more likely to be accepted by CONCORDE.

Given the domino-comb $\{F; D_1, D_2, \ldots, D_p\}$, we begin by calculating $\overline{\mu}_e, e \in E$ as defined in Theorem 3.4. Then for each domino $D_k = E(A_k : B_k), k = 1, 2, \ldots, p$, we check if $\overline{\mu}_e = 1$ in the semicuts corresponding to any of our three choices for pairs of inside compartments, i.e. $E(A_k : B_k)$, $E(A_k : C_k)$ or $E(B_k : C_k)$, where $C_k = V \backslash D_k$. If so, we choose the corresponding pair of sets from A_k, B_k and C_k to be inside the domino. This may involve switching compartments in domino D_k, but we know from Theorem 3.4 that this does not affect the corresponding DP-constraint, and we know from Corollary 3.5 that this does not affect the regular status of other dominoes in the domino-comb.

Note that after maximizing the number of regular dominoes for our domino-comb, we may still be able to improve on our closed-set form (16) for our DP-constraint. We call a domino *almost-regular* if it satisfies properties (i) and (ii) for regular dominoes, but not property (iii). It can easily be verified that the derivation of the closed-set form (16) is still valid if we let $D_1, D_2, \ldots, D_r$ in (16) represent the regular and almost-regular dominoes. Thus in our testing, after maximizing the number of regular dominoes for our domino-comb, we did a final check of the non-regular dominoes to see if they happened to be almost-regular, in which case we adjusted our set $D_1, \ldots, D_r$ accordingly.

3.1.3 Adding All Violated DP-constraints Found

A final variation that we made in the Letchford algorithm deals with which DP-constraints are identified as violated. In the final step of the algorithm, one looks for a minimum weight odd cycle in an auxillary graph. If this cycle has weight less than 1, then this corresponds to a violated DP-constraint, in fact the one which is violated by the most, and it is this single constraint that is identified by the algorithm. As is pointed out in the paper [16], all odd cycles in this auxillary graph of weight less than 1 correspond to a violated DP-constraint. In our implementation of the algorithm, instead of only identifying just the most-violated DP-constraint, we instead identified the minimum cost odd cycle containing node v for every node $v \in V$, and then kept all the unique DP-constraints from these which corresponded to cycles of weight less than 1. Thus we identified a set of violated DP-constraints rather than just one. Our thinking behind this is that the more violated DP-constraints we generate at each iteration, the faster we will (hopefully) be able to reach an optimal solution.

3.2 Testing Details and Results

We implemented Letchford's DP-constraint separation algorithm along with the enhancements discussed in Section 3.1 in a program called DP-CUTFINDER. A complete description of the details of this implementation can be found in [26].

Recall that the points x^* to which we apply DP-CUTFINDER must satisfy the subtour elimination constraints (3) and the non-negativity constraints (4) , although by the enhancement described in Section 3.1.1 they do not need to have planar support graphs. We chose to satisfy these constraints by ensuring that we stayed inside the polytope $SEP(n)$. An overview of our initial testing process, which we call test T1, is as follows:

Start with $LP = \emptyset$;
Find the optimal solution x^* for the SEP;
Repeat
    Use DP-CUTFINDER to look for DP-constraints which violate x^*;
    Add these constraints to the current LP and reoptimize over the
        LP constraints and the SEP constraints (2)-(5) to obtain a new x^*;
Until one of the stopping criteria is encountered.

There are three possible stopping criteria for our process:

(i) The current x^* is integral (which means that x^* is an optimal solution for the $STSP$).

(ii) The current x^* has a non-planar support graph which was not made planar via the node shrinking heurisitics.

(iii) No violated DP-constraints were found by DP-CUTFINDER for x^*.

Note that if stopping criterion (iii) occurs, it implies that there exist no DP-constraints violated by x^* only in the case that x^* has a planar support graph.

In the above process we used CONCORDE to find the initial optimal solution for the SEP and for the reoptimization step. Our test data consisted of 46 two-dimensional Euclidean $STSP$ instances obtained from TSPLIB [25], a public problem test library for the $STSP$. These 46 problems represent all of the two-dimensional Euclidean problems of size at most 1060 listed in TSPLIB, except for $berlin52$ and $pr107$. Both of these were excluded from testing because their optimal SEP solutions are $STSP$ tours. For all 46 problems the optimal $STSP$ solutions are known. Note that when we refer

to these problems in the test results we use the names given in TSPLIB, which indicate the size of the problem as well as the initials of the creator(s) of the problem.

The first set of results for test T1 are summarized in Table 1, where we compare our results with two other cutting algorithm results we obtained using CONCORDE. CONCORDE does not require files of cuts to be provided from outside the package, as it already contains many different separation routines (exact and heuristic) for different classes of facet-inducing inequalities for $STSP(n)$. The user can select which of these separation routines are turned on and active during the cutting process. In Table 1, DP-CUTFINDER refers to the results we obtain using only DP-CUTFINDER in the method we described, CONC1 refers to the results obtained by running CONCORDE (cutting only, no branching) with all of its internal separation algorithms related to comb constraints and subtour elimination constraints activated, and CONC2 refers to the results obtained by activating the default options for CONCORDE (again with no branching). Note that for the final point x^* found by each of DP-CUTFINDER, CONC1 and CONC2, cx^* provides a lower bound on the value of an optimal solution for the $STSP$, and this lower bound is as good as or better (i.e. bigger) than that provided by optimizing over $SEP(n)$. As in [15], the measure we used for comparison of the lower bounds for the three tests is the ratio

$$R = (100 * (LB - SUBT)/(TOUR - SUBT)),$$

where LB is the lower bound cx^*, $SUBT$ is the optimal solution value for the SEP and $TOUR$ is the optimal tour length for the $STSP$ instance as recorded in TSPLIB. Observe that if $R = 100\%$ then the lower bound obtained is optimal. The ratio R reveals what percentage of the gap between the SEP optimal and the $STSP$ optimal is covered by the lower bound. As can be seen by the results shown in Table 1, on average we covered 95.1% of the gap with DP-CUTFINDER, while CONC1 and CONC2 covered 93.6% and 92.1% of the gap respectively. Thus, on average, using just the DP-constraints and the enhanced Letchford separation algorithm routine did slightly better than using the methods of CONCORDE for this problem set.

Table 1: Lower Bound Comparison

Problem Name	**R value for DP-CUTFINDER**	**R value for CONC1**	**R value for CONC2**
eil51	100	100	100
eil51	100	100	100
st70	100	100	100
eil76	100	100	100
pr76	71.2	71.2	100
rat99	100	100	100
kroA100	100	98.5	100
kroB100	100	100	100
kroC100	100	100	100
kroD100	100	100	96.9
kroE100	100	100	85.6
rd100	100	100	100
eil101	100	100	100
lin105	100	100	100
pr124	97.9	94.5	98.4
bier127	100	100	100
ch130	100	100	100
pr136	93.2	88.5	69.3
pr144	100	100	100
ch150	100	100	92.4
kroA150	98.5	94.5	96.2
kroB150	100	99.4	96.1
pr152	100	100	29.8
u159	100	100	100
rat195	85.6	82.7	81.7
d198	96.3	83.8	47.9
kroA200	100	100	100
kroB200	100	100	95.6
ts225	46.7	59.9	100
tsp225	93.8	92.9	97.0
pr226	100	100	100
gil262	100	95.7	95.1
pr264	100	100	100
a280	92.3	89.7	92.3

continued on next page

Table 1: *continued*

Name	**R value for DP-CUTFINDER**	**R value for CONC1**	**R value for CONC2**
pr299	95.3	92.5	99.4
lin318	100	80.8	81.7
rd400	85.4	82.6	77.8
pr439	64.9	83.8	86.3
pcb442	89.9	86.7	90.1
d493	99.2	86.6	86.9
u574	100	97.6	94.5
rat575	93.3	86.1	86.3
p654	100	100	100
d657	87.1	85.9	91.3
u724	95.0	85.3	85.1
rat783	100	96.1	96.8
u1060	90.4	90.8	87.6

A more detailed look at the results found by DP-CUTFINDER for test T1 can be seen in Table 2. The first two columns contain the problem name and the number of times CONCORDE was invoked for the reoptimization step. The problems whose names are shown in italics (30 out of the 46) are those for which we reached optimality. The third column refers to the iteration(s) where the reoptimization step yielded a solution x^* with a non-planar support graph, and node shrinking heuristics had to be applied before searching for violated DP-constraints. The fourth column contains the total number of DP-constraints added to the problem, reported as two numbers; the first number refers to the DP-constraints that were comb constraints, while the second number is the number of DP-constraints that were not comb constraints. In the final column we report the total number of the added DP-constraints which were tight for the final LP solution x^* (i.e. satisfied at equality by x^*), again separated into those that were comb constraints and those that were not. These represent the DP-constraints that are required and necessary for the final LP and solution x^*.

Table 2: DP-CUTFINDER Results

Name	**# Iter CONC**	**Iter Non-planar**	**# DP (C/NonC)**	**# Tight DP (C/NonC)**
eil51	7	—	(40/10)	(32/7)
st70	4	—	(15/3)	(9/2)
eil76	5	—	(19/1)	(8/0)
pr76	5	4,5	(31/13)	(22/7)
rat99	7	—	(19/5)	(11/1)
kroA100	7	—	(23/9)	(22/9)
kroB100	9	—	(45/6)	(14/3)
kroC100	5	—	(28/2)	(8/2)
kroD100	4	—	(27/4)	(26/4)
kroE100	10	5,6,7,8	(87/29)	(69/22)
rd100	3	—	(17/4)	(15/4)
eil101	3	—	(17/5)	(15/4)
lin105	2	—	(1/0)	(1/–)
pr124	11	3,4,5,9,10,11	(12/0)	(12/–)
bier127	7	3,6	(51/11)	(42/8)
ch130	9	—	(60/11)	(52/11)
pr136	10	3,6,7	(130/30)	(94/8)
pr144	3	—	(18/10)	(13/10)
ch150	7	—	(91/15)	(74/9)
kroA150	10	—	(98/31)	(65/17)
kroB150	8	—	(90/17)	(84/14)
pr152	5	—	(37/8)	(31/6)
u159	5	—	(14/0)	(9/–)
rat195	11	3,5,10,11	(196/69)	(44/13)
d198	11	—	(158/41)	(82/20)
kroA200	10	—	(163/88)	(133/83)
kroB200	7	—	(83/12)	(69/8)
ts225	6	3,4,5,6	(51/4)	(10/0)
tsp225	13	—	(261/94)	(221/83)
pr226	4	—	(14/2)	(14/2)
gil262	15	3,5-9,11,13-15	(310/201)	(34/16)
pr264	5	—	(23/3)	(23/3)
a280	10	—	(123/31)	(71/23)
pr299	14	5-14	(229/126)	(44/16)

continued on next page

Table 2: *continued*

Name	**# Iter CONC**	**Iter Non-planar**	**# DP (C/NonC)**	**# Tight DP (C/NonC)**
lin318	8	3,4,5,6	(157/113)	(104/83)
rd400	16	4,5,6,7,9,11-16	(425/285)	(247/208)
pr439	6	2,3,4,6	(131/35)	(16/2)
pcb442	16	4,5,6	(265/40)	(133/16)
d493	45	34,40	(1505/800)	(408/120)
u574	12	9,11	(507/393)	(464/365)
rat575	16	—	(711/406)	(495/259)
p654	3	—	(18/8)	(6/0)
d657	12	5,6,8-12	(685/327)	(364/172)
u724	16	5,7-11,13-16	(892/591)	(602/356)
rat783	13	11,12	(752/249)	(490/102)
u1060	16	8,9,15,16	(674/218)	(144/10)

In looking at the results in column 3 of Table 2, we can see that for just under half of the problems, non-planar support graphs were encountered along the way. This demonstrates the importance of our enhancement to Letchford's algorithm which allows us to shrink in order to continue using the algorithm to move towards a better (i.e. larger) lower bound, even when non-planar points where encountered. In fact, for six of the problems where non-planar graphs arose, optimality was eventually reached.

In looking at the results in columns 4 and 5 of Table 2, it is worth noting that for every problem, more combs were found by DP-CUTFINDER than non-combs. On average, of the total number of violated DP-constraints found, 21.9 % were non-combs. Also note that of the tight DP-constraints, an average of 20.8 % were non-combs. Thus non-comb DP-constraints were definitely required and active for our final solutions.

In addition to those just described, we carried out another test, test T2. For this test we ran CONCORDE without DP-CUTFINDER to obtain the CONC2 lower bound (as previously described) and the assoicated final x-vector, then we ran DP-CUTFINDER just once on this final x-vector (if it was not already optimal). After that, we used CONCORDE once to reoptimize over the LP consisting of just the DP-constraints found by DP-CUTFINDER and the SEP constraints. The goal of this test was to see what kind of improvement the DP-constraints generated by our program might

have on the lower bound CONC2 obtained by CONCORDE, and to see how often DP-CUTFINDER could "cut-off" the final x-vector found for CONC2. Note that CONCORDE was not able to find any violated constraints for this x-vector using its default separation routines, which in particular included its separation heuristics for comb constraints.

The results are shown in Table 3. Column 2 lists the optimal tour value, column 3 lists the lower bound CONC2 value and column 4 lists the new lower bound obtained using the new DP-constraints found by DP-CUTFINDER (note that "OTL" denotes that the optimal tour value was reached). Column 5 lists the number of violated DP-constraints that were found in one iteration of DP-CUTFINDER and column 6 lists the number of these which were tight. For columns 5 and 6 the numbers are once again partitioned into the number of comb and non-comb DP-constraints.

As can be seen from the table, DP-CUTFINDER was able to "cut-off" and thus increase the lower bound for 22 if the 26 problems tested, although the average increase in the ratio R was only 3.5 %. There were four exceptions for which the lower bound did not increase. For $tsp225$ and $a280$, DP-CUTFINDER found no violated constraints. For $pcb442$ and $rat783$, adding the violated DP-constraints found by DP-CUTFINDER had no effect on the lower bound.

Another interesting observation from Table 3 deals specifically with the problems $kroD100$, $kroB200$ and $pr439$. For all three of these points there was an increase in the lower bound, yet none of the violated DP-constraints we found for these problems were tight for the final solution. At first this seemed quite puzzling, so we checked our DP-constraints by hand, and verified that they were indeed violated by the x-vector from CONC2, and were not tight for the final x-vector which gave the increased lower bound. We suspect that these anomolies were caused by the necessary preprocessing of LPs by CONCORDE, as we know that CONCORDE sometimes ignores some of the cuts in a cut file (when solving the LP) if they do not appear to be useful on a sparser version of the graph.

We did three more tests, T3, T4 and T5, this time only for the problem $lin318$. Tests T3 and T4 were slight variations of our original test process, test T1. In test T3, only violated comb constraints found by DP-CUTFINDER were added to the constraints file. The motivation behind this test was to determine if restricting the kinds of DP-constraints being added to be only comb constraints would help us in maintaining planarity of our

Table 3: Improving on CONC2

Problem Name	**Tour Value**	**CONC2 Value**	**After DP-CUT**	**# DP (C/NonC)**	**# Tight DP (C/NonC)**
kroD100	21294	21289.333	21290.800	(2/0)	(0/–)
kroE100	22068	22029.250	22038.909	(9/0)	(6/–)
pr124	59030	59014.571	59025.639	(12/0)	(11/–)
pr136	96772	96514.667	96603.600	(10/0)	(10/–)
ch150	6528	6525.125	OTL	(9/9)	(2/0)
kroA150	26524	26515.417	26517.667	(13/2)	(12/2)
kroB150	26130	26114.500	OTL	(19/13)	(19/13)
pr152	73682	73349.500	73362.300	(4/0)	(2/–)
rat195	2323	2318.660	2319.454	(3/0)	(2/–)
d198	15780	15741.167	15756.250	(15/2)	(12/0)
kroB200	29437	29425.000	29431.708	(2/0)	(0/–)
tsp225	3916	3914.880	—	(–/–)	(–/–)
gil262	2378	2376.857	2377.000	(4/0)	(4/0)
a280	2579	2578.000	—	(–/–)	(–/–)
pr299	48191	48186.500	48188.000	(1/0)	(1/–)
lin318	42029	42003.310	42021.300	(18/40)	(17/19)
rd400	15281	15253.500	15255.067	(7/0)	(1/–)
pr439	107217	107040.503	107055.051	(5/2)	(0/0)
pcb442	50778	50750.333	50750.333	(4/0)	(4/–)
d493	35002	34979.272	34985.525	(50/42)	(15/3)
u574	36905	36894.587	36897.629	(30/9)	(12/1)
rat575	6773	6766.310	6767.671	(28/5)	(4/1)
d657	48912	48872.117	48877.603	(30/56)	(3/0)
u724	41910	41871.685	41873.843	(15/1)	(2/0)
rat783	8806	8804.932	8804.932	(3/0)	(1/–)
u1060	224094	223914.463	223921.907	(24/7)	(15/2)

support graphs. For this particular example we found that this was not the case.

In test T4, only the most violated inequalities were added to the constraints file at each iteration, just as Letchford had originally set out in [16]. The purpose of this test was to see if our use of all the violations found per iteration of DP-CUTFINDER, as opposed to only the one violated by the most, was worthwhile. For this particular problem, it seems that adding all the violations did greatly decrease the number of iterations of CONCORDE required; when adding only the maximally violated inequalities, 26 iterations of CONCORDE were required as opposed to the 8 iterations required in our test T1.

In test T5, we did a test similar to test T2, but the code ensuring that the final solution satisfied all the subtour constraints was removed. It had been suggested to us by the makers of CONCORDE that CONCORDE's results were often better if the condition that the subtour constraints had to be satisfied was relaxed. For *lin*318, this was indeed true; test T5 yielded a lower bound of 42,014.429, while our original test T2 resulted in a lower bound of 42,003.310. So a slightly higher lower bound was achieved.

3.3 Summary of Test Results

We feel that our results demonstrate the practical usefulness of the *DP*-constraints and the enhanced Letchford separation algorithm for solving *STSP*s within a branch and cut framework. In test T1, using only *DP*-constraints and DP-CUTFINDER performed very well in terms of solution quality when compared to the solutions produced by a state-of-the-art tool such as CONCORDE. Moreover, non-comb *DP*-constraints were definitely required and active for our final solutions. Also, in test T2, we were able to cut off the final solutions obtained by CONCORDE (without branching) for all but 2 of the 46 problems tested, and both violated comb and non-comb constraints were found. Thus, in particular, the enhanced Letchford algorithm was able to find violated comb constraints missed by the heuristic comb separation algorithms used by CONCORDE.

We also feel that the test results demonstrated the usefulness of the enhancements we made to the Letchford algorithm. In particular, for just under half of the 46 problems tested, non-planar support graphs were encountered, often very early in the cutting-plane process. In these cases we were able to use our shrinking algorithm to continue moving towards a better lower bound. Thus being able to shrink and use the algorithm for non-planar support graphs was a very useful and important enhancement for the Letchford algorithm.

4 Domino-combs which have exactly one non-regular domino

Recall from Section 3.1.2 that when all the dominoes in a domino-comb are regular, we have a comb, and the corresponding DP-constraint is known to be facet-inducing. In this section we consider all domino-combs which have exactly one non-regular domino, and characterize those which are facet-inducing.

We begin this section by describing a certain subclass of the domino-combs which have exactly one non-regular domino.

Definition A *twisted comb* is a domino-comb with $p-1$ ($p \geq 3$) regular dominoes $D_1, \ldots, D_{p-1}$ and one non-regular domino, D_p, which contains at least two regular dominoes in each of its three compartments A_p, B_p and C_p. The corresponding DP-constraint is called a *twisted comb constraint*. If the regular dominoes in a twisted comb are all 2-matching dominoes (that is, $|A_i| = |B_i| = 1$ for $1 \leq i \leq p-1$), then it is called a *2-matching twisted comb*, and the corresponding constraint is a *2-matching twisted comb constraint*.

Three simple examples of 2-matching twisted combs are illustrated in Figure 4. Note that the 2-matching twisted comb shown in (c) can be obtained from the one shown in (b) by switching compartments in domino D_p, as discussed in Theorem 3.3.

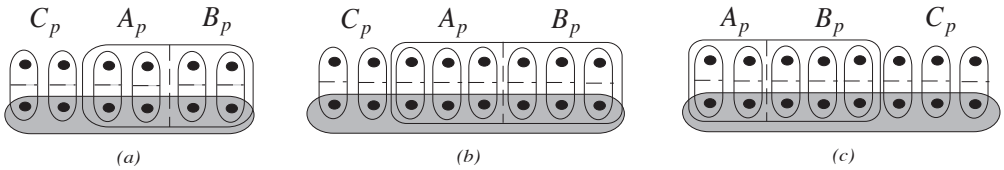

Figure 4: 2-matching twisted combs

The twisted comb constraints were first introduced in [2] where they were shown to be facet-inducing for STSP(n). The main result of this section will be to characterize the facet-inducing domino-comb constraints which have exactly one non-regular domino by showing that any DP-constraint which has exactly one non-regular domino is facet-inducing if and only if it is either a twisted comb constraint or else equivalent to a comb inequality.

In the proofs that follow we will need to illustrate many tight tours in K_n. A *tour* is the 0-1 incidence vector $x^* \in \mathbf{R}^E$ of a Hamilton cycle in K_n. A tour x^* is called *tight*, or *a-tight* for an inequality $ax \geq \alpha$ (or $ax \leq \alpha$)

if $ax^* = \alpha$. The following proposition will make it easier to recognize tight tours for the twisted comb inequalities.

Proposition 4.1 *Let x^* be a tour in K_n, let $\{F; D_1, \ldots, D_p\}$ be a domino-comb in K_n, and let $\mathcal{S}$ represent the set of all node subsets A_i, B_i and D_i for $i = 1, 2 \ldots, p$. Then a tour x^* is tight for the DP-constraint corresponding to $\{F; D_1, \ldots, D_p\}$ if and only if x^* satisfies one of the following:*

(1) $x^*(\delta(S)) = 2$ for each subset $S \in \mathcal{S}$, and $x^*_e = 1$ for exactly one edge in F, or

(2) x^* uses no edges in F (i.e. $x^*_e = 0$ for all $e \in F$), and $x^*(\delta(S)) = 2$ for all $S \in \mathcal{S}$ except one set S' for which $x^*(\delta(S')) = 4$.

Proof Consider the DP-constraint in form (7). It can be rewritten as

$$\frac{1}{2}\sum_{i=1}^{p} x(\delta(A_i)) + \frac{1}{2}\sum_{i=1}^{p} x(\delta(B_i)) + \frac{1}{2}\sum_{i=1}^{p} x(\delta(D_i)) + x(F) \geq 3p + 1.$$

Since any tour x^* satisfies $x^*(\delta(S)) \geq 2$ for any node subset S and $x^*_e \geq 0$ for all $e \in E$, the result follows. □

5 Distinctness of the Twisted Comb Constraints

In this section, we will show that the family of twisted comb constraints is distinct from other well-known families of facet-inducing inequalities for $STSP(n)$. We note that Caprara, Fischetti and Letchford [4] have already verified this for the single twisted comb constraint corresponding to the twisted comb shown in Figure 4(a).

Much of our argument is based on a result of Naddef and Rinaldi from 1992 [19]. First we require two definitions from that paper.

Definition [19] An inequality constraint $c^T x \geq \zeta$ is in *tight triangular form* (or *TT-form*) if and only if

(1) the coefficients c_e satisfy the triangle inequality, *i.e.*, $c_{uv} \leq c_{uw} + c_{vw}$ for any set of three distinct nodes u, v and w;

(2) for every node w, there exist $u, v \in V \backslash \{w\}$ for which the triangle inequality is tight, *i.e.*, $c_{uv} = c_{uw} + c_{vw}$.

Notice that tight triangularity depends only on the edge coefficients c_e, and not on the right hand side ζ of the constraint. As an example, it is not difficult to show that the subtour elimination constraint $x(\delta(S)) \geq 2$ satisfies the triangle inequality and is in tight triangular form for any node subset S satisfying $2 \leq |S| \leq |V| - 2$.

Definition [19] A tight triangular inequality $c^T x \geq \zeta$ is *simple* if $c_e > 0$ for all $e \in E$. If $c^T x \geq \zeta$ is not simple, then we can partition the node set $V = V_1 \cup \cdots \cup V_k$ so that

(1) $c_e = 0$ for all $e \in \gamma(V_i)$, $1 \leq i \leq k$;

(2) $c_e = c_f$ for all $e, f \in E(V_i : V_j)$, $1 \leq i < j \leq k$.

The *simple inequality associated with* $c^T x \geq \zeta$ is obtained by collapsing each partite V_i into a single node i. More precisely, it is the inequality $(\tilde{c})^T \tilde{x} \geq \zeta$ on $STSP(k)$ defined by $\tilde{c}_{ij} = c_e$,where $e \in E(V_i, V_j)$.

Note that $(\tilde{c})^T \tilde{x} \geq \zeta$ is tight triangular if and only if $c^T x \geq \zeta$ is tight triangular.

Lemma 5.1 *The TT-form of the twisted comb constraint is form (9), i.e.*

$$\sum_{i=1}^{p} x(E(A_i : B_i)) + \sum_{i=1}^{p} x(E(A_i : C_i)) + \sum_{i=1}^{p} x(E(B_i : C_i)) + x(F) \geq 3p + 1.$$

Proof We begin by finding the simple inequality associated with the twisted comb constraint, which is an underlying 2-matching twisted comb constraint.

Let S be a node subset that is entirely contained within one compartment of *every* domino, and on one shore of the handle; then any edge $e \in \gamma(S)$ is not in $E(A_i : B_i)$, $E(A_i : C_i)$ nor $E(B_i : C_i)$ for any $1 \leq i \leq p$, nor in F. Thus, $c_e = 0$ for all $e \in \gamma(S)$. In particular, this applies to $S = A_i$ or B_i, $1 \leq i \leq p - 1$ (of course, it applies trivially if any of these sets is a singleton). It also applies to any set of nodes that is entirely contained within *one* of the six locations indicated with open circles in Figure 5. We denote these sets by $V_1, \ldots, V_6$; note that one or more of these may be empty. Moreover, $c_e = c_f$ for all $e, f \in E(S_1 : S_2)$, where S_1, S_2 are non-empty sets in $\{A_1, \ldots, A_{p-1}, B_1, \ldots, B_{p-1}, V_1, \ldots, V_6\}$. Partition the node set as

$$V = (A_1 \cup B_1) \cup \cdots \cup (A_{p-1} \cup B_{p-1}) \cup V_1 \cup \cdots \cup V_6,$$

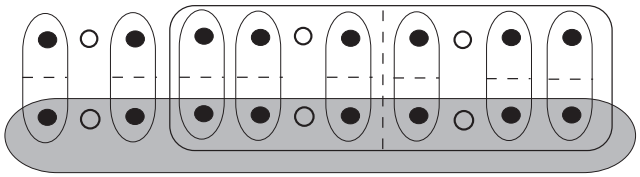

Figure 5: Locations for node sets $V_1, \ldots, V_6$

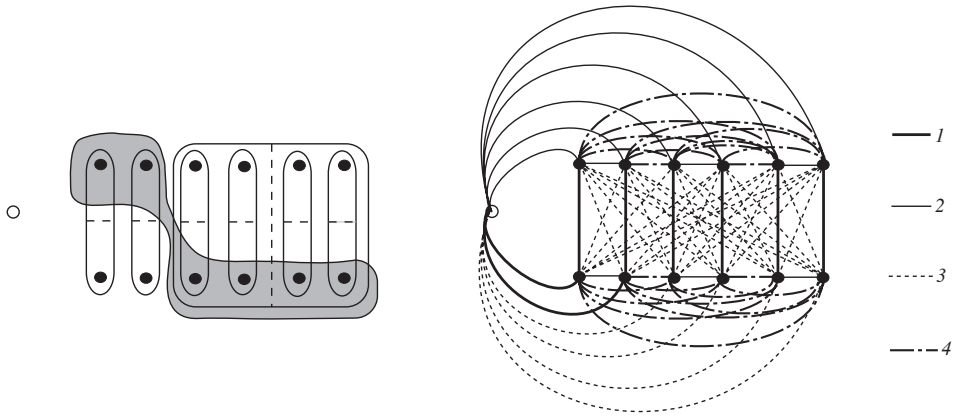

Figure 6: Edge coefficients for a 2-matching twisted comb constraint

then collapse each A_i and B_i, $1 \leq i \leq p-1$, and each non-empty V_j. Note that this simple inequality associated with the twisted comb constraint is a 2-matching twisted comb constraint.

We next demonstrate that the 2-matching twisted comb constraints in form (9) are tight-triangular. Figure 6 shows a simple 2-matching twisted comb and the corresponding edge coefficients of this form of the constraint. The unusual configuration of the handle gives an edge coefficient diagram that better displays the symmetry with respect to the compartments of the irregular domino.

It is straighforward to verify from this diagram that the edge coefficients are all positive and satisfy the triangle inequality, and that moreover each node is part of a tight triangle. This example can clearly be generalized to all 2-matching twisted combs.

Since the simple inequality associated with the twisted comb constraints is tight triangular, it follows that the general twisted comb constraints are tight triangular also. □

Next we require an additional fact from [19] about the tight triangular

form of inequalities on $STSP(n)$.

Proposition 5.2 (Naddef and Rinaldi, 1992) *Let $c^T x \geq \zeta$ be a facet-inducing constraint on $STSP(n)$. An inequality $d^T x \geq \delta$ equivalent to $c^T x \geq \zeta$ is tight triangular if and only if $d^T = \gamma c^T + \lambda^T A$ and $\delta = \gamma\delta + \lambda^T \mathbf{2}$ where $\gamma > 0$ and $\lambda \in \mathbf{R}^n$ satisfy, for each $u \in V(K_n)$,*

$$\lambda_u = \frac{1}{2}\gamma \max\{c_{vw} - c_{uv} - c_{uw} \mid v, w \in V(K_n)\backslash\{u\},\ v \neq w\}.$$

A consequence of this proposition is that *any* facet-inducing inequality on $STSP(n)$ can be put into TT-form, and moreover, this form is unique up to multiplication by a positive scalar. This can be used to determine when two facet-inducing constraints are distinct.

Proposition 5.3 *The twisted comb constraints are distinct from all of the SEP inequalities (3),(4) and (5) i.e. the subtour elimination constraints, the non-negativity constraints and the upper bound constraints .*

Proof The usual form of the upper bound constraint $x_{uv} \leq 1$ where $uv \in E(K_n)$ is not tight triangular. This inequality can, in fact, can equivalently be written as the subtour elimination constraint $x(\delta(S)) \geq 2$ for $S = \{u, v\}$. As noted earlier in this section, the subtour elimination constraints $x(\delta(S)) \geq 2$ are in TT-form for any node subset S satisfying $2 \leq |S| \leq |V| - 2$. Note that the coefficient of each edge in this constraint is either 0 or 1. From Figure 6 we can see that the positive edge coefficients in the TT-form of a twisted comb constraint take on four different values (namely, 1 through 4). Therefore, the twisted comb constraints are distinct from the subtour elimination constraints and upper bound constraints.

The usual form of the non-negativity constraint, $x_{uv} \geq 0$, where $uv \in E(K_n)$ is not tight triangular. It can be put into TT-form by adding one copy of the degree equation (2) for each $w \in V(K_n)\backslash\{u, v\}$; this yields $c^T x \geq 2(n-2)$, where $c_e = 1$ if $e \in \delta(\{u, v\})$, and $c_e = 0$ otherwise. Since the edge coefficients take on only two different values, we may conclude that the twisted comb constraints are distinct from the non-negativity constraints. □

The nontriviality of the twisted comb constraints is not surprising. Given the derivation of the general DP-constraints given by Letchford in [16], it seems more probable that they belong to the large family of constraints whose definition involves node subsets that can be separated into handles and teeth. The comb constraints are the prototypes of this family; generalizations include the *clique tree* constraints, the *path, wheelbarrow* and *bicycle*

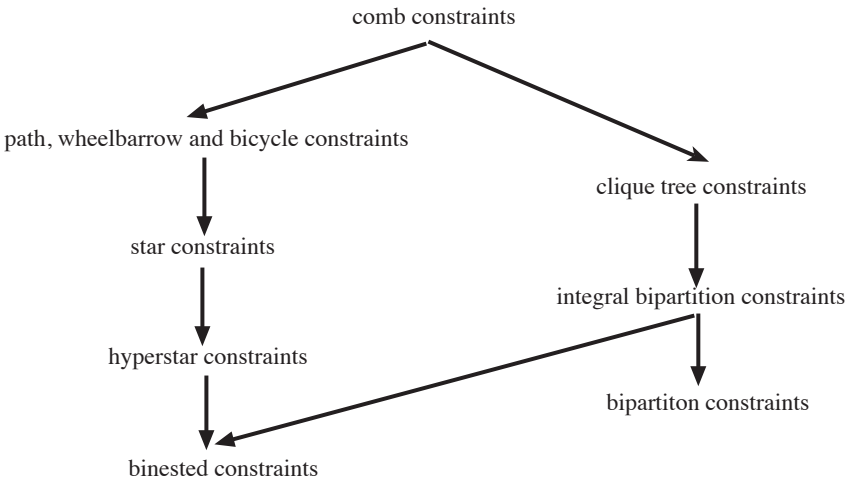

Figure 7: Inclusion diagram for $STSP(n)$ facet-inducing constraints

constraints, the *star* and *hyperstar* constraints, the *bipartition* constraints, and the *binested* constraints. For a survey of these constraints, see [17]. In Figure 7 we reproduce a diagram from that paper indicating inclusions among these sets of constraints.

We will show next that the twisted comb constraints are not in the family of binested constraints. The complete definition of this family is complicated and can be found in [17]; we include here only what is needed for our proof.

Definition A set of subsets $\{S_1, \ldots, S_s\}$ is *nested* if for any pair of sets $S_i \neq S_j$, either $S_i \subset S_j$ or $S_j \subset S_i$ or $S_i \cap S_j = \emptyset$.

A binested inequality is defined on two nested sets of node sets, handles $\mathcal{H} = \{H_1, \ldots, H_h\}$ and teeth $\mathcal{T} = \{T_1, \ldots, T_t\}$, together with corresponding positive integers $\{\alpha_1, \ldots, \alpha_h\}$ and $\{\beta_1, \ldots, \beta_t\}$ (called the *multiplicities* of the handles and teeth respectively), which must satisfy a number of conditions. The TT-form of the binested constraint is

$$\sum_{i=1}^{h} \alpha_i x(\delta(H_i)) + \sum_{j=1}^{t} \beta_j x(\delta(T_j)) \geq 2\Gamma(\mathcal{H}, \mathcal{T}),$$

where $\Gamma(\mathcal{H}, \mathcal{T})$ is a positive integer which is a function of the handles, the teeth, and their multiplicities. Notice that the edge coefficients are determined solely by the cuts associated to the binested handles and teeth (together with their multiplicities).

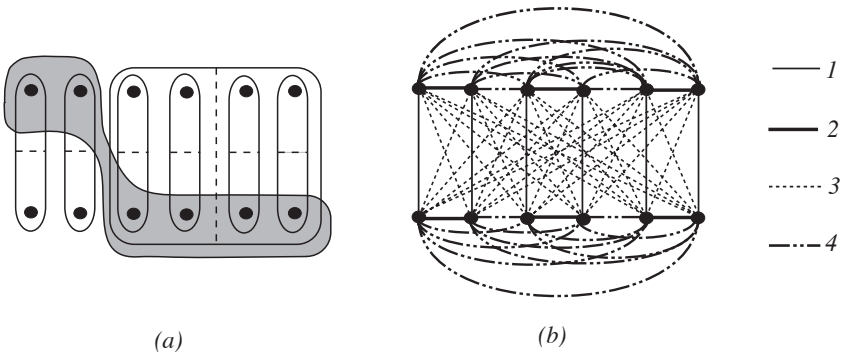

Figure 8: Coefficients for the 12-node twisted comb constraint

At first glance, the twisted comb constraints do not appear to fit the pattern of the binested inequalities. To prove this, we consider the 12-node twisted comb in Figure 8(a), which is contained as a subset of any twisted comb. The edge coefficients in the associated constraint are shown in Figure 8(b); this pattern must appear in any constraint associated with any twisted comb.

First we require a couple of technical lemmas regarding tight triangles.

Lemma 5.4 *Let $c^T x \geq \zeta$ be the TT-form of a binested inequality on $STSP(n)$, and let $S \subset V(K_n)$ be a handle or a tooth. If x, y and z are distinct nodes in $V(K_n)$ satisfying $c_{xy} + c_{yz} = c_{xz}$, then y must be on the same shore of $\delta(S)$ as x or z (or both).*

Proof For any handle or tooth S, the cut $\delta(S)$ either has all three nodes on the same shore, or has two nodes on one shore and the remaining node on the opposite shore. Note that in the first case, the cut contributes nothing to the edge coefficients c_{xy}, c_{yz} and c_{xz}. Let ψ denote the sum of the multiplicities of all handles and/or teeth whose cuts have x on one shore and y, z two on the opposite shore; similarly, let ρ represent the sum of the multiplicities of all node sets whose cuts isolate y from the other two, and σ, the sum of all multiplicities of node sets whose cuts isolate z. Then ψ, ρ and σ are non-negative integers that satisfy

$$c_{xy} = \psi + \rho; \quad c_{yz} = \rho + \sigma; \quad c_{xz} = \sigma + \psi.$$

Since $c_{xy} + c_{yz} = c_{xz}$, we must conclude that $\rho = 0$; in other words, there is no handle or tooth whose cut separates y from $\{x, z\}$.□

If $c_{xy} + c_{yz} = c_{xz}$, we will call xz the *hypotenuse* of the tight triangle.

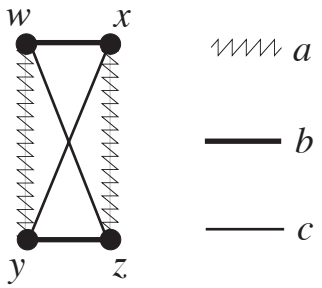

Figure 9: A tight square with $a = b + c$

Corollary 5.5 *Let $c^T x \geq \zeta$ be the TT-form of a binested inequality on STSP(n), and let $S \subset V(K_n)$ be a handle or a tooth of this inequality. If w, x, y and z are distinct nodes in $V(K_n)$ satisfying $c_{xy}+c_{yz} = c_{xw}+c_{wz} = c_{xz}$ and $c_{wx} + c_{xy} = c_{wz} + c_{yz} = c_{wy}$, then either*

(1) all four nodes are on the same shore of $\delta(S)$, or

(2) $\{x, w\}$ and $\{y, z\}$ are on opposite shores, or

(3) $\{x, y\}$ and $\{w, z\}$ are on opposite shores.

Proof In Figure 9, we give an example of four nodes satisfying the conditions of the corollary, under the assumption that $a = b + c$.

By Lemma 5.4,

(a) w must be on the same shore of $\delta(S)$ as x or z,

(b) y must be on the same shore as x or z,

(c) x must be on the same shore as w or y, and

(d) z must be on the same shore as w or y.

Suppose that it is not the case that all four nodes are on the same shore. Conditions (a) through (d) imply that $\delta(S)$ cannot separate one of these nodes from the other three; they also imply that we cannot have $\{x, z\}$ on one shore and $\{w, y\}$ on the other. □

We will call a set of four nodes whose edge coefficients satisfy the conditions of Corollary 5.5 a *tight square*; the hypotenuses of the tight square are the two hypotenuses of the four tight triangles within it. In Figure 9, for example, $\{x, z\}$ and $\{w, y\}$ are the hypotenuses. Corollary 5.5 says that the endnodes of one hypotenuse of a tight square cannot appear in the opposite shore from the endnodes of the other hypotenuse.

Theorem 5.6 *The twisted comb constraints are distinct from the binested inequalities.*

Proof Assume that there exists a twisted comb constraint, $c^T x \geq \zeta$, which is equivalent to a binested constraint, $\tilde{c}^T x \geq \tilde{\zeta}$. As noted earlier, any twisted comb contains a subset of twelve nodes $V_0 = \{p, q, r, s, t, u, p', q', r', s', t', u'\}$ whose edge coefficients in the TT-form of the corresponding constraint are as shown in Figure 10. For clarity, we have omitted edges that are hypotenuses of tight triangles.

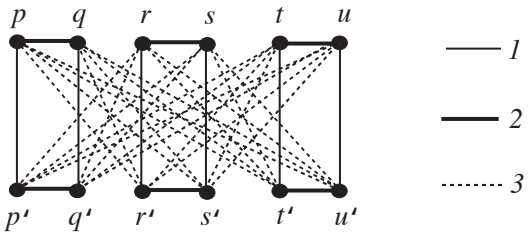

Figure 10: Diagram for V_0

In order for this figure to represent the edge coefficients of (some positive multiple of) the TT-form of a binested inequality, every pair of nodes in V_0 must be separated by the cut of at least one tooth or handle. In particular, there must be at least one node set H whose cut separates p and p'; without loss of generality, we can assume H is a handle. Since the nodes $\{p, q, p', q'\}$ form a tight square with hypotenuses pq' and $p'q$, by Corollary 5.5 $\{p, q\}$ and $\{p', q'\}$ must be on opposite shores of $\delta(H)$; without loss of generality, we assume $\{p, q\} \subset H$. Now, $\{q, r, q', r'\}$ is also a tight square, with hypotenuses qr and $q'r'$; since it cannot be the case that q is the only node of this square appearing in H, and we already know that $q' \in \overline{H}$, Corollary 5.5 implies that $r' \in H$ and $r \in \overline{H}$. Repeating this argument progressively with the tight squares $\{r, s, r', s'\}$, $\{s, t, s', t'\}$ and $\{t, u, t', u'\}$, we conclude that $\{p, q, r', s', t, u\} = H \cap V_0$ and $\{p', q', r, s, t', u'\} = \overline{H} \cap V_0$ (see Figure 11).

Notice that $\delta(H)$ separates x from x', for all $x \in W_0 = \{p, q, r, s, t, u\}$. Moreover, if we had begun by looking for a node set whose cut separates any x from x', the tight squares (considered in an appropriate sequence) would force the intersection of this set (or its complement) with V_0 to be precisely

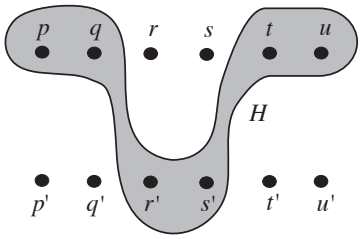

Figure 11: Nodes in handle H

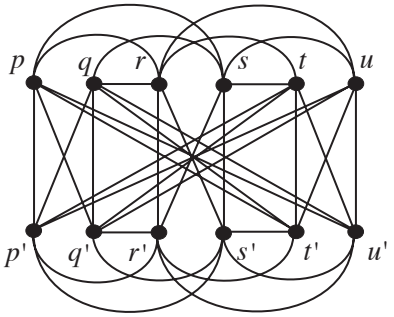

Figure 12: Edges which get a contribution of α

$H \cap V_0$. Hence, for each $x \in W_0$, $H \cap V_0$ is the *unique* subset of V_0 (up to complements) whose cut separates x from x'. In particular, since the handles in a binested constraint are nested, this implies that *any* handle H' satisfies $H' \cap V_0 \in \{\emptyset, H \cap V_0, \overline{H} \cap V_0, V_0\}$. Let α denote the sum of the multiplicites of all handles in $V(K_n)$ whose whose cut contributes to the coefficients of edges in $\gamma(V_0)$; this accounts for a contribution of α to the coefficients $\tilde{c}_e$ of the edges shown in Figure 12.

To account for the positive edge coefficients on the edges of $\gamma(V_0)$ not appearing in Figure 12, we must have other node sets, which, since they cannot be handles, must be teeth. If T is such a tooth, $\delta(T)$ cannot separate x from x', for any $x \in W_0$; if it did, then by the argument in the previous paragraph, $T \cap V_0 = H \cap V_0$ (or $\overline{T} \cap V_0 = H \cap V_0$), and thus $\delta(T)$ would not contribute to the edges with zero coefficient in Figure 12. Hence, $T \cap V_0$ must consist of one or more of the pairs $\{p, p'\}$, $\{q, q'\}$, $\{r, r'\}$, $\{s, s'\}$, $\{t, t'\}$ and $\{u, u'\}$. Let $\beta > 0$ denote the sum of the multiplicities of all teeth in $V(K_n)$ whose cuts separate $\{p, p'\}$ and $\{u, u'\}$. Then

$$\tilde{c}_{pu} = \beta < \alpha + \beta = \tilde{c}_{p'u}.$$

From Figure 10, however, we see that in the twisted comb constraint,

$$c_{pu} = 4 > 3 = c_{p'u}.$$

Since this disparity cannot be resolved with a positive scalar multiple, this contradicts our assumption that the twisted comb constraint is equivalent to a binested constraint. □

Note that it follows from Theorem 5.6 that the twisted comb constraints are not equivalent to any of the families of facet-inducing constraints for $STSP(n)$ which are shown in Figure 7, since they are contained in the binested constraints (the only exception are the non-integral bipartition inequalites, which are not known to be facet-inducing for $STSP(n)$ in general). Other families of nontrivial, facet-inducing constraints for $STSP(n)$ which do not belong to the set of binested inequalities are the *chain* constraints, the *crown* constraints and the *ladder* constraints. Showing that the twisted comb constraints do not belong to these families is comparatively straightforward.

It is shown in [6] that the TT-form of the associated simple inequality for the chain constraint has 1, 2 and 3 as the only possible edge coefficients, whereas a simple twisted comb constraint in TT-form has four different positive edge coefficients.

The simple crown constraints are defined on $STSP(4k)$, for $k \geq 1$. For any labelling of the nodes as $1, 2, \ldots, 4k$, the associated simple crown inequality is $c^T x \geq 12k(k-1) - 2$, where for any $v \in V(K_{4k})$,

$$c_{v,v+j} = \begin{cases} 4k - 6 + |j|, & 1 \leq |j| \leq 2k-1, \\ 2(k-1), & j = 2k. \end{cases}$$

This constraint is shown to be valid and facet-inducing for $STSP(4k)$, and in TT-form, by Naddef and Rinaldi in [20]. Even for $k = 1$, there are five different positive edge coefficients.

The ladder inequalities, introduced by Boyd, Cunningham, Queyranne and Wang in 1995 [3], are defined on two handles and a positive even number of teeth, which satisfy different conditions than the binested inequalities. Also, unlike the binested constraints but like the twisted comb constraints, the left hand side of the TT-form of the ladder constraints includes $\sum \alpha_i x(\delta(H_i)) + \sum \beta_j x(\delta(T_j))$ plus some additional terms. In [3], it is shown that these constraints have Chvátal rank 2, and so they cannot be equivalent to the twisted comb constraints which, as shown in [6], have Chvátal rank exactly 1.

Acknowledgments

We would like to gratefully thank David Applegate, Bill Cook, Vašek Chvátal and Robert Bixby for allowing us to use CONCORDE for our testing, and give a special thanks to Bill Cook and Sanjeeb Dash who were instrumental in helping us adapt CONCORDE to suit our needs. We are also grateful to Adam Letchford, Lisa Fleischer and Denis Naddef for their helpful suggestions regarding this work.

References

[1] D. Applegate, R. Bixby, V. Chvátal and W. Cook, "Finding cuts in the TSP (A Preliminary Report)," DIMACS Technical Report 95-05, Rutgers University, New Brunswick, NJ, 1995.

[2] S. Boyd and S. Cockburn, "A family of facet-inducing domino-parity inequalities for the $STSP$," Technical Report, University of Ottawa, 2001.

[3] S. Boyd, W. H. Cunningham, M. Queyranne and Y. Wang, "Ladders for travelling salesmen," SIAM Journal on Optimization 5, pp. 408-420, 1995.

[4] A. Caprara, M. Fischetti and A. Letchford, "On the separation of maximally violated mod-k cuts," Mathematical Programming 87, pp. 37-56, 2000.

[5] R. Carr, "Separating clique trees and bipartition inequalities having a fixed number of handles and teeth in polynomial time," Mathematics of Operations Research 22-2, pp. 257-265, 1997.

[6] S. Cockburn, On DP-constraints for the Traveling Salesman Polytope, Master's Thesis, University of Ottawa, Ottawa, 2001.

[7] W. Cook, W. Cunningham, W. Pulleyblank and A. Schrijver, Combinatorial Optimization, John Wiley and Sons, Inc., New York, 1998.

[8] J. Edmonds, "Maximum matching and a polyhedron with 0,1-vertices," Journal of Research of the National Bureau of Standards (B) 69, pp. 125-130, 1965.

[9] L. Fleischer and É. Tardos, "Separating maximally violated comb inequalities in planar graphs," Mathematics of Opererations Research 24, pp. 130-148, 1999.

[10] M. Goemans and K.Talluri, *2-Change for k-Connected Networks*, Operations Research Letters 10, 1991, 113-117.

[11] M. Grötschel and M. Padberg, "On the symmetric travelling salesman problem I: Inequalities," Mathematical Programming 16, 1979, 265-280.

[12] M. Grötschel and M. Padberg, "On the symmetric travelling salesman problem II: lifting theorems and facets," Mathematical Programming 16, pp. 281-302, 1979.

[13] M. Grötschel and W. Pulleyblank, "Clique tree inequalities and the symmetric traveling salesman problem," Mathematics of Operations Research 11-4, pp. 537-569, 1986.

[14] D.S. Johnson and C.H. Papadimitriou, The Traveling Salesman Problem (E.L. Lawler et al, eds.), John Wiley & Sons, Chichester, pp. 37-85, 1985.

[15] M. Jünger, G. Reinelt and G. Rinaldi, Handbook on Operations Research and Management Sciences: Networks, (M. Ball et al., eds.), North-Holland, pp. 225-330, 1995.

[16] A.N. Letchford, "Separating a superclass of comb inequalities in planar graphs," Mathematics of Operations Research 25-3, pp. 443 - 454, 2001.

[17] D. Naddef, "Handles and teeth in the Symmetric Traveling Salesman Problem", DIMACS Series in Discrete mathematics and Theoretical Computer Science I, pp. 61-74, 1990.

[18] D. Naddef, "On the domino inequalities for the Symmetric Traveling Salesman Problem", Technical Report, Laboratoire ID-IMAG, 2001.

[19] D. Naddef and G. Rinaldi, "The graphical relaxation: A new framework for the Symmetric Traveling Salesman Problem," Mathematical Programming 58, pp. 53-88, 1992.

[20] D. Naddef and G. Rinaldi, "The crown inequalities for the Symmetric Traveling Salesman Polytope," Mathematics of Operations Research 17, No. 2, pp. 308-326, 1992.

[21] M.W. Padberg and M. Grötschel, "Polyhedral computations," in *The Traveling Salesman Problem* (Lawler et. al., eds.), John Wiley and Sons, Chichester, pp. 307-360, 1985.

[22] M.W. Padberg and G. Rao, "Odd minimum cut-sets and b-matchings," Mathematics of Operations Research 7, pp. 67-80, 1982.

[23] M.W. Padberg and G. Rinaldi, "Facet identification for the symmetric traveling salesman polytope," Mathematical Programming 47, pp. 219-257, 1990.

[24] M.W. Padberg and G. Rinaldi, "A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems," SIAM Reviews 33, pp. 60-100, 1991.

[25] G. Reinelt, "TSPLIB–A traveling salesman problem library," ORSA Journal of Computing 3, pp. 376-384, 1991.

[26] D. Vella, Using DP-constraints to Obtain Improved TSP Solutions, Master's Thesis, University of Ottawa, Ottawa, 2001.