INFORMS Journal on Computing
Vol. 19, No. 3, Summer 2007, pp. 356–365
ISSN 1091-9856 | EISSN 1526-5528 | 07 | 1903 | 0356

DOI 10.1287/ijoc.1060.0204

Computing with Domino-Parity Inequalities for the Traveling Salesman Problem (TSP)

William Cook
Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, Georgia 30332,
bico@isye.gatech.edu

Daniel G. Espinoza
Departamento de Ingeniería Industrial, Facultad de Ciencias Físicas y Matemáticas, Universidad de Chile,
Santiago 837-0439, Chile, daespino@dii.uchile.cl

Marcos Goycoolea
School of Business, Universidad Adolfo Ibáñez, Peñalolen, Santiago 794-1169, Chile,
marcos.goycoolea@uai.cl

We describe methods for implementing separation algorithms for domino-parity inequalities for the symmetric traveling salesman problem. These inequalities were introduced by Letchford (2000), who showed that the separation problem can be solved in polynomial time when the support graph of the LP solution is planar. In our study we deal with the problem of how to use this algorithm in the general (nonplanar) case, continuing the work of Boyd et al. (2001). Our 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 modify existing inequalities as the LP solution changes. We report computational results showing the strength of the new routines, including the optimal solution of a 33,810-city instance from the TSPLIB.

Key words: combinatorial optimization; traveling salesman problem; cutting-plane algorithm
History: Accepted by W. David Kelton, Editor-in-Chief; received October 2005; revised July 2006; accepted July 2006. Published online in *Articles in Advance* July 20, 2007.

1. Introduction

We consider the *traveling salesman problem* (TSP) with symmetric travel costs, that is, the cost to travel from city a to city b is the same as traveling from b to a. The input to the problem can be described as a complete graph $G = (V, E)$ with nodes V, edges E, and edge costs $(c_e: e \in E)$. Here V represents the cities and the problem is to find a *tour* of minimum total edge cost, where a tour is a cycle that visits each node exactly once (also known as a Hamiltonian cycle).

The TSP has long been an active platform for testing new ideas in integer programming and combinatorial optimization. Among the solution techniques proposed to date, the most successful has been the Dantzig et al. (1954) *cutting-plane method*. In their work, a tour is 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. Given any linear system $Ax \le b$ that is satisfied by every tour vector, the solution of the linear-programming (LP) problem

$$\text{minimize} \sum_{e \in E} c_e x_e \quad \text{subject to} \quad Ax \le b$$

provides a lower bound for the TSP. The cutting-plane method improves this bound by iteratively adding further linear inequalities, or *cutting planes*, that are satisfied by all tour vectors but not satisfied by the current LP solution vector $x^\star$; the task of finding such violated inequalities among a specified class is known as the *separation problem* for the class. Surveys of the wide body of work on the cutting-plane method for the TSP can be found in Jünger et al. (1995) and Naddef (2002).

An interesting approach to TSP separation was proposed by Letchford (2000), building on earlier work of Fleischer and Tardos (1999). Given a vector x^*, the support graph G^* is the subgraph of G having edge set $E^* = \{e \in E: x^*_e > 0\}$. Letchford introduced a class of TSP inequalities called *domino-parity constraints* and described a polynomial-time separation algorithm in the case where G^* is a planar graph. In this work Letchford exploits planarity by solving a key component of the separation problem in the planar dual graph. An initial study of his algorithm by Boyd et al. (2001), combining a computer implementation with by-hand computations, showed that the method

can produce effective cutting planes for instances with up to 1,000 nodes.

In this paper we present a further study of Letchford's algorithm, using the Concorde TSP code of Applegate et al. (2003) as the starting point for our work. We fully automate Letchford's method, handling the general (nonplanar) case by creating a nearby planar graph that can be used as a proxy for G^*. We demonstrate the strength of the new routines on a range of test instances obtained from the TSPLIB collection of Reinelt (1991), including the optimal solution of a 33,810-city example, the largest TSPLIB instance solved to date.

The paper is organized as follows. The domino-parity constraints are described in Section 2, together with a review of results from Letchford (2000) and a description of the steps adopted in our implementation to improve the practical efficiency of the separation algorithm. In Section 3 we describe shrinking techniques that allow us to handle large instances, and in Section 4 we describe heuristic methods to handle the common case where G^* is not planar. A local-search procedure for improving domino-parity (DP) constraints is described in Section 5 and computational results are presented in Section 6.

2. DP Inequalities and Letchford's Algorithm

For any $S \subseteq V$ let $\delta(S)$ denote the set of edges with exactly one end in S; a set of the form $\delta(S)$ for a proper subset $S \subseteq V$ is called a *cut*. 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 $x(F) = \sum(x_e\colon e \in F)$.

Every tour of G satisfies the *subtour-elimination constraints* $x(\delta(S)) \geq 2$, $\forall\, \varnothing \neq S \subsetneq V$. Using network-flow methods, the separation problem for these inequalities can be solved efficiently. That is, given a nonnegative vector x^*, a violated subtour-elimination constraint can be found in polynomial time, provided one exists. The subtour-elimination constraints were employed by Dantzig et al. (1954) and are a basic ingredient of modern implementations of the cutting-plane method. The solution set of the TSP relaxation

$$x(\delta(\{v\})) = 2 \quad \forall\, v \in V$$
$$x(\delta(S)) \geq 2 \quad \forall\, \varnothing \neq S \subsetneq V$$
$$0 \leq x_e \leq 1 \quad \forall\, e \in E$$

is known as the *subtour-elimination polytope* and is denoted by SEP(n), where $n = |V|$.

After the subtour-elimination constraints, the second most important class of cutting planes used in current TSP codes are the *comb inequalities* developed by Chvátal (1973) and Grötschel and Padberg (1979a). A comb is defined by subsets $H, T_1, \ldots, T_p$ of V such that p is odd, $T_1, \ldots, T_p$ are pairwise disjoint, and for each $i = 1, \ldots, p$ we have $H \cap T_i \neq \varnothing$ and $T_i \backslash H \neq \varnothing$. The set H is called the *handle* of the comb and the sets $T_1, \ldots, T_p$ are the *teeth*. Given any comb, the corresponding comb inequality $x(\delta(H)) + \sum(x(\delta(T_i))\colon i = 1, \ldots, p) \geq 3p + 1$ is satisfied by every tour vector. A nice theoretical property of comb inequalities is that, like subtour-elimination constraints, they induce facets of the convex hull of all tours (Grötschel and Padberg 1979b). Padberg and Rao (1982) provided a polynomial-time separation algorithm for the class of combs having $|T_i| = 2$ for all i; Fleischer et al. (2006) extended this result to the case when each tooth T_i satisfies either $|T_i \cap H| = 1$ or $|T_i \backslash H| = 1$. It is not known, however, if the separation problem for the full class of comb inequalities is polynomial-time solvable or whether it is NP-hard.

A direct generalization of combs was described by Letchford (2000). He defines a *domino* as a pair $\{A, B\}$ of nonempty subsets of V satisfying $A \cap B = \varnothing$ and $A \cup B \neq V$, extending the concept used in Applegate et al. (1995). Consider an odd number p of dominoes $\{A_1, B_1\}, \ldots, \{A_p, B_p\}$ together with an additional set $F \subset E$ such that for some $H \subseteq V$ the cut $\delta(H)$ is precisely the set of edges that appear an odd number of times among the sets $F, E(A_1 : B_1), \ldots, E(A_p : B_p)$. The *domino-parity* (*DP*) *inequality*

$$x(F) + \sum(x(E(A_i : B_i)) + x(\delta(A_i \cup B_i))\colon i = 1, \ldots, p) \geq 3p + 1 \tag{1}$$

is satisfied by all tour vectors, as shown by Letchford (2000). A comb is cast in this form using dominoes $\{T_i \cap H, T_i \backslash H\}$, $i = 1, \ldots, p$ and $F = \delta(H) \backslash \bigcup_{i=1,\ldots,p} E(T_i \cap H : T_i \backslash H)$. Note, however, that the ground sets of general dominoes in a DP inequality can intersect in arbitrary ways, leading to a much richer class than the comb inequalities. Despite this extra freedom, Naddef and Wild (2003) have shown that a broad subclass of DP inequalities also induce facets of the convex hull of tour vectors.

In working with DP inequalities we adopt notation introduced by Letchford (2000). Let $E_1, \ldots, E_k$ be a family of subsets of E and for each $e \in E$ define μ_e as the number of subsets E_i that contain e. The family of subsets is said to *support* the cut $\delta(K)$ if $\delta(K) = \{e \in E\colon \mu_e \text{ is odd}\}$. Now defining $(\mu_e\colon e \in E)$ for the family of subsets $F, E(A_1 : B_1), \ldots, E(A_p : B_p)$, the DP inequality (1) can be written as

$$\sum(\mu_e x_e\colon e \in E) + \sum(x(\delta(A_i \cup B_i))\colon i = 1, \ldots, p) \geq 3p + 1.$$

Considering $\mu = (\mu_e\colon e \in E)$ as a vector, the sum $\sum(\mu_e x_e\colon e \in E)$ can be written as an inner product μx, allowing us to write the DP inequality as

$$\mu x + \sum(x(\delta(A_i \cup B_i))\colon i = 1, \ldots, p) \geq 3p + 1.$$

The set H, called the *handle* of the DP inequality, is defined implicitly by μ, and it is unique (up to taking its complement in V). Note that a DP inequality may be characterized by its set of dominoes and either by its handle H, or by the set of extra edges F, or by the vector of multipliers μ. We will use these three descriptions interchangeably. Note also that H may be the empty set, and thus it may not define a cut in the usual sense.

Letchford (2000) proposed a two-stage algorithm that separates the class of DP constraints in polynomial time, provided the support graph G^* is planar and $x^\star \in \mathrm{SEP}(n)$. In the first stage, a set of candidate dominoes is constructed. In the second stage, a handle and an odd number of dominoes are selected so as to define a maximally violated constraint, provided one exists.

For the remainder of this section, assume that $x^\star \in \mathrm{SEP}(n)$. Also, 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 $\bar{G}^*$. The nodes and edges of $\bar{G}^*$ are denoted by $V(\bar{G}^*)$ and $E(\bar{G}^*)$ respectively.

2.1. Building Dominoes

The starting point of Letchford's algorithm is the observation that a domino can be constructed from a set of three $s-t$ paths in the dual graph $\bar{G}^*$. In the statement of this result we treat a path p as a set of edges in $\bar{G}^*$.

Lemma 1 (Letchford 2000). *Consider $s, t \in V(\bar{G}^*)$ and three edge-disjoint $s-t$ paths p_1, p_2, p_3 in $\bar{G}^*$. There exists a domino $\{A, B\}$ such that*

$$(\overline{\delta(A \cup B) \cap E(G^*)}) \cup (\overline{E(A:B) \cap E(G^*)}) = p_1 \cup p_2 \cup p_3.$$

Algorithm 1 describes the procedure to build a domino from the three paths. In Step 3 of the algorithm we have a choice among the sets S_1, S_2, and S_3 in forming the components of the domino; for storage purposes we choose A and B having minimum cardinality.

Algorithm 1. Primalizing Dual Dominoes (`prim_dom`(p_1, p_2, p_3))

Require: p_1, p_2, and p_3 are three edge-disjoint simple $s-t$ paths in $\bar{G}^*$.

1. Compute $\hat{p}_1$, $\hat{p}_2$, and $\hat{p}_3$, three noncrossing $s-t$ paths in $\bar{G}^*$ such that $\bigcup_{1\le i\le 3} p_i = \bigcup_{1\le i\le 3} \hat{p}_i$.
2. Compute S_1, S_2, and S_3, a partition of V with $\overline{\delta(S_1) \cap E(G^*)} = \hat{p}_1 \cup \hat{p}_2$, $\overline{\delta(S_2) \cap E(G^*)} = \hat{p}_2 \cup \hat{p}_3$, $\overline{\delta(S_3) \cap E(G^*)} = \hat{p}_3 \cup \hat{p}_1$.
3. Let A be the smallest $\{S_i\}_{1\le i\le 3}$ and let B be the second smallest $\{S_i\}_{1\le i\le 3}$.
4. **return** $\{A, B\}$

A key result of Letchford (2000) is that it suffices to use as candidate dominoes in DP inequalities only those that can be generated from Lemma 1. These dominoes can be obtained by computing for each pair of nodes (s, t), three edge-disjoint $s-t$ paths of minimum total weight, where the weight of an edge in $\bar{G}^*$ is its LP value x^*_e. To carry this out efficiently, several practical steps need to be adopted. The first observation is that by defining the weight of a domino $\{A, B\}$ as $w(\{A, B\}) = x(\delta(A \cup B)) + x(E(A:B)) - 3$ a DP inequality with domino-set $\{A_j, B_j\}_{1\le j\le p}$ can be written as

$$x(F) + \sum (w(\{A_j, B_j\}) : 1 \le j \le p) \ge 1.$$

Our assumption that $x^* \in \mathrm{SEP}(n)$ implies that for any domino $\{A, B\}$ we have $w(\{A, B\}) \ge 0$. Thus, if we are interested in violated inequalities we should consider only dominoes of weight less than one. We can therefore restrict our attention to paths satisfying $x^*(p_1 \cup p_2 \cup p_3) < 4$.

More can be done with the bounds on the weights of the paths p_1, p_2, and p_3 if we assume $x^* \in \mathrm{SEP}(n)$. First, note that no node at distance two or more from either s or t can be present in any of the three paths. Indeed, suppose such a node is contained in path p_3. We know that p_1, p_2 form a cycle in $\bar{G}^*$ and thus correspond to a cut in G^*. It follows that $x^*(p_1 \cup p_2) \ge 2$. Combining this with $x^*(p_3) \ge 2$ we violate the bound of 4. This allows us to run the domino-finding algorithm on a graph that is typically much smaller than the original. Moreover, using a successive shortest-path algorithm described in Ahuja et al. (1993) to compute p_1, p_2, and p_3, we obtain paths such that $x^*(p_1) \le x^*(p_2) \le x^*(p_3)$. Thus, sufficient conditions for the bound to be violated are $3x^*(p_1) \ge 4$ or $x^*(p_1) + 2x^*(p_2) \ge 4$, allowing us to prune further the search for dominoes.

To take advantage of these bounds we have implemented Dijkstra's algorithm with heaps so that whenever the latest labeled node has a value greater than a given bound, the algorithm is stopped. We call this function `dijkstra`$(G, s, w, \mathit{bound})$, where G is a graph, s is a node in $V(G)$, w is a weight vector on the edges of G, and *bound* is the stopping bound. This function returns a vector of distances from s to all nodes in G having distance less than *bound*, and infinite otherwise.

A detailed version of our domino-finding implementation is given in Algorithm 2. Note that the algorithm has a parameter α as input. To obtain all dominoes that might be used in a violated constraint it suffices to set $\alpha = 1$. In practice, however, we have seen that choosing $\alpha = 0.55$ greatly reduces the computation time for the routine, and it seems not to hurt the quality of the inequalities that are produced; this is the value used in the tests presented in Section 6.

ALGORITHM 2. Generating Candidate Dominoes (get_all_dominoes(G^*, α))

1. $\mathcal{L} \leftarrow \varnothing$.
2. $w(e) \leftarrow x^*_e, \forall e \in E(\bar{G}^*)$ {weight definition}
3. $c(e) \leftarrow 1, \forall e \in E(\bar{G}^*)$ {capacity of edges}
4. **for all** $s \in V(\bar{G}^*)$ **do**
5. $d_o \leftarrow$ dijkstra$(\bar{G}^*, s, w, 2)$
6. **for all** $t \in V(\bar{G}^*)$ and $t > s$ and $d_o(t) < (3+\alpha)/3$ **do**
7. $d \leftarrow d_o$
8. Send unit flow along the shortest $s-t$ path according to d.
9. Update the residual graph $\bar{G}^*_r$, residual costs w, and capacity c.
10. $val \leftarrow d(t)$
11. $bound \leftarrow \min((3+\alpha-val)/2, 2)$
12. $d \leftarrow$ dijkstra$(\bar{G}^*_r, s, w, bound)$
13. **if** $val + 2d(t) < 3+\alpha$ **then**
14. Send unit flow along the shortest $s-t$ path according to d.
15. Update the residual graph $\bar{G}^*_r$, residual costs w, and capacity c.
16. $val \leftarrow val + d(t)$
17. $bound \leftarrow \min(bound, 3+\alpha-val)$
18. $d \leftarrow$ dijkstra$(\bar{G}^*_r, s, w, bound)$.
19. **if** $val + d(t) < 3+\alpha$ **then**
20. Send unit flow along the shortest $s-t$ path according to d.
21. Compute the three unit-flow paths p_1, p_2, p_3 from s to t.
22. $D_{st} \leftarrow$ prim_dom(p_1, p_2, p_3), $w(D_{st}) \leftarrow val + d(t)$.
23. $\mathcal{L} \leftarrow \mathcal{L} \cup D_{st}$
24. **end if**
25. **end if**
26. **end for**
27. **end for**
28. **return** $\mathcal{L}$

The choice of α is based on computational tests showing a sharp divide in the performance for values above and below $\alpha = 0.50$. A summary of one set of experiments is given in Table 1, with α ranging from 0.1 to 1.0. In this test, our DP-separation routines were added to Concorde and the code was run with the option -mC20 to allow local cuts up to size 20 (see Applegate et al. 2003). The results in Table 1 are the average values over 20 randomly generated Euclidean instances, each having 2,500 cities. The column "subtour gap closed" reports the percentage of the gap between the cost of the optimal tour and the optimal value over SEP(n) that is closed by the run of Concorde with the DP separator; i.e., letting Z_{SEP} denote that optimum over SEP(n) and letting Z_{DP} denote the bound returned by Concorde + DP, the values are

$$100\left(1 - \frac{OPT - Z_{\text{DP}}}{OPT - Z_{\text{SEP}}}\right).$$

Table 1 DP Cuts on 2,500-City Random Euclidean Instances

α	Subtour gap closed (%)	CPU seconds	DP cuts found
0.10	94.753	673	0
0.20	94.753	676	0
0.30	94.753	689	0
0.40	94.760	687	2
0.45	94.854	747	29
0.50	97.571	3,161	2,620
0.55	97.515	3,157	2,608
0.60	97.585	3,259	2,618
0.70	97.603	3,551	2,677
0.80	97.602	3,579	2,628
0.90	97.550	3,707	2,612
1.00	97.588	3,839	2,598

The "CPU seconds" column gives the total CPU time on an 2.66 GHz Intel Xeon workstation; the "DP cuts found" column reports the total number of DP cuts that were added to the LP relaxation.

In the tests reported in Table 1, the reduction in running time for $\alpha = 0.55$ versus $\alpha = 1.0$ is 18%, without a significant difference in the bound produced by Concorde + DP. Moreover, this running-time reduction grows with the size of the instance, resulting in substantial savings for large test problems.

Another possibility to speed up the domino-generation step is to do Steps 4–27 of Algorithm 2 in parallel. The computations to obtain all dominoes originating at a node s are independent of the computations needed to obtain dominoes from any other node t. This easy parallelization allowed us to use a cluster of 50–100 machines to generate all dominoes, greatly reducing the (actual) time needed for testing the code.

2.2. The Odd-Cycle Problem

The next stage in Letchford's algorithm is to build an inequality from the collection of dominoes and the edges in the dual graph. For this, define an auxiliary multigraph M^* with node set $V(M^*) = V(\bar{G}^*)$. For each edge $e = \{u, v\} \in E(\bar{G}^*)$ define an *even* edge $e = \{u, v\} \in E(M^*)$ with weight $w_e = x^*_e$, and for each domino $D_{uv} \in \mathcal{L}$ define an *odd* edge $e = \{u, v\} \in E(M^*)$ with weight $w_e = w(D_{uv})$. An *odd cycle* in M^* is a cycle with an odd number of odd edges.

THEOREM 2 (LETCHFORD 2000). *There exists a violated DP inequality in G^* if and only if there exists an odd cycle in M^* with weight less than one. Furthermore, if such a cycle exists, a minimum-weight odd cycle in M^* corresponds to a maximally violated DP inequality, where F is defined by the even edges in the cycle, and $\mathcal{T}$ by the odd edges in the cycle.*

In fact, given any odd cycle $C \subseteq E(M^*)$ with weight $w(C) < 1$, it is possible to construct a DP inequality with violation $1 - w(C)$, by defining the set F as the even edges in C and choosing the set of dominoes $\mathcal{T}$ to be those corresponding to odd edges in C.

The process of building the inequality is summarized in Algorithm 3. Note that we can omit from M^* all edges $e \in E(\bar{G}^*)$ such that $x^*_e \geq 1 - \varepsilon$, where ε is the minimum violation that we would like to obtain.

ALGORITHM 3. DP-Inequality Separation
1. max_*violation* $\leftarrow 0$
2. $\mathcal{L} \leftarrow$ `get_all_dominoes`$(G^*, 1)$.
3. build graph M^*
4. **for all** $v \in V(M^*)$ **do**
5. Compute a minimum odd cycle C passing through v
6. **if** $1 - w(C) >$ max_*violation* **then**
7. max_*violation* $\leftarrow 1 - w(C)$
8. $F \leftarrow$ even edges in C
9. $\mathcal{T} \leftarrow \{D_{uv} \in \mathcal{L}$: e_{uv} odd, $e_{uv} \in C\}$
10. **end if**
11. **end for**
12. **return** F, $\mathcal{T}$

Our statement of the algorithm uses the standard method for obtaining a minimum odd cycle in a graph (Barahona and Mahjoub 1986, Grötschel et al. 1993): For each node in M^* we compute a minimum odd cycle containing that node. Adopting this approach, Boyd et al. (2001) proposed to keep all violated inequalities that arise during the algorithm, rather than just the single constraint of maximal violation. This addresses the practical concern that a selection of cutting planes is usually superior to a single violated inequality. In our implementation we found it useful to extend this idea by adopting a heuristic search procedure to attempt to find additional inequalities. The technique we use is to sample the odd cycles by performing random walks starting at each node. At each step of the walk we select an edge to extend the current path, such that we create neither an even cycle nor an odd cycle with a single odd edge. (Odd cycles with one odd edge generate DP inequalities with one domino, which are not violated if $x^* \in$ SEP(n).) The edges are selected with probability proportional to their x^*-weight. We restart the walk if the resulting path has total weight greater than $1 - \varepsilon$ or if we find an odd cycle, in which case we record the corresponding constraint. In our tests we spend 10–30 seconds in this sampling process, evenly distributed among all nodes in M^*.

On large examples the random-walk procedure is able to find several hundreds of thousands of different inequalities, leading us to the problem of selecting which inequalities to report. Clearly we cannot return them all, and keeping the set of most violated ones leads to storing multiple inequalities that are almost identical. Following the ideas studied by Andreello et al. (2007), we choose a strategy that balances keeping highly violated inequalities and inequalities that cover different parts of the graph.

3. Safe Shrinking

In applying the DP-separation algorithm it is crucial to preprocess G^* to reduce the size of the graph that must be handled. Given a graph G and two nodes $u, v \in V(G)$, let $G/\{u, v\}$ denote the graph obtained by contracting the nodes u, v into a single node y and eliminating any resulting self-loop edges. This operation is called *shrinking* u, v in G. The following result provides conditions that allow us to shrink pairs of nodes (u, v), guaranteeing that violated DP constraints will be available in the shrunken graph $G^*/\{u, v\}$ if violated DP constraints were present in G^*.

THEOREM 3. *Let $x^* \in$ SEP(n) and let $u, v, t \in V(G^*)$ be such that $x^*_{uv} = 1$ and $x^*_{ut} + x^*_{tv} = 1$. If there exists a violated DP inequality in G^*, then there exists a DP inequality in $G^*/\{u, v\}$ with violation no less than the violation of the original inequality.*

Padberg and Rinaldi (1991) proved that, under these *safe-shrinking* conditions, if there exists a violated inequality for the TSP (i.e., x^* is not in the convex hull of tours), then the shrunken graph will also have a violated inequality. Their result, however, does not imply that, if there is a cutting plane from a particular class (like DP inequalities), then there remains a cutting plane from that class.

This section is devoted to a proof of Theorem 3. In our computational tests, repeated use of this result was able to reduce the size of the support graphs we considered by 60% on average.

3.1. Domino Transformations

We begin by describing transformations that take a DP inequality described by F, $\mathcal{T}$ and return another DP inequality F', $\mathcal{T}'$ with violation greater than or equal to the violation of the original inequality. We will sometimes interpret $D \subseteq E(G^*)$ as a set of edges and at other times as an $|E(G^*)|$-vector with 1 for each edge in D and 0 for all other edges. For a domino $T = \{A_T, B_T\} \in \mathcal{T}$ we use the short-hand $\delta(T)$ to mean $\delta(A_T \cup B_T)$ and we denote by C_T the set $V \backslash \{A_T \cup B_T\}$.

3.1.1. Duplicate-Domino Elimination. Let $T_1, T_2 \in \mathcal{T}$ be such that $T_1 = T_2$. Define $\mathcal{T}' = \mathcal{T} \backslash \{T_1, T_2\}$, and $F' = F$. Note that since $\mu - \mu' = 2E(A_{T_1} : B_{T_1})$, the vectors μ and μ' support the same cut (since μ_e and μ'_e have the same parity for each edge e). Furthermore,

the violation of the new inequality is at least the violation of the original. Indeed

$$\begin{aligned}
&\mu x + \sum_{T\in\mathcal{T}} x(\delta(T)) - 3|\mathcal{T}| - 1 \\
&\quad = \mu' x + \sum_{T\in\mathcal{T}'} x(\delta(T)) - 3|\mathcal{T}'| - 1 + \underbrace{w(T_1)}_{\geq 0} + \underbrace{w(T_2)}_{\geq 0} \\
&\quad \geq \mu' x + \textstyle\sum(x(\delta(T))\colon T\in\mathcal{T}') - 3|\mathcal{T}'| - 1.
\end{aligned}$$

We can therefore delete from $\mathcal{T}$ any pair of duplicate dominoes without decreasing the violation of the cutting plane.

3.1.2. Domino Reduction. Let $T_o \in \mathcal{T}$ and let $\varnothing \neq A' \subseteq A_{T_o}$, $\varnothing \neq B' \subseteq B_{T_o}$ be subsets of the components of the domino. Define $T_o' = \{A', B'\}$, and suppose that $x(\delta(T_o')) \leq x(\delta(T_o))$. To simplify notation, call $S = E(A_{T_o} : B_{T_o})$ and $S' = E(A' : B')$. Define a new inequality with $\mathcal{T}' = (\mathcal{T}\setminus\{T_o\}) \cup \{T_o'\}$ and $F' = F\Delta(S\setminus S')$, where, as usual, $P\Delta Q$ denotes the symmetric difference of sets P and Q. We have

$$\begin{aligned}
\mu - \mu' &= F - F\Delta(S\setminus S') + S - S' \\
&= F - F\Delta(S\setminus S') + (S\setminus S') \quad (\text{note } S'\subseteq S) \\
&= 2F \cap (S\setminus S').
\end{aligned}$$

Thus μ and μ' support the same cut. Finally, note that the left-hand side of the new inequality is less than or equal to that of the original constraint:

$$\begin{aligned}
&\mu x + \sum_{T\in\mathcal{T}} x(\delta(T)) \\
&\quad = \mu' x + \underbrace{2x(F\cap(S\setminus S'))}_{\geq 0} + \sum_{T\in\mathcal{T}\setminus\{T_o\}} x(\delta(T)) + \underbrace{x(\delta(T_o))}_{\geq x(\delta(T_o'))} \\
&\quad \geq \mu' x + \textstyle\sum(x(\delta(T))\colon T\in\mathcal{T}').
\end{aligned}$$

We thus obtain a new DP inequality $(F', \mathcal{T}')$ with violation at least that of the original and with T_o replaced by T_o'.

3.1.3. Domino Rotation. Boyd et al. (2001) have shown that in a DP inequality a domino $\{A_T, B_T\} \in \mathcal{T}$ can be *rotated* to $\{A_T, C_T\}$ without altering the inequality. To see this let $T_o \in \mathcal{T}$ and define $A'_{T_o} = A_{T_o}$, $B'_{T_o} = C_{T_o}$, $T_o' = \{A'_{T_o}, B'_{T_o}\}$, $\mathcal{T}' = (\mathcal{T}\setminus\{T_o\}) \cup \{T_o'\}$, and $F' = F$. To simplify the notation, call $S_1 = E(A_{T_o} : B_{T_o})$, $S_2 = E(C_{T_o} : A_{T_o})$, and $S_3 = E(C_{T_o} : B_{T_o})$. Then $\mu' - \mu = -S_1 + S_2$ and

$$\mathrm{Odd}(\mu') = \mathrm{Odd}(\mu)\Delta\delta(A_{T_o}) = \delta(H)\Delta\delta(A_{T_o}) = \delta(H\Delta A_{T_o}),$$

where $\mathrm{Odd}(\mu)$ denotes the edges having odd value μ_e. Thus, $H\Delta A_{T_o}$ is the handle of the new DP inequality. Note also that the left-hand side of the inequality does not change:

$$\begin{aligned}
&\mu x + \sum_{T\in\mathcal{T}} x(\delta(T)) \\
&\quad = (\mu - S_1)x + x(S_1) + \sum_{T\in\mathcal{T}\setminus\{T_o\}} x(\delta(T)) + \underbrace{x(S_2) + x(S_3)}_{x(\delta(T_o))} \\
&\quad = \underbrace{(\mu - S_1 + S_2)x}_{\mu' x} + \sum_{T\in\mathcal{T}'\setminus\{T_o'\}} x(\delta(T)) + \underbrace{x(S_1) + x(S_3)}_{x(\delta(T_o'))} \\
&\quad = \mu' x + \textstyle\sum(x(\delta(T))\colon T\in\mathcal{T}').
\end{aligned}$$

We thus obtain a new DP inequality $(F', \mathcal{T}')$ with violation equal to the violation of the original constraint, with T_o replaced by T_o' and H replaced by $H\Delta A_{T_o}$.

3.2. Proof of Safe-Shrinking Conditions

Let λ be the maximum violation for x^* over all DP inequalities and let u, v, t satisfy the conditions in Theorem 3, namely $x^*_{uv} = 1$ and $x^*_{ut} + x^*_{vt} = 1$. Now let $(F, \mathcal{T})$ be a DP inequality of violation λ that minimizes the coefficient of the edge uv. We prove Theorem 3 by showing that the coefficient of uv is zero, and thus we can shrink (u, v) into a single node and keep the same violation.

To begin, note that the coefficient of uv in the DP inequality can be written as

$$\begin{aligned}
\mathrm{coeff}(uv) = {} & |\{T\in\mathcal{T}\colon uv\in\delta(T)\}| + |\{T\colon uv\in E(A_T : B_T)\}| \\
& + |\{uv\}\cap F|.
\end{aligned}$$

Suppose $\mathrm{coeff}(uv) > 0$.

Claim 1. *We may assume that for all $T\in\mathcal{T}$ we have $uv\notin\delta(T)$.*

Proof. Suppose $T_o\in\mathcal{T}$ is such that $uv\in\delta(T_o)$. We may assume that $u\in A_{T_o}$ and $v\in C_{T_o}$. By rotating T_o we obtain an equivalent constraint $(F', \mathcal{T}')$ with $|\{T\in\mathcal{T}\colon uv\in\delta(T)\}| > |\{T\in\mathcal{T}'\colon uv\in\delta(T)\}|$. □

Claim 2. *If $uv\in E(A_T : B_T)$ for $T\in\mathcal{T}$, then we may assume $A_T = \{u\}$ and $B_T = \{v\}$.*

Proof. Suppose $T_o\in\mathcal{T}$ has $uv\in E(A_{T_o} : B_{T_o})$. We may assume $u\in A_{T_o}$ and $v\in B_{T_o}$. Note that $x^*_{uv} = 1$ implies that $x^*(\delta(\{u, v\})) = 2$, and $x^*\in\mathrm{SEP}(n)$ implies that $x^*(\delta(A_{T_o}\cup B_{T_o})) \geq 2$. We can apply Domino Reduction with $A'_{T_o} = \{u\}$ and $B'_{T_o} = \{v\}$. □

Combining Claim 2 with Duplicate-Domino Elimination, we may assume that there is at most one domino in $\mathcal{T}$ having the form $T_{\mathrm{I}} = \{\{u\}, \{v\}\}$.

Claim 3. *We have* $\mathrm{coeff}(uv) \leq 1$.

Proof. If $\mathrm{coeff}(uv) = 2$ then $T_{\mathrm{I}}\in\mathcal{T}$ and $uv\in F$. In this case we define $\mathcal{T}' = \{T_{\mathrm{II}}\}\cup\mathcal{T}\setminus\{T_{\mathrm{I}}\}$, where $T_{\mathrm{II}} = \{\{u, v\}, \{t\}\}$, and $F' = F\Delta\{uv, ut, vt\}$. We show that $(F', \mathcal{T}')$ defines a DP inequality with $\mathrm{coeff}(uv) = 0$,

and with violation at least that of $(F, \mathcal{T})$. To simplify notation, call $S_1 = \{uv, vt, ut\}$, $S_2 = E(A_{T_I} : B_{T_I}) = \{uv\}$, and $S_3 = E(A_{T_{II}} : B_{T_{II}}) = \{vt, ut\}$. First note that μ' supports the same cut as μ: $\mu - \mu' = F - F\Delta S_1 + S_2 - S_3 = F - F\Delta S_1 + S_1 - 2S_3 = 2(F \cap S_1 - S_3)$. Now using the fact that $x^*_{ut} + x^*_{tv} = 1$, we have $x(\delta(T_I)) = x(\delta(T_{II})) = 2$. It follows that the left-hand side of the new inequality is less than or equal to the left-hand side of the original constraint:

$$\begin{aligned}
&\mu x + \sum_{T\in\mathcal{T}} x(\delta(T)) \\
&\quad = \mu' x + \underbrace{2x(S_1 \cap F)}_{\geq 2} - \underbrace{2x(S_3)}_{=2} + \sum_{T\in\mathcal{T}\setminus\{T_I\}} x(\delta(T)) + \underbrace{x(\delta(T_I))}_{=x(\delta(T_{II}))} \\
&\quad \geq \mu' x + \textstyle\sum(x(\delta(T)) \colon T \in \mathcal{T}').
\end{aligned}$$

Thus we have obtained a DP inequality with $\text{coeff}(uv) = 0$ and with violation at least as great as that of $(F, \mathcal{T})$, a contradiction. □

Claim 4. *Edge uv is not in F.*

Proof. If $uv \in F$, then $T_I \notin \mathcal{T}$ and we may assume that $u \in H$ and $v \in V\setminus H$. We will define a new inequality $(F', \mathcal{T}')$ so that $\text{coeff}(uv) = 0$. In the new inequality we let $\mathcal{T}' = \mathcal{T}$ and $F' = F\Delta\delta(\{v\})$.

Note that μ' supports the cut $H \cup \{v\}$: $\text{Odd}(\mu') = \text{Odd}(\mu)\Delta(\delta(\{v\})) = \delta(H)\Delta\delta(\{v\}) = \delta(H\Delta\{v\}) = \delta(H \cup \{v\})$. Also note that the left-hand side of the new inequality is less than or equal to the left-hand side of the original inequality:

$$\begin{aligned}
&\mu x + \sum_{T\in\mathcal{T}} x(\delta(T)) - \mu' x - \sum_{T\in\mathcal{T}'} x(\delta(T)) \\
&\quad = x(F) - x(F\Delta\delta(\{v\})) \\
&\quad = \underbrace{x(F \cap \delta(v))}_{\geq x(uv)} - \underbrace{x((F\Delta\delta(v)) \cap \delta(v))}_{\leq 2 - x(uv)} \\
&\quad \geq 2x(uv) - 2 = 0.
\end{aligned}$$

Thus the new inequality $(F', \mathcal{T}')$ has violation at least as great as that of the original constraint and $\text{coeff}(uv) = 0$, a contradiction. □

It follows that $T_I \notin \mathcal{T}$, since otherwise $\text{coeff}(uv) = 0$. We assume that $u, t \in H$ and $v \in V\setminus H$ (the alternative case, when v and t are on the same side of the handle, is analogous to this one). It follows that that μ_{vt} is odd.

Claim 5. *We may assume $vt \notin F$.*

Proof. If $vt \in F$, then we can replace T_I by a new domino T_{II} and add v to the handle as follows. Let y represent the set $V\setminus\{u, v, t\}$ and consider the aggregated graph given in Figure 1. In the new inequality we let $F' = F\Delta\{vt, uy\}$, $T_{II} = \{\{y\}, \{u, v\}\}$, and $\mathcal{T}' = (\mathcal{T}\setminus\{T_I\}) \cup \{T_{II}\}$.

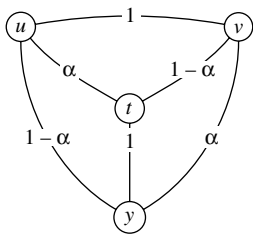

Figure 1 Aggregated Graph with x^*-Values on Edges

Clearly $w(T_I) = w(T_{II})$. Moreover, the left-hand side of the new DP inequality $(F', \mathcal{T}')$ is less than or equal to the left-hand side of the original DP inequality $(F, \mathcal{T})$:

$$\begin{aligned}
LHS - LHS' &= x(F) - x(F\Delta\{vt, uy\}) + w(T_I) - w(T_{II}) \\
&= \underbrace{x(F \cap \{vt, uy\})}_{\geq x(vt)} - \underbrace{x(\{vt, uy\}\setminus F)}_{\leq x(uy)} \\
&\geq (1-\alpha) - (1-\alpha) = 0.
\end{aligned}$$

We finally show that the cut supported by μ' is $\delta(H \cup \{v\})$. First observe that $\mu - \mu' = F - F\Delta\{vt, uy\} + E(A_{T_I} : B_{T_I}) - E(A_{T_{II}} : B_{T_{II}}) = \{vt, uy\} \cap F - \{vt, uy\}\setminus F + \{uv\} - \{uy, vy\}$. Now if $uy \in F$, we have $\mu - \mu' = \{vt, uv\} - \{vy\}$, and if $uy \notin F$ we have $\mu - \mu' = \{vt, uv\} - \{vy\} - 2\{uy\}$. In either case, $\text{Odd}(\mu - \mu') = \delta(\{v\})$, which in turn implies that $\text{Odd}(\mu') = \delta(H \cup \{v\})$. Thus we have a new DP inequality with $vt \notin F'$. □

With $vt \notin F$ and with μ_{vt} odd, there must exist a domino T_{II} such that $v \in A_{T_{II}}$, $t \in B_{T_{II}}$. Moreover, since $\text{coeff}(uv) = 1$ and $T_I \in \mathcal{T}$, we have $u \in A_{T_{II}}$. Now, by Domino Reduction we may assume that $A_{T_{II}} = \{u, v\}$ and $B_{T_{II}} = \{t\}$. Note that this (plus parity, since $ut \notin \delta(H)$) implies that $\mu_{ut} \geq 2$.

Claim 6. *We may assume $ut \notin F$.*

Proof. If $ut \in F$ then we can eliminate T_I, T_{II} from $\mathcal{T}$ and redefine H as $H \cup \{v\}$ as follows. Define $\mathcal{T}' = \mathcal{T}\setminus\{T_I, T_{II}\}$ and $F' = F\Delta\{ut, yv\}$. Using arguments similar to those above, it can be checked that the cut supported by μ' is $H \cup \{v\}$ and that the new violation is at least as great as that of the original inequality. □

It follows that there exists a domino T_{III} such that $u \in A_{T_{III}}$ and $t \in B_{T_{III}}$. Using the same arguments as before, we can transform T_{III} into T_{II} and then we have $\mu_{ut} = |\{T_{II} \in \mathcal{T}\}| = \mu_{tv}$. But this contradicts the fact that μ_{vt} is odd and μ_{ut} is even, thus completing the proof of Theorem 3.

4. Finding a Planar Graph

Large TSP instances rarely produce LP solutions with planar support. To succeed in practice it is therefore necessary to modify G^* and x^* to obtain a planar graph that can be used as a substitute in the

DP-separation algorithm. Such a process may lose violated inequalities, but if the new planar graph is close (under some measure) to G^* then we can still produce a good selection of cutting planes.

Nonplanar graphs were encountered in test instances studied by Boyd et al. (2001). Their approach was to perform general (possibly unsafe) shrinking steps by hand, using a visual inspection of a drawing of G^* to guide the process to produce a planar graph. We adopt such an approach, using planarity-testing algorithms to automate the process as suggested in Vella (2001).

Efficient algorithms are available that return either a planar embedding of a graph or a $K_{3,3}$ or K_5 minor. To use such an algorithm to obtain a planar graph, whenever a minor K is returned we select nodes u, v having degree at least three in K, replace G^* by $G^*/\{u, v\}$, and repeat. With this approach, if $x^* \in \mathrm{SEP}(n)$ then the new shrunken fractional solution also satisfies the subtour-elimination constraints. A drawback is that the resulting fractional solution does not necessarily satisfy the *degree constraints* $x(\delta(\{v\})) = 2$ for all $v \in V$. This detail is not crucial, however, since the DP inequalities and the separation algorithm are valid also for the *graphical traveling salesman problem*, where nodes and edges can be used more than once in the tour. A discussion of this point is given in Letchford (2000).

4.1. Edge-Elimination Planarization

An alternative way to obtain planar graphs is repeatedly to delete edges found in a $K_{3,3}$ or K_5 minor. An effective way to choose an edge to delete is to use binary search to identify the minimum-weight edge such that the subgraph containing all edges of greater weight is planar. Thus the minimum weight to eliminate from the graph to make it planar is at least the x^*_e-value of the selected edge. Since the complexity of planarity testing is $\mathcal{O}(|V(G^*)|)$, the total complexity to select the edge is $\mathcal{O}(\log(|E(G^*)|)|E(G^*)|)$.

We have found that the total weight of eliminated edges is usually quite small. A problem with the method, however, is that it produces a vector x^* that does not satisfy the degree constraints or the subtour-elimination constraints. This implies that the weight of the dominoes found during the domino-generation step may be negative, which may create negative cycles in M^*. To avoid this issue we simply set the weight of all negative dominoes to zero. Now, before returning the cuts found during the second phase of the algorithm, we re-compute exactly the actual violation for the inequality in the original graph.

Note that many more schemes are possible to generate planar graphs from a graph, e.g., mixing the shrinking and edge-deletion steps. Neither of the heuristic methods we presented dominates the other and in our computational tests we separate the DP inequalities in the graphs obtained from both methods. The problem of obtaining a planar graph that represents well an LP solution deserves further study. For a general discussion of graph planarity and the TSP we refer the reader to Letchford and Pearson (2005).

5. Tightening DP Inequalities

Applegate et al. (2003) proposed a cutting-plane *tightening* procedure for the TSP that modifies existing inequalities in response to changes in the LP solution vector. This is an effective way of dealing with successive vectors x^* that differ only slightly. In our case, where we run the DP-separation algorithm on an approximation to G^*, a tightening process can help correct for any flaws we introduce in our planarization procedure.

The Applegate et al. (2003) process works by making a series of greedy steps to adjust the sets that define a TSP cutting plane. To adopt their approach in our case, note that a DP inequality is completely defined by a family of dominoes and a handle. The modification steps we consider are to add or remove nodes from dominoes and the handle, to move a node from one side of a domino to the other, and simultaneously to change sides in a domino and move in or out of the handle. The basic steps are organized into an algorithm following the general strategy of Applegate et al. (2003).

A node u is *relevant* in the tightening heuristic if there exists $e \in \delta(u)$ such that it has a nonzero coefficient in the DP inequality. We begin by computing the *best move*, that is, among all feasible moves in all relevant nodes we find the one giving the greatest improvement in the violation of the constraint. While this violation improvement is sufficiently positive, we perform the move and update the new best move. If the best move gives improvement less than ε, we attempt to perform a sequence of moves to enlarge either the handle or a domino, followed by moves that flip elements within a domino, and finally moves that shrink a domino or a handle. We continue this process until some ε-improving move is found (and then go back to our greedy approach), or until we cannot make any further moves. The algorithm will never cycle, since each move can be performed only once within each ε-improvement phase. In our tests ε was chosen as 10^{-6}.

6. Computational Results

We have implemented the DP-separator routines in the C-programming language. The routines are incorporated into the Concorde TSP code of Applegate et al. (2003), searching for DP inequalities in each

Table 2 DP Cuts on TSPLIB Instances (% of Subtour Gap Closed)

Name	Concorde (%)	DP (%)	Gap Δ (%)	Concorde hours	DP hours
pcb3038	96.834	99.185	73.1	41.3	9.2
fl3795	74.149	98.299	92.8	79.6	121.0
fnl4461	98.594	99.297	50.0	7.5	9.8
rl5915	98.237	99.424	66.7	98.8	28.1
rl5934	98.389	99.339	59.1	21.8	21.5

full pass through Concorde's native cutting-plane routines. The computations were performed on a 2.66 GHz Intel Xeon workstation using ILOG CPLEX 6.5 as an LP solver. The planarity-testing algorithm is due to Boyer and Myrvold (2004) as implemented by J. Boyer.

Our tests use the option -mC48 to allow Concorde to call local cuts repeatedly up to size 48; this setting requires additional CPU time, but it allows Concorde to obtain substantially better lower bounds. Local cuts of size 48 are the largest value that can be effectively handled in Concorde. Our test bed consists of all instances in the TSPLIB collection of Reinelt (1991) having at least 3,000 cities.

In Table 2 we report mean values for a set of ten trials for each mid-sized TSPLIB instance. The column "Concorde" gives the percentage of the subtour gap closed when running Concorde without DP inequalities. The column "DP" gives the same result when we include the DP-separator routines. The "Gap Δ" column gives the percentage of the average remaining gap (between the Concorde bound and the optimal tour value) that is closed when DP inequalities are added. The final two columns report the running times in hours. The improvements in the gaps are considerable, given the already strong bounds obtained by the -mC48 runs of Concorde.

Results for all TSPLIB problems having at least 6,000 cities are reported in Table 3. Each of these results is for a single trial. In these tests we first ran Concorde without the DP-separator routines, and then used the final LP as the starting point for running Concorde together with the new routines. The "Optimal" column reports the value of the optimal tour; in the case of pla85900 this is the value of the best known tour, found by Helsgaun (2000). The "Concorde" and "Conc + DP" columns report the LP bounds obtained by the two runs of Concorde. The "Gap Δ" column gives the percentage gap closed in the second run. Again, the additional gap closed is substantial, although the improvement is decreasing as the problem size increases.

Table 4 LP Bounds for d18512 and pla33810

Name	Optimal	Concorde (with pool)	Concorde + DP (with pool)	Gap Δ (%)
d18512	645,238	645,202	645,209	19.4
pla33810	66,048,945	66,018,619	66,037,858	63.4

6.1. Solution of d18512 and pla33810

The large improvements in the LP bounds indicates that the DP-separator routines could aid in the solution of difficult instances of the TSP. As case studies, we focused our attention on d18512 and pla33810, two of the remaining three unsolved problems in the TSPLIB. The first of these problems is a collection of cities in Germany and the second arose in a VLSI application at AT&T. For each problem we began with the best available LP relaxation, found with Concorde by gathering cuts into a pool during a sequence of three branch-and-cut runs (stopping each run after it reached 1,000 active subproblems). Using the DP-separator routines these were improved to the values reported in Table 4, using two additional branch-and-cut runs in the case of pla33810.

A branch-and-cut run on d18512, starting with the 645,209 LP and an upper bound of 645,239, required 424,241 subproblems to establish the optimality of the 645,238-value tour found by Tamaki (2003). The total running time was approximately 57.5 CPU years, carried out on a network of Xeon compute nodes.

For pla33810 we established the optimal value of 66,048,945, a slight improvement on the best reported heuristic tour of value 66,050,499, found by Helsgaun (2000) with a variant of his LKH code. The branch-and-cut run that solved the instance used 577 subproblems (given the upper bound of one greater than Helsgaun's tour). We also solved the instance a second time starting with a 66,037,858 LP (obtained using

Table 3 DP Cuts on Largest TSPLIB Instances

Name	Optimal	Concorde	Conc + DP	Gap Δ (%)	Concorde hours	DP hours
pla7397	23,260,728	23,255,280	23,258,947	67.3	70.5	268.2
rl11849	923,288	923,053	923,209	66.4	118.0	104.4
usa13509	19,982,859	19,979,209	19,981,200	54.5	81.2	109.9
brd14051	469,385	469,321	469,354	51.6	53.2	159.5
d15112	1,573,084	1,572,863	1,572,967	47.1	124.0	152.7
d18512	645,238	645,166	645,195	40.3	73.9	186.5
pla33810	66,048,945	65,972,887	66,001,234	37.3	19.0	231.0
pla85900	(142,382,641)	142,265,646	142,296,660	26.5	224.2	174.1

the cuts from the earlier run) and an upper bound of one greater than the optimal value; the branch-and-cut run in this case used 135 subproblems. The total CPU time was approximately 15.7 CPU years (the additional branch-and-cut run of 135 nodes took 86.6 days).

Our solutions of d18512 and pla33810 should be viewed only as evidence of the potential strength of the new procedures; the computational studies were made as we were developing our code and the runs were subject to arbitrary decisions to terminate tests as the code improved. The 33,810-city TSP is currently the largest test instance that has been solved, improving on the 24,978-city instance solved with Concorde. The relatively small search tree for pla33810 may be due in part to the natural structure in the VLSI-derived data set that is not present in d18512.

Acknowledgments

The authors were supported by Office of Naval Research Grant N00014-03-1-0040 and by National Science Foundation Grant DMI-0245609. The authors thank John Boyer for allowing them to use his excellent implementation of the Boyer-Myrvold planarity testing algorithm. The authors also thank the referees for suggestions that improved the presentation of the results.

References

Ahuja, R. K., T. L. Magnanti, J. B. Orlin. 1993. *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall, Englewood Cliffs, NJ.

Andreello, G., A. Caprara, M. Fischetti. 2007. Embedding $\{0, \frac{1}{2}\}$-cuts in a branch-and-cut framework: A computational study. *INFORMS J. Comput.* **19** 229–238.

Applegate, D., R. Bixby, V. Chvátal, W. Cook. 1995. Finding cuts in the TSP (A preliminary report). DIMACS Report 95-05, Rutgers University, New Brunswick, NJ.

Applegate, D., R. Bixby, V. Chvátal, W. Cook. 2003. Implementing the Dantzig-Fulkerson-Johnson algorithm for large traveling salesman problems. *Math. Programming* **97** 91–153.

Barahona, F., A. R. Mahjoub. 1986. On the cut polytope. *Math. Programming* **36** 157–173.

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, Ottawa, Canada.

Boyer, J. M., W. Myrvold. 2004. On the cutting edge: Simplified $O(n)$ planarity by edge addition. *J. Graph Algorithms Appl.* **8** 241–273.

Chvátal, V. 1973. Edmonds polytopes and weakly Hamiltonian graphs. *Math. Programming* **5** 29–40.

Dantzig, G., R. Fulkerson, S. Johnson. 1954. Solution of a large-scale traveling salesman problem. *Oper. Res.* **2** 393–410.

Fleischer, L., É. Tardos. 1999. Separating maximally violated comb inequalities in planar graphs. *Math. Oper. Res.* **24** 130–148.

Fleischer, L. K., A. N. Letchford, A. Lodi. 2006. Polynomial-time separation of a superclass of simple comb inequalities. *Math. Oper. Res.* **31** 696–713.

Grötschel, M., M. W. Padberg. 1979a. On the symmetric traveling salesman problem I: Inequalities. *Math. Programming* **16** 265–280.

Grötschel, M., M. W. Padberg. 1979b. On the symmetric traveling salesman problem II: Lifting theorems and facets. *Math. Programming* **16** 281–302.

Grötschel, M., L. Lovász, A. Schrijver. 1993. *Geometric Algorithms and Combinatorial Optimization*, 2nd ed. Springer, Berlin, Germany.

Helsgaun, K. 2000. An effective implementation of the Lin-Kernighan traveling salesman heuristic. *Eur. J. Oper. Res.* **126** 106–130.

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.

Letchford, A. N. 2000. Separating a superclass of comb inequalities in planar graphs. *Math. Oper. Res.* **25** 443–454.

Letchford, A. N., N. Pearson. 2005. Exploiting planarity in separation routines for the symmetric traveling salesman problem. Preprint, Department of Management Science, Lancaster University, Lancaster, UK.

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, The Netherlands, 29–116.

Naddef, D., E. Wild. 2003. The domino inequalities: Facets for the symmetric traveling salesman polytope. *Math. Programming* **98** 223–251.

Padberg, M. W., M. R. Rao. 1982. Odd minimum cut-sets and b-matchings. *Math. Oper. Res.* **7** 67–80.

Padberg, M., G. Rinaldi. 1991. A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems. *SIAM Rev.* **33** 60–100.

Reinelt, G. 1991. TSPLIB—A traveling salesman library. *ORSA J. Comput.* **3** 376–384.

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.

Vella, D. 2001. *Using DP-Constraints to Obtain Improved TSP Solutions*. M.S. thesis, School of Information Technology and Engineering, University of Ottawa, Ottawa, Canada.