Operations Research Letters 37 (2009) 11–15

Contents lists available at ScienceDirect

Operations Research Letters

journal homepage: www.elsevier.com/locate/orl

Certification of an optimal TSP tour through 85,900 cities

David L. Applegate [a], Robert E. Bixby [b], Vašek Chvátal [c], William Cook [d,*], Daniel G. Espinoza [e], Marcos Goycoolea [f], Keld Helsgaun [g]

[a] AT&T Labs - Research, United States
[b] Graduate School of Management, Rice University, United States
[c] Department of Computer Science and Software Engineering, Concordia University, Canada
[d] School of Industrial and Systems Engineering, Georgia Tech, Atlanta, GA, 30332, United States
[e] Departamento de Ingeniería Industrial, Universidad de Chile, Chile
[f] School of Business, Universidad Adolfo Ibáñez, Chile
[g] Department of Computer Science, Roskilde University, Denmark

ARTICLE INFO

Article history:
Received 1 October 2007
Accepted 26 September 2008
Available online 15 October 2008

Keywords:
Traveling salesman problem
Integer programming
Linear programming

ABSTRACT

We describe a computer code and data that together certify the optimality of a solution to the 85,900-city traveling salesman problem pla85900, the largest instance in the TSPLIB collection of challenge problems.

1. Introduction

An important aspect of computational research is the verifiability of reported benchmark results. General remarks on this topic are given in [20], as part of a broad discussion on the state of the literature in the field of operations research. In this note we consider verification within the context of the traveling salesman problem (TSP).

Computational work on the TSP began in earnest with the classic paper of Dantzig et al. [7], where the cutting-plane algorithm is used to compute an optimal TSP tour through 49 cities in the United States. In this 1954 study, the authors are directly concerned with a certification of their result, writing:

"*The optimal tour $\bar{x}$ is shown in Fig. 16. The proof that it is optimal is given in Fig. 17*".

Dantzig et al.'s proof of optimality consists of a linear programming (LP) relaxation of the TSP, together with a dual LP solution providing a lower bound that matches the travel cost of their proposed tour.

The short, hand-verifiable, proof of [7] cannot be duplicated on the large TSPLIB [19] instances studied by Applegate et al. [2], the largest example, pla85900, having 85,900 cities. As an alternative, the publicly available Concorde TSP code [3] employs an architecture that ensures the validity of the lower-bound computations, independent of the accuracy of the LP software used to drive the cutting-plane implementation. Nevertheless, in the particular case of pla85900, the very long computation, consuming over 136 CPU years, makes the solution difficult to replicate and verify.

In this paper, we describe a computer-checkable TSP proof, along with a description of the certification data for pla85900. The certification data consists of a TSP tour together with a rooted search tree and an associated LP relaxation for each leaf of the tree. The full set of leaves comprises a collection of relaxations such that every TSP tour is a feasible solution to exactly one relaxation in the collection. For each leaf, a dual LP solution is stored that establishes a lower bound for a restricted version of the corresponding TSP subproblem. The lower bounds together prove that the specified tour is optimal.

The full proof-checking code and the certification data for pla85900 are available on the web page www.tsp.gatech.edu/pla85900/proof. The computer code and data permit the verification of the optimal pla85900 tour with a modest amount of human checking of the source code and a reasonable length computation with the certification data.

* Corresponding author.
E-mail addresses: david@research.att.com (D.L. Applegate), bixby@rice.edu (R.E. Bixby), chvatal@cse.concordia.ca (V. Chvátal), bico@isye.gatech.edu (W. Cook), daespino@dii.uchile.cl (D.G. Espinoza), marcos.goycoolea@uai.cl (M. Goycoolea), keld@ruc.dk (K. Helsgaun).

0167-6377/$ – see front matter
doi:10.1016/j.orl.2008.09.006

2. Problem description

A TSP instance can be described by a complete graph with node-set V, edge-set E, and edge costs $(c_e : e \in E)$. The problem is to find a tour through V of minimum total edge cost, where a tour is a cycle that visits each node exactly once.

The instances in the TSPLIB have c_e integer for all $e \in E$. In the case of pla85900, these costs are determined by pairs of coordinates (x_v, y_v) for each node $v \in V$; the value of c_e for an edge $e = (u, v)$ joining nodes u and v is the Euclidean distance

$$\sqrt{(x_u - x_v)^2 + (y_u - y_v)^2}$$

rounded up to the nearest integer. All coordinates for pla85900 are integers in the range [548000, 1452000].

3. Tour

A tour can be recorded in our certification data as a permutation of the cities, leaving a simple computation to verify its cost from the TSPLIB information. In the case of the pla85900 TSP, the optimal tour value 142,382,641 matches the cost of a tour first obtained with the LKH 2.0 heuristic code [12,13].

4. LP relaxation

The LP relaxations we consider have variables $(x_e : e \in E)$, with a tour T represented by assigning $x_e = 1$ for all edges e in T and $x_e = 0$ for all edges not in T. For any subset S of 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 of V is called a *cut*. For disjoint sets $S, W \subseteq V$ let $E(S, W)$ denote the set of edges having one end in S and one end in W. For any set $F \subseteq E$ define $x(F) = \sum(x_e : e \in F)$.

The *degree LP relaxation*

$$\begin{aligned} &\text{minimize} \sum(c_e x_e : e \in E) \\ &\text{subject to} \\ &x(\delta(\{v\})) = 2 \quad \text{for all } v \in V \\ &0 \le x_e \le 1 \quad \text{for all } e \in E, \end{aligned}$$

is the starting point for the TSP cutting-plane algorithm.

Many of the standard inequalities for the TSP, such as subtour inequalities [7], comb inequalities [5,9] and clique-tree inequalities [10], are specified by giving a family $\mathcal{S}$ of (not necessarily distinct) subsets of V and an integer β such that

$$\sum(x(\delta(S)) : S \in \mathcal{S}) \ge \beta \tag{1}$$

is satisfied by all tour vectors. We say that (1) is in *hypergraph format*.

In addition to inequalities in hypergraph format, we also include in our LP relaxation a particular class of constraints known as *domino-parity* (DP) inequalities [16]. Although these inequalities can also be written in hypergraph format, the natural form presented below results in a large saving in space.

A *domino* D is a pair $\{D_A, D_B\}$ of non-empty subsets of V satisfying $D_A \cap D_B = \emptyset$ and $D_A \cup D_B \neq V$. Consider a set of dominoes $\mathcal{D}$ with $|\mathcal{D}|$ odd, together with an additional set $F_{\mathcal{D}}$ of edges such that for some set $H_{\mathcal{D}}$ of nodes the cut $\delta(H_{\mathcal{D}})$ is precisely the set of edges that appear an odd number of times among the sets $F_{\mathcal{D}}$ and $(E(D_A, D_B) : D \in \mathcal{D})$. Note that $F_{\mathcal{D}}$ is determined by $\mathcal{D}$ and $H_{\mathcal{D}}$. The domino-parity inequality corresponding to $(\mathcal{D}, H_{\mathcal{D}})$ is

$$x(F_{\mathcal{D}}) + \sum_{D \in \mathcal{D}} (x(E(D_A, D_B)) + x(\delta(D_A \cup D_B))) \ge 3|\mathcal{D}| + 1.$$

Our LP relaxation consists of the degree LP together with a collection of inequalities in hypergraph format specified by set families $\mathcal{S}_1, \ldots, \mathcal{S}_r$ and right-hand-side values $\beta_1, \ldots, \beta_r$, and a collection of domino-parity inequalities specified by $(\mathcal{D}_1, H_{\mathcal{D}_1}), \ldots, (\mathcal{D}_s, H_{\mathcal{D}_s})$.

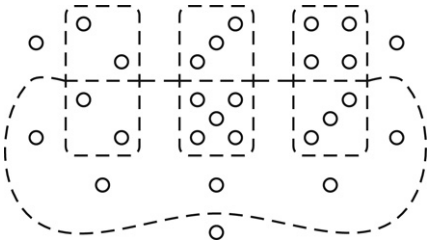

Fig. 1. Venn diagram representing a comb inequality on a graph with 27 nodes.

5. Validating TSP inequalities

To use an LP relaxation in an optimality proof, we must establish that each of the inequalities in the system of constraints is indeed valid for all tour vectors. In the case of a domino-parity inequality, this is a simple matter of checking that the pair $(\mathcal{D}, H_{\mathcal{D}})$ satisfies the definition given above. The process is more complicated, however, for general inequalities in hypergraph format.

For a hypergraph inequality $(\mathcal{S}, \beta)$, we first check if $\mathcal{S}$ matches one of the entries in a list of known templates, such as subtours, combs, or clique trees. In this process, we consider the possibility of replacing some members S of $\mathcal{S}$ by their complements $V \backslash S$ to obtain a set family matching a given template, and yielding an equivalent inequality. This procedure may fail however, since the *local cuts* routine adopted in Concorde can deliver cutting planes that do not match any of the known classes of inequalities for the TSP. For these unclassified inequalities, we must rely on a different verification method.

Given an inequality $\sum(\pi_e x_e : e \in E) \ge \pi_0$, it is possible to determine if it is valid for all tours by solving the TSP with edge costs $(\pi_e : e \in E)$ and comparing the optimal value to π_0. This approach is not practical, however, since for each inequality it requires the solution of a TSP of size equal to that of our original problem. Fortunately, in the case of hypergraph inequalities it is possible to reduce the problem to that of verifying the validity of a similar inequality in a smaller sized instance of the TSP.

Define the *atoms* of the hypergraph $(V, \mathcal{S})$ as the minimal non-empty regions in a Venn diagram representation of $\mathcal{S}$. That is, an atom is a non-empty set of the form

$$\bigcap_{S \in \mathcal{S}_{\text{in}}} S \backslash \bigcup_{S \in \mathcal{S}_{\text{out}}} S$$

for a partition of $\mathcal{S}$ into subfamilies $\mathcal{S}_{\text{in}}, \mathcal{S}_{\text{out}}$. Here, the intersection of an empty collection is defined as V and the union of an empty collection is defined as the empty set.

Let $V' \subseteq V$ be obtained by selecting a single representative for each non-empty atom of $(V, \mathcal{S})$; the set V' is called a *backbone* of the hypergraph $(V, \mathcal{S})$. See Figs. 1 and 2 for an example. Now let $\mathcal{S}'$ denote the collection of sets $(V' \cap S : S \in \mathcal{S})$. In Chapter 5 of [2], it is observed that the hypergraph inequality (1) is valid for all tours through V if and only if

$$\sum(x(\delta(S)) : S \in \mathcal{S}') \ge \beta \tag{2}$$

is valid for all tours through V'.

These smaller TSP instances on V' can be solved, for example, with an implementation of the branch-and-bound algorithm of Held and Karp [11].

In cases where the backbone TSP requires a long solution time with our Held–Karp code, we alternatively validate (2), in a recursive fashion, by storing certification data of the general form described in this paper. These stored backbone TSP proofs are then checked as part of the proof-checking process for the full TSP.

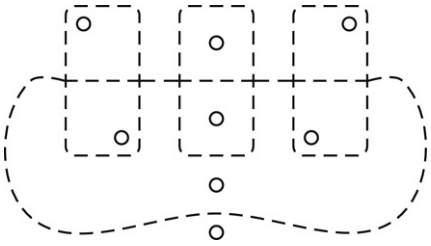

Fig. 2. Venn diagram representing a backbone (on 8 nodes) of the hypergraph inequality depicted in Fig. 1.

6. Lower bounds and dual feasible solutions

The variables in the dual LP for the TSP relaxation are $y = (y_v : v \in V)$, $Y = (Y_i : i = 1, \ldots r)$, $U = (U_i : i = 1, \ldots s)$, and $z = (z_e : e \in E)$, corresponding to, respectively, the degree equations, the hypergraph inequalities, the domino-parity inequalities, and the bounds $x_e \leq 1$ for all $e \in E$. The *reduced cost* of $e = (u, v) \in E$, denoted by α_e, is given by the formula

$$\begin{aligned}\alpha_e = c_e &- y_u - y_v \\ &- \sum_{i=1}^{r} |\{S \in \mathcal{S}_i : e \in \delta(S)\}| Y_i \\ &- \sum_{i=1}^{s} |\{D \in \mathcal{D}_i : e \in E(D_A, D_B)\}| U_i \\ &- \sum_{i=1}^{s} |\{D \in \mathcal{D}_i : e \in \delta(D_A \cup D_B)\}| U_i \\ &- \sum_{i=1}^{s} |e \cap F_{\mathcal{D}_i}| U_i.\end{aligned}$$

With these values, the dual LP constraints can be written as

$$\alpha_e + z_e \geq 0 \quad \text{for all } e \in E \tag{3}$$

together with the nonnegativity requirements $Y \geq 0$, $U \geq 0$, and $z \geq 0$.

A problem with standard linear programming (LP) solvers is that they often return invalid (or non-optimal) solutions due to the use of floating point arithmetic. Since branch-and-bound algorithms use the values of LP relaxations as bounds to optimal solutions, invalid LP solutions can potentially lead to incorrect bounds. In order to deal with this, observe that the z variables permit us to convert any values of y, Y, and U to a valid dual LP solution, by increasing Y and U to be nonnegative and setting z so that (3) is satisfied. Note that the z variables appear with coefficient -1 in the dual LP objective function, so slight increases in z result in slight decreases in the dual LP lower bound for the TSP. Since typically the solution values y, Y, z, U obtained from LP solvers, when not feasible, are near-feasible, this procedure results in tight bounds in practice.

Following the technique used in Concorde, we record the values of y, Y, and U in fixed-point arithmetic, representing each number by 32 bits before the decimal point and 32 bits after the decimal point. Addition and subtraction of such fixed-point numbers can be carried out without rounding errors, provided the operations are checked for possible overflow. This process allows us to obtain a valid lower bound on the cost of a TSP tour through V.

7. Pricing and edge elimination

The application of the above lower-bounding technique requires, in order to determine which of the dual constraints (3) is invalid, that we compute the reduced cost α_e for any edge $e \in E$ such that $\alpha_e < 0$; this is known as edge *pricing*. Although it is a straightforward computation, the quadratic number of edges in E makes pricing a time-consuming process. As a practical speed-up, Applegate et al. [1] use an easily computed lower bound on α_e to avoid its exact calculation when possible.

The technique used in [1] was developed for hypergraph inequalities; we generalize it to handle domino-parity inequalities. For each $v \in V$ define

$$\begin{aligned}\gamma_v = y_v &+ \sum_{i=1}^{r} |\{S \in \mathcal{S}_i : v \in S\}| Y_i \\ &+ \sum_{i=1}^{s} |\{D \in \mathcal{D}_i : v \in D_A \cup D_B\}| U_i \\ &+ \sum_{i=1}^{s} |v \cap H_{\mathcal{D}_i}| U_i.\end{aligned}$$

Note that, for any $i = 1, \ldots, s$, an edge $e \in F_{\mathcal{D}_i}$ is either a member of $\delta(H_{\mathcal{D}_i})$ or included in at least one of the sets $(E(D_A, D_B) : D \in \mathcal{D}_i)$. Therefore, for $e = (u, v) \in E$ we have

$$\alpha_e \geq c_e - \gamma_u - \gamma_v.$$

We need only compute α_e for those edges having $c_e - \gamma_u - \gamma_v < 0$.

Let t^* denote the cost of the TSP tour stored in the certification data and let l^* be a valid lower bound provided by the technique described in the previous section. Using a standard technique from integer programming, the full set of edges can be reduced by eliminating from further consideration any $e \in E$ satisfying $\alpha_e > t^* - l^* - 1$ and setting $x_e = 1$ for any $e \in E$ satisfying $\alpha_e < -(t^* - l^* - 1)$.

By storing the value of l^* as part of the certification data, the pricing and elimination phases can be carried out in a single pass through the full edge set, computing α_e only for those edges having

$$c_e - \gamma_u - \gamma_e \leq \max(0, t^* - l^* - 1).$$

The procedure concludes by verifying that the lower bound determined by the dual LP solution is at least as great as the stored value l^*.

8. Branch-and-bound tree

The Concorde TSP code follows a *branch-and-cut* scheme [14,18], embedding the cutting-plane algorithm within a branch-and-bound search. Following the technique of Clochard and Naddef [6], subproblems are created in the branching step by choosing a proper subset $S \subseteq V$ and adding the condition $x(\delta(S)) = 2$ to form one child subproblem and $x(\delta(S)) \geq 4$ to form the second child subproblem. Note that this branching step includes as a special case the traditional method of choosing an edge $e \in E$ and adding the conditions $x_e = 0$ and $x_e = 1$ to form the two subproblems.

The branching steps create a search tree, with the original problem as the root node. To simplify the certificate of optimality, all cutting planes that are used in the leaves of the search tree are added to the root LP relaxation. To traverse the search tree, therefore, we need only to add the corresponding branching condition to move to a child node. By traversing the tree we establish that the leafs correspond to a set of subproblems that include every tour for our TSP.

In the proof-checking code, the tree is traversed using depth-first search. A lower bound is established at each leaf using the stored LP dual solution and the bound is verified to be greater than the optimal tour value minus one.

Table 1
Distribution of inequalities in pla85900 LP.

Class	Number in LP
Subtour [7]	2,027
Comb [5,9]	5,216
Star [8,17]	8,982
Clique-tree and bipartition [10,4]	49
Domino-parity [16]	861
Unclassified	1,177

9. The pla85900 proof

Our computer code for the proof-checking procedure is written in the C programming language [15], with a total of 6646 lines; an LP solver is not used in the verification process. The proof-checker is constructed to handle general TSP instances specified in TSPLIB format, but it requires problem-specific certification data to be given as an input file. To demonstrate the code we provide a certification file for the 532-city instance att532, together with a certification file for pla85900. The size of the pla85900 file is 8.1 MB when compressed with the gzip utility, and 32.2 MB when uncompressed.

The pla85900 certification file includes a permutation of the cities in the claimed optimal tour order. The cost of the tour is 142,382,641 as computed using the problem information in the TSPLIB file; this tour cost is used at the target value for the verification. The tour is also used to store the sets of the hypergraph and domino-parity inequalities in a compressed format, where each set is specified as a list of intervals in the permutation. This compression scheme is adopted in the Concorde code and it is described in detail in Chapter 12 of [2].

The pla85900 LP has 18,312 inequalities. The checking code first verifies that 17,135 of these inequalities match known templates for the TSP, as indicated in Table 1.

An additional five inequalities are verified by stored proofs for their backbone TSPs. Of the remaining 1,172 unclassified inequalities, 346 of the corresponding hypergraphs are isomorphic to others in the collection. The certification data provides the indices of the isomorphic hypergraphs; after renumbering the nodes, each isomorphism is verified by a comparison of the sets of the hypergraphs. The finally remaining 826 inequalities are verified with an implementation of the Held–Karp branch-and-bound algorithm.

These inequalities, together with the stored dual LP solution, give a lower bound of

142381678.191306799650

on the length of any tour for pla85900. This value is stored in the data file, to permit the checker to eliminate edges in a single pricing pass through the complete set of 3.7 billion edges that define the pla85900 TSP. The elimination phase reduces the problem to 265,259 edges, with 1,662 of the corresponding variables fixed to the value one.

The search tree for pla85900 is depicted in Fig. 3, where the height of a node in the tree corresponds to the lower bound given by the corresponding LP relaxation; the cut-off value of 142,382,640 is indicated by the thick line at the bottom of the figure. Note that the leaves of the tree not drawn as nodes all have LP bounds greater than 142,382,700. Using the reduced 265,259-edge problem, the code verifies that each of the leaves of the tree has a lower bound larger than 142,382,640, using the stored dual LP solutions. Since all edge costs are integer, this establishes the optimality of the 142,382,641 tour.

The pla85900 verification was run on several different computing platforms. The total running time on a Sun Microsystems V20z compute node, equipped with a 2.4-GHz AMD Opteron 250 processor, was approximately 568.9 h. The distribution of the running time over each phase is given in Table 2. The bulk of the time is used in the Held–Karp runs to verify the unclassified inequalities, the longest of these requiring 39.6 h. The proof-checking code includes master-worker parallel software to permit Held–Karp computations to be carried out on a network of computers.

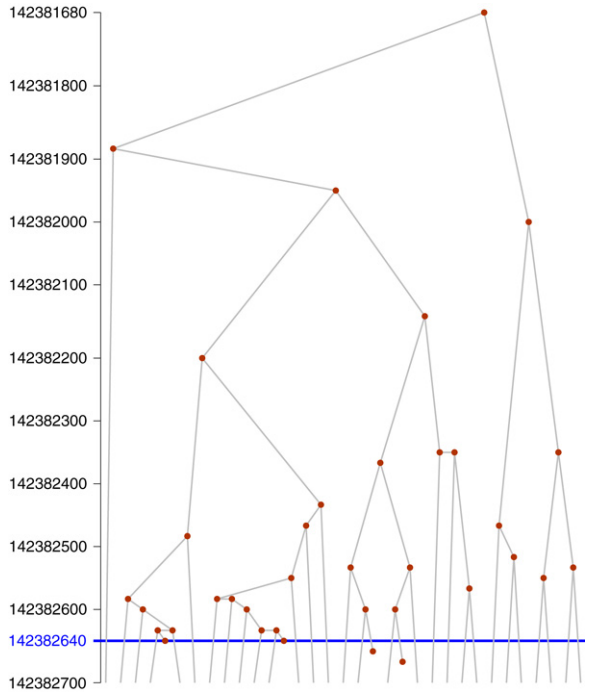

Fig. 3. Branching tree for pla85900.

Table 2
Running time for pla85900 verification.

Phase of Algorithm	CPU Time
Classifying inequalities	1.8 s
Held–Karp TSPs for 826 inequalities	568.5 h
Edge pricing and elimination	1465.6 s
Lower bounds at leaves of tree	69.9 s

The five inequalities that are verified with stored branch-and-cut proofs required even longer Held–Karp runs in preliminary computations, with each backbone TSP solution time exceeding 100 h. Note that the overall running time of the checking code could be reduced considerably by including additional branch-and-cut proofs for backbone TSPs, but this would greatly increase the size of the certification file due to the rather large branch-and-cut trees that are created for some of the small, but difficult, backbone instances.

Acknowledgements

The work of W. Cook was supported by ONR Grant N00014-03-1-0040 and by NSF Grant CMMI-0726370.

References

[1] D. Applegate, R.E. Bixby, V. Chvátal, W. Cook, Implementing the Dantzig–Fulkerson–Johnson algorithm for large traveling salesman problems, Math. Program. 97 (2003) 91–153.
[2] D. Applegate, R.E. Bixby, V. Chvátal, W. Cook, The Traveling Salesman Problem: A Computational Study, Princeton University Press, Princeton, New Jersey, USA, 2006.
[3] D. Applegate, R.E. Bixby, V. Chvátal, W. Cook, Concorde. Available at: www.tsp.gatech.edu.
[4] S.C. Boyd, W.H. Cunningham, Small travelling salesman polytopes, Math. Oper. Res. 16 (1991) 259–271.