Implementing Domino-Parity Inequalities for the Traveling Salesman Problem

William Cook*
bico@isye.gatech.edu

Daniel Espinoza
despinoz@isye.gatech.edu

Marcos Goycoolea
mgoycool@isye.gatech.edu

Industrial and Systems Engineering
Georgia Institute of Technology
765 Ferst Drive NW
Atlanta, GA. 30332. USA.

May 30, 2005

Abstract

We describe an implementation of Letchford's domino-parity inequalities for the (symmetric) traveling salesman problem. The 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 set of safe-shrinking routines, a random-walk heuristic to extract additional violated constraints, and a tightening routine to allows us to modify existing domino-parity inequalities as the LP solution changes. We report computational results showing that combining the new separation algorithms with the Concorde TSP code allow us to substantially raise the linear programming bounds that are obtained.

1 Introduction

Let $G = (V, E)$ be a complete graph with edge costs $(c_e : e \in E)$. The symmetric traveling salesman problem, or TSP, is to find a minimum-cost tour in G, that is, a Hamiltonian cycle of minimum total edge cost. A tour can be represented as a 0-1 vector $x = (x_e : e \in E)$, where $x_e = 1$ if edge e is used in the tour and $x_e = 0$ otherwise. In the Dantzig, Fulkerson, and Johnson [6] cutting-plane method for the TSP, a linear programming (LP) relaxation is created by iteratively finding linear inequalities that are satisfied by all tour vectors. This

*Supported by ONR Grant N00014-03-1-0040

approach has been the most successful exact solution procedure proposed to date for the TSP.

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

Every tour of G satisfies the *subtour-elimination constraints*

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

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

There is a large research literature devoted to the study of classes of inequalities that are valid for the TSP, generalizing and extending the subtour-elimination constraints in many different ways. (For a survey see Naddef [10].) Many properties of these classes of inequalities are known, but for the most part polynomial-time separation algorithms have proven to be elusive. A notable exception is the separation algorithm for blossom-inequalities by Padberg and Rao [12]; variations of the Padberg-Rao algorithm are included in most current codes for the TSP. The absence of other efficient separation algorithms has lead to the use of various heuristic methods for handling TSP inequalities within cutting-plane algorithms. The heuristics have proven to be effective in many cases (see Padberg and Rinaldi [13], Applegate et al. [1], and Naddef and Thienel [11]), but additional exact methods could be critical in pushing TSP codes on to larger test instances.

An interesting new approach to TSP separation problems was adopted by Letchford [9], building on earlier work of Fleischer and Tardos [7]. Given an LP solution vector x^*, the support graph G^* is the sub-graph of G induced by the edge-set $E^* = \{e \in E : x^*_e > 0\}$. Letchford [9] introduces a new class of TSP inequalities, called domino-parity constraints, and provides a separation algorithm in the case where G^* is a planar graph. An initial computational study of this algorithm by Boyd et al. [3], combining a computer implementation with by-hand computations, showed that the method can produce strong cutting planes for instances with up to 1,000 nodes.

In this paper we present a further study of Letchford's algorithm, reporting computational results on a range of TSPLIB test instances. The domino parity constraints are described in Section 2, together with a review of results from Letchford [9] and a short 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 also to handle the common case where G^* is not planar. A local-search procedure for improving domino-parity constraints is described in Section 4 and computational results are presented in Section 5.

2 Letchford's Algorithm

We begin by introducing the domino-parity constraints and giving an overview of Letchford's separation algorithm. We highlight some important algorithmic steps, placing emphasis on

our implementation. All lemmas and theorems not proved in this section can be found in Letchford [9].

Let $\Lambda = \{E_1, E_2, \ldots, E_k\}$, where $E_i \subseteq E$, $\forall i = 1, \ldots k$. Define $\mu_e = |\{F \in \Lambda : e \in F\}|$ The family Λ is said to *support* the cut $\delta(H)$ if $\delta(H) = \{e \in E : \mu_e \text{ is odd}\}$. Define a *domino* as a pair $\{A, B\}$ satisfying $\emptyset \neq A, B \subseteq V$, $A \cap B = \emptyset$, and $A \cup B \neq V$.

Theorem 1 *Let p be a positive odd integer, let $\{A_j, B_j\}$ be dominoes for $j = 1, \ldots, p$, and let $H \subseteq V$. Suppose that $F \subseteq E$ is such that $\{E(A_1 : B_1), \ldots, E(A_p : B_p), F\}$ supports the cut $\delta(H)$ and define μ_e accordingly. Then, the domino-parity (DP) constraint,*

$$\sum_{e \in E} \mu_e x_e + \sum_{j=1}^{p} x(\delta(A_j \cup B_j)) \geq 3p + 1 \tag{2}$$

is valid for all tours.

The set H is called the handle of the constraint.

Letchford [9] proposes a two-stage algorithm which exactly separates this class of constraints, provided that the support graph G^* is planar and all subtour-elimination constraints are satisfied. 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 in such a way as to define a maximally violated constraint, provided one exists.

For the remainder of this section, assume that all of the subtour-elimination constraints are satisfied. Also, assume that G^* is a planar graph and let $\overline{G^*}$ be the planar dual of G^*. For any subset $F \subseteq E(G^*)$, denote by $\overline{F}$ the corresponding edges in $\overline{G^*}$.

Lemma 2.1 *Consider $s, t \in V(\overline{G^*})$ and three node-disjoint s-t paths P_1, P_2, P_3 in $\overline{G^*}$. There exists a domino $\{A, B\}$ such that $\overline{\delta(A \cup B)} \cap E(\overline{G^*}) \cup \overline{E(A : B)} \cap E(\overline{G^*}) = E(P_1) \cup E(P_2) \cup E(P_3)$.*

Given three paths in $\overline{G^*}$ as described above it is easy to construct a domino in G^*. Surprisingly, it suffices to use only these dominoes as candidates. The following algorithm describes the procedure by which to obtain them.

ALGORITHM 2.1 GENERATING CANDIDATE DOMINOES

Set $\mathcal{L} = \emptyset$.
Compute $\overline{G^*}$, the dual of graph G^*.
To each edge $e \in E(\overline{G^*})$ assign weight x^*_e.
for $s, t \in V(G^*)$:
  Find three edge disjoint paths P_1, P_2, P_3 of minimum weight joining s and t.
  if $x^*(P_1 \cup P_2 \cup P_3) \leq 4$.
    Construct a domino D_{st} and add it to $\mathcal{L}$.
    Define $\hat{w}_{st} = x^*(P_1 \cup P_2 \cup P_3) - 3$.
return $\mathcal{L}$

To solve the second stage of the problem, define an auxiliary multigraph M^* with node set $V(M^*) = V(\overline{G^*})$. For each edge $e = \{u, v\} \in E(\overline{G^*})$ define an *even* edge $e = \{u, v\} \in E(M^*)$ with weight $w_e = x^*_e$, and for each $D_{uv} \in \mathcal{L}$ define an *odd* edge $e = \{u, v\} \in E(M^*)$ with weight $w_e = \hat{w}_{uv}$. An *odd cycle* in M^* is a cycle with an odd number of odd edges.

Lemma 2.2 *Given an odd cycle $C \subseteq E(M^*)$ with weight $w(C) < 1$ it is possible to construct a DP-inequality with violation $1 - w(C)$.*

In fact, to construct the DP-inequality from the cycle it suffices to define the set F as the even edges in C, and choose as dominoes those corresponding to odd edges in C.

Theorem 2 *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.*

From these results the following algorithm directly follows.

ALGORITHM 2.2 SEPARATION OF DP-INEQUALITIES

Build M^* as defined above.
Solve the minimum weight odd cycle problem in M^*.
  Let C^* be an optimal solution.
if $w(C^*) < 1$
  return corresponding DP-inequality.
else
  return no violated DP-inequality exists.

We briefly describe the techniques that had the greatest impact in speeding-up our implementation of the algorithm.

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

Flow Computations To find the min-weight node-disjoint paths between pairs $s, t \in V(\overline{G^*})$ we used the augmenting-shortest-path network-flow algorithm. For this, we build a network $\mathcal{N}$ for the graph $\overline{G^*}$ defining two arcs for each edge (one in each direction), assigning a capacity of one to each. This algorithm computes the $s-t$ flow by solving three successive $s-t$ shortest path problems on reduced capacity networks successively derived from $\mathcal{N}$. Using this algorithm several speed-ups were possible. Firstly, for fixed $s \in V(\overline{G^*})$ the first $s-t$ flow for all nodes t can be obtained by solving a single Dijkstra algorithm in $\mathcal{N}$ rooted at s. The additional shortest-path computations need only be computed for nodes t at distance not greater than 4/3 from s. Finally, when computing the $s-t$ three-flow in $\mathcal{N}$, one only need consider intermediary nodes at distance not greater than 2 from s and t. This follows from the fact that every cycle in $\overline{G^*}$corresponds to a cut in G^*, and hence, has weight at least 2 (due to subtour-elimination constraints). Thus, if a node is used which has distance at least 2, since the other two paths will define a cycle, the bound of 4 would be exceeded.

Random Walk The algorithm as formally defined in Letchford [9] computes exactly one constraint. In practice, one would like the algorithm to compute as many violated constraints as possible. To achieve this, instead of just solving the shortest odd-cycle problem

in M^* we additionally run a random walk algorithm that attempts to find small-weight odd cycles. This algorithm is fast, easy to implement, and in our tests generally produced a large number of additional cuts, only the best of which were kept.

3 Shrinking and Non-Planar Graphs

Given a support graph G^* it may happen that we are unable to separate a solution x^* by use of DP-inequalities because either G^* is too large or G^* is not planar. In either case it may still be possible to do separation by contracting edges in G^*, redefining the vector x^*, and solving the separation problem in the new, smaller, graph. This procedure is called *shrinking*; details for general TSP inequalities can be found in Padberg and Rinaldi [13].

A natural concern is, if we know there is a violated DP-inequality in G^*, can we assure that there will also be a violated DP-inequality in the graph obtained after a contraction? If so, the contraction is called *safe*. Although it is not always the case that shrinking is safe, it is possible to give conditions under which it will be.

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

This suggests the following algorithm. While there exist nodes u, v, t satisfying the conditions of Theorem 3, contract edge $\{u, v\}$. This procedure can greatly reduce the size of the graph over which we work, but the resulting graph may still be non-planar. If this is the case, our implementation does non-safe shrinks until a planar graph is obtained, as in Boyd et al. [3].

A graph is planar if and only if it contains no $K_{3,3}$ or K_5 minor. This suggests the following natural heuristic. If G^* is not planar, identify a forbidden minor $M \subseteq E(G^*)$. Choose an edge $e \in M$, and contract it, iterating until a planar graph is obtained. There are several ways in which M and $e \in M$ may be selected, and we found that the way in which the selection is made can make an important difference in the performance of the algorithm.

4 Tightening

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

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

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

5 Computational Results

In Tables 1 and 2 we report on a set of tests on medium-sized instances from the TSPLIB. The computations were performed on a single processor of a dual 2.66 GHz Intel Xeon workstation. The LP solver used was ILOG CPLEX 6.5. The algorithm used for planarity testing was Boyer and Myrvold[2] [4]. In these tests we used the Concorde command line option -mC48 to allow Concorde to repeatedly call the local-cuts routine up to size 48 (see Applegate et al. [2]); this setting requires additional CPU time over the default version of Concorde, but it allows Concorde to obtain substantially better lower bounds. (Local-cuts of size 48 are the largest value that can be effectively handled in Concorde.)

Table 1: DP-Cuts on Mid-Sized Instances

Name	Optimal	Concorde	Concorde+DP	Gap Closed by DP
pcb3038	137694	137660	137687	79%
fnl4461	182566	182555	182559	36%
rl5915	565530	565384	565482	67%
rl5934	556045	555929	556007	67%
pla7396	23260728	23255280	23259532	78%

In each case, the addition of the DP-separator routines made a significant in the LP lowerbound, with a resonable amount of extra running time. We will carry out much more extensive testing this Fall and Winter, on instances with up to 85,900 nodes.

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

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

Table 2: Times for Mid-Sized Instances (CPU Hours)

Name	Concorde	Concorde+DP
pcb3038	24.9	8.6
fnl4461	7.9	3.4
rl5915	103.7	46.1
rl5935	17.5	48.3
pla7396	133.7	106.9

References

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

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

[3] Boyd, S., S. Cockburn, D. Vella. 2001. On the domino-parity inequalities for the STSP. Computer Science Technical Report TR-2001-10. School of Information Technology, and Engineering, University of Ottawa.

[4] Boyer, J. M., W. Myrvold. 2004. On the cutting edge: simplified O(n) planarity by edge addition. Journal of Graph Algorithms and Applications. To appear.

[5] Chvátal, V. 1973. Edmonds polytopes and weakly hamiltonian graphs. Mathematical Programming **5**, 29–40.

[6] Dantzig, G., R. Fulkerson, S. Johnson. 1954. Solution of a large-scale traveling salesman problem. Operations Research **2**, 393–410.

[7] Fleischer, L., É. Tardos. 1999. Separating maximally violated comb inequalities in planar graphs. Mathematics of Operations Research **24**, 130–148.

[8] Grötschel, M., O. Holland. 1991. Solution of large-scale symmetric travelling salesman problems. Mathematical Programming **51**, 141–202.

[9] Letchford, A. N. 2000. Separating a superclass of comb inequalities in planar graphs. Mathematics of Operations Research **25**, 443–454.

[10] Naddef, D. 2002. Polyhedral theory and branch-and-cut algorithms for the symmetric traveling salesman problem. In G. Gutin and A. Punnen, editors. *The Traveling Salesman Problem and Its Variations*. Kluwer, Dordrecht, pp. 29–116.

[11] Naddef, D., S. Thienel. 2002. Efficient separation routines for the symmetric traveling salesman problem II: separating multi handle inequalities. Mathematical Programming **92**, 257–283.

[12] Padberg, M. W., M. R. Rao. 1982. Odd minimum cut-sets and b-matchings. Mathematics of Operations Research **7**, 67–80.

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