# TSP Cuts Which Do Not Conform to the Template Paradigm

David Applegate[1], Robert Bixby[2], Vašek Chvátal[3], and William Cook*[4]

[1] Algorithms and Optimization Department, AT&T Labs – Research, Florham Park, NJ 07932, USA
[2] Computational and Applied Mathematics, Rice University, Houston, TX 77005, USA
[3] Department of Computer Science, Rutgers University, Piscataway, NJ 08854, USA
[4] Program in Applied and Computational Mathematics, Princeton University, Princeton, NJ 08544, USA

**Abstract.** The first computer implementation of the Dantzig-Fulkerson-Johnson cutting-plane method for solving the traveling salesman problem, written by Martin, used subtour inequalities as well as cutting planes of Gomory's type. The practice of looking for and using cuts that match prescribed templates in conjunction with Gomory cuts was continued in computer codes of Miliotis, Land, and Fleischmann. Grötschel, Padberg, and Hong advocated a different policy, where the template paradigm is the only source of cuts; furthermore, they argued for drawing the templates exclusively from the set of linear inequalities that induce facets of the TSP polytope. These policies were adopted in the work of Crowder and Padberg, in the work of Grötschel and Holland, and in the work of Padberg and Rinaldi; their computer codes produced the most impressive computational TSP successes of the nineteen eighties. Eventually, the template paradigm became the standard frame of reference for cutting planes in the TSP. The purpose of this paper is to describe a technique for finding cuts that disdains all understanding of the TSP polytope and bashes on regardless of all prescribed templates. Combining this technique with the traditional template approach was a crucial step in our solutions of a 13,509-city TSP instance and a 15,112-city TSP instance.

## 1 The Cutting-Plane Method and Its Descendants

The groundbreaking work of Dantzig, Fulkerson, and Johnson [19] on the traveling salesman problem introduced the *cutting-plane method*, which can be used to attack any problem

$$\text{minimize } c^T x \text{ subject to } x \in \mathcal{S}, \tag{1}$$

where $\mathcal{S}$ is a finite subset of some Euclidean space $\mathbb{R}^m$, provided that an efficient algorithm to recognize points of $\mathcal{S}$ is available. This method is iterative; each

* Supported by ONR Grant N00014-01-1-0058

M. Jünger and D. Naddef (Eds.): Computat. Comb. Optimization, LNCS 2241, pp. 261–303, 2001.
© Springer-Verlag Berlin Heidelberg 2001

of its iterations begins with a *linear programming relaxation* of (1), meaning a problem

$$\text{minimize } c^T x \text{ subject to } Ax \leq b, \tag{2}$$

where the polyhedron $P$ defined as $\{x : Ax \leq b\}$ contains $\mathcal{S}$ and is bounded. Since $P$ is bounded, we can find an optimal solution $x^*$ of (2) which is an extreme point of $P$. If $x^*$ belongs to $\mathcal{S}$, then it constitutes an optimal solution of (1); otherwise, some linear inequality *separates* $x^*$ *from* $\mathcal{S}$ in the sense of being satisfied by all the points in $\mathcal{S}$ and violated by $x^*$; such an inequality is called a *cutting plane* or simply a *cut*. In the latter case, we find a family of cuts, add them to the system $Ax \leq b$, and use the resulting tighter relaxation of (1) in the next iteration of the cutting-plane method.

Each iteration of the method requires first finding $x^*$ and then finding a family of cuts. Finding $x^*$ presents no problem: this is what the simplex method and other LP algorithms are for. Finding cuts is the challenge that has to be answered with each new application of the cutting-plane method; we shall return to this challenge later.

Progress of the cutting-plane method towards solving a particular instance of problem (1) is often estimated by the increase in the optimal value of its LP relaxation; as more and more cuts are added, these increases tend to get smaller and smaller. When they become unbearably small, the sensible thing to do may be to *branch*: having chosen a vector $\alpha$ and numbers $\beta', \beta''$ with $\beta' < \beta''$ such that

$$\alpha^T x^* \in (\beta', \beta'') \quad \text{and} \quad \{\alpha^T x : x \in \mathcal{S}\} \subset (-\infty, \beta'] \ \cup \ [\beta'', +\infty),$$

we solve the two *subproblems*,

$$\text{minimize } c^T x \text{ subject to } x \in \mathcal{S},\ \alpha^T x \leq \beta'$$

and

$$\text{minimize } c^T x \text{ subject to } x \in \mathcal{S},\ \alpha^T x \geq \beta'',$$

separately. (If all the elements of $\mathcal{S}$ are integer vectors and some component $x_e^*$ of $x^*$ is not an integer, then we may choose $\alpha$ so that $\alpha^T x$ is identically equal to $x_e$ and set $\beta' = \lfloor x_e^* \rfloor$, $\beta'' = \lceil x_e^* \rceil$.) At some later time, one or both of these two subproblems may be split into sub-subproblems, and so on; in the resulting binary tree of subproblems, each node has the form

$$\text{minimize } c^T x \text{ subject to } x \in \mathcal{S},\ Cx \leq d \tag{3}$$

for some system $Cx \leq d$ of linear inequalities and each leaf will have been either solved without recourse to branching or else found irrelevant since the optimal value of its LP relaxation turned out to be at least as large as $c^T x$ for some previously known element $x$ of $\mathcal{S}$. This scheme is one of the many variants of the *branch-and-bound* method. (The term "branch-and-bound", coined by Little, Murty, Sweeney, and Karel [44], refers to a general class of algorithms that originated in the work of Bock [7], Croes [16], Eastman [20], Rossman and Twery

[65], and Land and Doig [42]; in this more general context, relaxations of (1) may come from a universe far wider than that of linear programming relaxations (2) and each subproblem may be split into more than two sub-subproblems.)

Computer codes written by Hong [39], Miliotis [48], and Grötschel, Jünger, and Reinelt [33] introduced a particular variant of this variant, where each subproblem is attacked by the cutting-plane method; in these codes, the cuts introduced in solving (3) are satisfied by all points of $\mathcal{S}$ (rather than merely by all points $x$ of $\mathcal{S}$ which satisfy $Cx \leq d$), and so they can be passed to any other subproblem later on. Padberg and Rinaldi [60] termed this approach *branch-and-cut*.

## 2 Ways of Finding Cuts

The symmetric traveling salesman problem, or TSP for short, is this: given a finite number of "cities" along with the cost of travel between each pair of them, find the cheapest way of visiting all of the cities and returning to your starting point. The travel costs are symmetric in the sense that traveling from city X to city Y costs just as much as traveling from Y to X; the "way of visiting all the cities" is simply the order in which the cities are visited. This problem is a special case of (1) with $m = n(n-1)/2$, where $n$ is the number of the cities and $\mathcal{S}$ consists of the set of incidence vectors of all the hamiltonian cycles through the set $V$ of the $n$ cities; in this context, hamiltonian cycles are commonly called *tours*. Dantzig, Fulkerson, and Johnson illustrated the power of their cutting-plane method by solving a 49-city instance of the TSP, an impressive size at the time. They let the initial polyhedron $P$ consist of all vectors $x$, with components subscripted by edges of the complete graph on $V$, that satisfy

$$0 \leq x_e \leq 1 \quad \text{for all edges } e \tag{4}$$

and

$$\sum(x_e : v \in e) = 2 \quad \text{for all cities } v. \tag{5}$$

(Throughout this paper, we treat the edges of a graph as two-element subsets of its vertex-set: $v \in e$ means that vertex $v$ is an endpoint of edge $e$; $e \cap Q \neq \emptyset$ means that edge $e$ has an endpoint in set $Q$; $e - Q \neq \emptyset$ means that edge $e$ has an endpoint outside set $Q$; and so on.) All but two of their cuts have the form $\sum(x_e : e \cap Q \neq \emptyset, e - Q \neq \emptyset) \geq 2$, where $Q$ is a nonempty proper subset of $V$; they are satisfied by all tours through $V$ because every such tour has to move from $Q$ to $V - Q$ at least once and it has to move back to $Q$ after each such crossing. Dantzig, Fulkerson, and Johnson called such inequalities "loop constraints"; nowadays, they are commonly referred to as "subtour elimination inequalities"; we are going to call them simply *subtour inequalities*. (As for the two exceptional cuts, Dantzig, Fulkerson, and Johnson give ad hoc combinatorial arguments to show that these inequalities are satisfied by incidence vectors of all tours through the 49 cities and, in a footnote, they say "We are indebted to I. Glicksberg of Rand for pointing out relations of this kind to us".)

An important class of problems (1) are the *integer linear programming* problems, where $\mathcal{S}$ is specified as the set of all integer solutions of some explicitly recorded system of linear constraints. For this class, Gomory [26,27,28] designed fast procedures for generating cuts from the optimal simplex basis (and proved that systematic use of these cuts makes the cutting-plane method terminate); cuts generated by these procedures are called *Gomory cuts.*

If an LP relaxation of a TSP instance includes all constraints (4), (5), then a nonempty set of cuts can be found routinely whenever $x^* \notin \mathcal{S}$: on the one hand, if $x^*$ is not an integer vector, then it violates a Gomory cut; on the other hand, if $x^*$ is an integer vector, then it is the incidence vector of the edge-set of a disconnected graph and each connected component of this graph yields a subtour cut. The first computer code for solving the TSP by the cutting-plane method, written by Martin [46], adopts this policy: some of its cuts are subtour inequalities and others are generated by a variation on Gomory's theme described in Martin [45]. In subsequent TSP codes, subtour inequalities became a stock item, but Gomory cuts fell into disuse when a different paradigm for finding cuts took over.

By a *template,* we mean a set of linear inequalities; we say that a cut *matches the template* if it belongs to the set. By the *template paradigm*, we mean the following two-part procedure used in the design of branch-and-cut algorithms:

(i) describe one or more templates of linear inequalities that are satisfied by all the points of $\mathcal{S}$,
(ii) for each template described in part (i), design an efficient *separation algorithm* that, given an $x^*$, attempts to find a cut that matches the template.

The separation algorithms in (ii) may be *exact* in the sense of finding a cut that separates $x^*$ from $\mathcal{S}$ and matches the template whenever one exists and they may be *heuristic* in the sense of sometimes failing to find such a cut even though one exists.

The primordial template of TSP cuts is the set of subtour inequalities; an exact separation algorithm for this template has been pointed out by Hong [39]. Next came the template of "blossom inequalities", introduced by Edmonds [21] in the context of 2-matchings and used in the branch-and-cut TSP computer code written by Hong [39]; then came the more general template of "comb inequalities", first used by Grötschel [30,31] in his solution of a 120-city TSP instance by the cutting-plane method. To describe these templates, let us define, for every vector $x$ with components subscripted by edges of the complete graph on $V$ and for every pair $A, B$ of disjoint subsets of $V$,

$$x(A,B) = \sum(x_e : \ e \cap A \neq \emptyset,\, e \cap B \neq \emptyset).$$

In this notation, subtour inequalities are recorded as $x(Q, V-Q) \geq 2$; a *comb inequality* is any inequality

$$x(H, V-H) + \sum_{i=1}^{k} x(T_i, V-T_i) \geq 3k+1, \tag{6}$$

where

- $H, T_1, T_2, \ldots T_k$ are subsets of $V$,
- $T_1, T_2, \ldots T_k$ are pairwise disjoint,
- $T_i \cap H \neq \emptyset$ and $T_i - H \neq \emptyset$ for all $i = 1, 2, \ldots k$,
- $k$ is odd. To see that every comb inequality is satisfied by all characteristic vectors $x$ of tours through $V$, let $j$ denote the number of sets $T_i$ that satisfy $x(T_i, V - T_i) = 2$. In this notation, $x(H, V - H) \geq j$; furthermore, if $j = k$, then $x(H, V-H) \geq k+1$ as $x(H, V-H)$ is even and $k$ is odd; since each $x(T_i, V-T_i)$ is a positive even integer, we have $\sum_{i=1}^{k} x(T_i, V-T_i) \geq 2j+4(k-j)$ and (6) follows. A *blossom inequality* is a comb inequality with $|T_i| = 2$ for all $i = 1, 2, \ldots k$.

Just like the subtour inequalities, the blossom inequalities, and the comb inequalities, more complicated templates of linear inequalities that are satisfied by all characteristic vectors $x$ of tours through $V$ are often described as *hypergraph inequalities*

$$\sum(\lambda_Q x(Q, V - Q) : Q \in \mathcal{H}) \geq \mu$$

where $\mathcal{H}$ is a collection of subsets of $V$ (also known as a *hypergraph* on $V$) and $\lambda_Q$ $(Q \in \mathcal{H})$ and $\mu$ are integers. Naddef [51] reviews a number of templates of hypergraph inequalities satisfied by all characteristic vectors $x$ of tours through $V$.

TSP codes of Miliotis [49], Land [41], and Fleischmann [23] used the template paradigm as their preferred source of cuts; whenever this source dried up, they provisionally switched to Gomory cuts. Grötschel and Padberg [35,36] and Padberg and Hong [58] advocated a different policy, where the template paradigm is the only source of cuts; furthermore, they argued for drawing the templates exclusively from the set of linear inequalities that induce facets of the *traveling salesman polytope*, meaning the convex hull of $\mathcal{S}$. These policies were adopted in TSP codes of Crowder and Padberg [18], Grötschel and Holland [32], and Padberg and Rinaldi [60,63], which produced the most impressive computational TSP successes of the nineteen eighties. The template paradigm is also the frame of reference for other papers related to TSP codes, such as Carr [10], Christof and Reinelt [11], Clochard and Naddef [13], Cornuéjols, Fonlupt, and Naddef [14], Fleischer and Tardos [22], Grötschel and Pulleyblank [37], Letchford [43], Naddef and Thienel [54,55], Padberg and Rao [59], and Padberg and Rinaldi [61, 62].

We have written a branch-and-cut computer code for solving the TSP. Its initial version was written in 1994; we presented some of its aspects at the 15th International Symposium on Mathematical Programming held at the University of Michigan in 1994, described them in Applegate et al. [1], and eventually distributed the code as Concorde 97.08.27 at the 16th ISMP held at the Ecole polytechnique fédérale de Lausanne in 1997. A later version was written in 1997; we presented some of its aspects at the 16th ISMP, outlined them in Applegate et al. [2], and eventually made the code available on the internet, as Concorde 99.12.15, at Applegate et al. [3]. In the initial 1994 version, cuts were found partly by following the template paradigm (with a couple of our own separation heuristics for comb inequalities thrown in) and partly by a couple of new tech-

niques with the common theme of innovating obsolete cuts. In the later 1997 version, we incorporated a technique that disdains all understanding of the traveling salesman polytope and bashes on regardless of all prescribed templates. This departure from the template paradigm is the subject of the present paper.

## 3 Cuts, Tours, and Shrinking

Let $x^*$ denote the optimal solution of the current LP relaxation that has been returned by the LP solver. In computer codes that search for TSP cuts, it is common practice – Land [41], Padberg and Hong [58], Crowder and Padberg [18], Padberg and Rinaldi [61,62], Grötschel and Holland [32], Naddef and Thienel [54,55] – to first reduce the size of the problem by shrinking pairwise disjoint subsets of the set $V$ of all cities into singletons and then to look for cuts in the shrunk version of $x^*$.

Shrinking is an intuitive concept; to illustrate it on a toy-size example, let us begin with $V = \{0, 1, 2, \ldots, 9\}$. Shrinking each of the sets $\{0, 6, 7\}$, $\{1, 8\}$, and $\{2, 9\}$ into a single vertex – $\{0, 6, 7\}$ into 0, $\{1, 8\}$ into 1, $\{2, 9\}$ into 2 – reduces each vector $x$ with components $x_{01}, x_{02}, \ldots, x_{89}$ (edges are two-point sets, but we prefer writing $x_{ij}$ to writing $x_{\{i,j\}}$) to the vector $\overline{x}$ with components $\overline{x}_{01}, \overline{x}_{02}, \ldots, \overline{x}_{45}$ defined by

$$
\begin{aligned}
\overline{x}_{01} &= x_{01} + x_{08} + x_{16} + x_{17} + x_{68} + x_{78},\\
\overline{x}_{02} &= x_{02} + x_{09} + x_{26} + x_{27} + x_{69} + x_{79},\\
\overline{x}_{03} &= x_{03} + x_{36} + x_{37},\\
\overline{x}_{04} &= x_{04} + x_{46} + x_{47},\\
\overline{x}_{05} &= x_{05} + x_{56} + x_{57},\\
\overline{x}_{12} &= x_{12} + x_{19} + x_{28} + x_{89},\\
\overline{x}_{13} &= x_{13} + x_{38},\\
\overline{x}_{14} &= x_{14} + x_{48},\\
\overline{x}_{15} &= x_{15} + x_{58},\\
\overline{x}_{23} &= x_{23} + x_{39},\\
\overline{x}_{24} &= x_{24} + x_{49},\\
\overline{x}_{25} &= x_{25} + x_{59},\\
\overline{x}_{34} &= x_{34},\\
\overline{x}_{35} &= x_{35},\\
\overline{x}_{45} &= x_{45}.
\end{aligned}
$$

In particular, if $x^*$ is defined by

$$
\begin{gathered}
x^*_{01} = 0.3,\ x^*_{02} = 0.3,\ x^*_{06} = 0.7,\ x^*_{07} = 0.7,\ x^*_{14} = 0.5,\ x^*_{16} = 0.2,\ x^*_{18} = 1.0,\\
x^*_{25} = 0.5,\ x^*_{27} = 0.2,\ x^*_{29} = 1.0,\ x^*_{34} = 0.5,\ x^*_{35} = 0.5,\ x^*_{36} = 0.5,\ x^*_{37} = 0.5,\\
x^*_{45} = 0.5,\ x^*_{48} = 0.5,\ x^*_{59} = 0.5,\ x^*_{67} = 0.6,\ x^*_{89} = 0.5,
\end{gathered}
$$

and $x^*_{ij} = 0$ for all the remaining $i$ and $j$ such that $0 \le i < j \le 9$, then its shrunk version $\overline{x}^*$ is defined by

$$\begin{array}{l} \overline{x}^*_{01} = 0.5,\ \overline{x}^*_{02} = 0.5,\ \overline{x}^*_{12} = 0.5, \\ \overline{x}^*_{03} = 1.0,\ \overline{x}^*_{14} = 1.0,\ \overline{x}^*_{25} = 1.0, \\ \overline{x}^*_{34} = 0.5,\ \overline{x}^*_{35} = 0.5,\ \overline{x}^*_{45} = 0.5, \end{array}$$

and $\overline{x}^*_{ij} = 0$ for all the remaining $i$ and $j$ such that $0 \le i < j \le 5$.

Looking for cuts in the shrunk version $\overline{x}^*$ of $x^*$ does not mean looking for linear inequalities satisfied by all tours through the shrunk version $\overline{V}$ of $V$ and violated by $\overline{x}^*$. In our toy example, the inequality

$$\overline{x}_{01} + \overline{x}_{02} + \overline{x}_{12} + \overline{x}_{03} + \overline{x}_{14} + \overline{x}_{25} \le 4$$

is satisfied by all tours through $\overline{V}$ – which is $\{0, 1, 2, \ldots, 5\}$ – and violated by $\overline{x}^*$; through substitution from the definitions of $\overline{x}_{ij}$, it yields the inequality

$$\begin{aligned} x_{01} + x_{08} + x_{16} + x_{17} + x_{68} + x_{78} \\ + x_{02} + x_{09} + x_{26} + x_{27} + x_{69} + x_{79} \\ + x_{12} + x_{19} + x_{28} + x_{89} \\ + x_{03} + x_{36} + x_{37} \\ + x_{14} + x_{48} \\ + x_{25} + x_{59} \le 4, \end{aligned}$$

which is not satisfied by the tour 0-2-5-9-1-4-8-7-6-3-0.

In this example, shrinking $\{0, 6, 7\}$ into 0, shrinking $\{1, 8\}$ into 1, and shrinking $\{2, 9\}$ into 2 reduces the tour 0-2-5-9-1-4-8-7-6-3-0 to the spanning closed walk 0-2-5-2-1-4-1-0-3-0; it reduces the incidence vector $x$ of the tour to the vector $\overline{x}$ such that

$$\overline{x}_{01} = 1,\ \overline{x}_{02} = 1,\ \overline{x}_{03} = 2,\ \overline{x}_{12} = 1,\ \overline{x}_{14} = 2,\ \overline{x}_{25} = 2,$$

and $\overline{x}_{ij} = 0$ for all the remaining $i$ and $j$ such that $0 \le i < j \le 5$. In general, shrinking $V$ into $\overline{V}$ reduces a tour through $V$ to a spanning closed walk through $\overline{V}$; it reduces the incidence vector $x$ of the tour to a vector $\overline{x}$ such that

- each $\overline{x}_e$ is a nonnegative integer,
- the graph with vertex-set $\overline{V}$ and edge-set $\{e : \overline{x}_e > 0\}$ is connected,
- $\sum(\overline{x}_e : v \in e)$ is even whenever $v \in \overline{V}$;

we will refer to the set of all the vectors $\overline{x}$ with these properties as *tangled tours* through $\overline{V}$. This notion, but not the term, was introduced by Cornuéjols, Fonlupt, and Naddef [14]; they refer to the convex hull of the set of all tangled tours through a prescribed set as the *graphical traveling salesman polytope*. Every inequality $\sum \overline{a}_{ij}\overline{x}_{ij} \le b$ yields, through substitution from the definitions of $\overline{x}_{ij}$, an inequality $\sum a_{ij}x_{ij} \le b$; if the original inequality is satisfied by all tangled tours through $\overline{V}$, then the new inequality is satisfied by all tours through $V$; if

the original inequality is violated by the shrunk version $\overline{x}^*$ of $x^*$, then the new inequality is violated by $x^*$. In our toy example, the blossom inequality

$$\begin{aligned}&\overline{x}(\{0,1,2\},\{3,4,5\})\\ +&\overline{x}(\{0,3\},\{1,2,4,5\})\\ +&\overline{x}(\{1,4\},\{0,2,3,5\})\\ +&\overline{x}(\{2,5\},\{0,1,3,4\}) \geq 10\end{aligned}$$

is satisfied by all tangled tours through $\overline{V}$ and violated by $\overline{x}^*$; it yields, through substitution from the definitions of $\overline{x}_{ij}$, the comb inequality

$$\begin{aligned}&x(\{0,1,2,6,7,8,9\},\{3,4,5\})\\ +&x(\{0,3,6,7\},\{1,2,4,5,8,9\})\\ +&x(\{1,4,8\},\{0,2,3,5,6,7,9\})\\ +&x(\{2,5,9\},\{0,1,3,4,6,7,8\}) \geq 10\,,\end{aligned}$$

which is satisfied by all tours through $V$ and violated by $x^*$.

As this example suggests, the change of variable from $\overline{x}$ to $x$ is particularly easy to implement in hypergraph inequalities: substitution from the definitions of $\overline{x}_{ij}$ converts each linear function $\overline{x}(\overline{Q}, \overline{V}-\overline{Q})$ to the linear function $x(Q, V-Q)$ where $Q$ is the set of all cities that are mapped into $\overline{Q}$ by the function that shrinks $V$ onto $\overline{V}$.

The subject of the present paper is a technique where, rather than first shrinking $x^*$ once and for all onto a $\overline{V}$ that may be large and then attempting to find many cuts in the single $\overline{x}^*$, we shrink $x^*$ many times onto sets $\overline{V}$ that are small (their typical size is at most thirty or so) and we find precisely one cut in each $\overline{x}^*$ that lies outside the graphical traveling salesman polytope on $\overline{V}$: see Algorithm 31.

The forthcoming Sect. 4 takes up about half of the paper and describes the collection of commonplace techniques that we have chosen to implement the body of the **for** loop in Algorithm 31 in Concorde, our computer code (a more detailed description will be presented in Applegate et al. [4]); the very short Sect. 5 describes Concorde's implementation of the control of the **for** loop. Section 6 attempts to give an idea of how useful the inclusion of Algorithm 31 in Concorde turned out to be. Section 7 places Algorithm 31 in a broader context and points out the potential for more general uses of the machinery assembled in Sect. 4.

**Algorithm 31.** *A scheme for collecting TSP cuts*

initialize an empty list $\mathcal{L}$ of cuts;
**for** selected small integers $k$ and
  partitions of $V$ into nonempty sets $V_0, V_1, \ldots, V_k$
**do** $\overline{x}^* =$ the vector obtained from $x^*$ by shrinking each $V_i$ into singleton $i$;
  $\overline{V} = \{0, 1, \ldots, k\}$;

**if** $\overline{x}^*$ lies outside the graphical traveling salesman polytope on $\overline{V}$
**then** find a hypergraph inequality that is
satisfied by all tangled tours through $\overline{V}$ and violated by $\overline{x}^*$;
change its variable from $\overline{x}$ to $x$, and
add the resulting hypergraph inequality to $\mathcal{L}$;
**end**
**end**
return $\mathcal{L}$;

## 4 Processing $\overline{x}^*$

We will illustrate Concorde's implementation of the body of the **for** loop in Algorithm 31 on the example where $\overline{V} = \{0, 1, \ldots, 7\}$ and

$$\begin{aligned}&\overline{x}^*_{01} = 0.7,\ \overline{x}^*_{02} = 0.1,\ \overline{x}^*_{04} = 0.9,\ \overline{x}^*_{07} = 0.9,\ \overline{x}^*_{12} = 0.7,\\ &\overline{x}^*_{15} = 0.6,\ \overline{x}^*_{23} = 0.8,\ \overline{x}^*_{25} = 0.4,\ \overline{x}^*_{34} = 1.0,\ \overline{x}^*_{35} = 0.1,\\ &\overline{x}^*_{36} = 0.1,\ \overline{x}^*_{47} = 0.1,\ \overline{x}^*_{56} = 0.9,\ \overline{x}^*_{67} = 1.0,\end{aligned}$$

and $\overline{x}^*_{ij} = 0$ for all the remaining $i$ and $j$ such that $0 \le i < j \le 7$.

### 4.1 Does $\overline{x}^*$ Lie Outside the Graphical Traveling Salesman Polytope?

**Rephrasing the Question.** In our example, every tangled tour $\overline{x}$ through $\overline{V}$ satisfies the fourteen inequalities

$$\begin{aligned}&\overline{x}_{03} \ge 0,\ \overline{x}_{05} \ge 0,\ \overline{x}_{06} \ge 0,\ \overline{x}_{13} \ge 0,\ \overline{x}_{14} \ge 0,\\ &\overline{x}_{16} \ge 0,\ \overline{x}_{17} \ge 0,\ \overline{x}_{24} \ge 0,\ \overline{x}_{26} \ge 0,\ \overline{x}_{27} \ge 0,\\ &\overline{x}_{37} \ge 0,\ \overline{x}_{45} \ge 0,\ \overline{x}_{46} \ge 0,\ \overline{x}_{57} \ge 0\end{aligned}$$

and the seven inequalities

$$\begin{aligned}&\overline{x}_{01} + \overline{x}_{12} + \overline{x}_{13} + \overline{x}_{14} + \overline{x}_{15} + \overline{x}_{16} + \overline{x}_{17} \ge 2,\\ &\overline{x}_{02} + \overline{x}_{12} + \overline{x}_{23} + \overline{x}_{24} + \overline{x}_{25} + \overline{x}_{26} + \overline{x}_{27} \ge 2,\\ &\overline{x}_{03} + \overline{x}_{13} + \overline{x}_{23} + \overline{x}_{34} + \overline{x}_{35} + \overline{x}_{36} + \overline{x}_{37} \ge 2,\\ &\overline{x}_{04} + \overline{x}_{14} + \overline{x}_{24} + \overline{x}_{34} + \overline{x}_{45} + \overline{x}_{46} + \overline{x}_{47} \ge 2,\\ &\overline{x}_{05} + \overline{x}_{15} + \overline{x}_{25} + \overline{x}_{35} + \overline{x}_{45} + \overline{x}_{56} + \overline{x}_{57} \ge 2,\\ &\overline{x}_{06} + \overline{x}_{16} + \overline{x}_{26} + \overline{x}_{36} + \overline{x}_{46} + \overline{x}_{56} + \overline{x}_{67} \ge 2,\\ &\overline{x}_{07} + \overline{x}_{17} + \overline{x}_{27} + \overline{x}_{37} + \overline{x}_{47} + \overline{x}_{57} + \overline{x}_{67} \ge 2\end{aligned}$$

and the two inequalities

$$\begin{aligned}&\overline{x}_{03} + \overline{x}_{13} + \overline{x}_{23} + \overline{x}_{35} + \overline{x}_{36} + \overline{x}_{37} +\\ &\overline{x}_{04} + \overline{x}_{14} + \overline{x}_{24} + \overline{x}_{45} + \overline{x}_{46} + \overline{x}_{47} \quad \ge 2,\\ &\overline{x}_{06} + \overline{x}_{16} + \overline{x}_{26} + \overline{x}_{36} + \overline{x}_{46} + \overline{x}_{56} +\\ &\overline{x}_{07} + \overline{x}_{17} + \overline{x}_{27} + \overline{x}_{37} + \overline{x}_{47} + \overline{x}_{57} \quad \ge 2;\end{aligned}$$

since $\overline{x}^*$ satisfies each of these twenty-three inequalities as an equation, it is a convex combination of tangled tours through $\overline{V}$ if only if it is a convex combination of those tangled tours through $\overline{V}$ that satisfy each of the twenty-three inequalities as an equation.

In general, every tangled tour $\overline{x}$ through $\overline{V}$ satisfies the inequalities

$$\begin{aligned} \overline{x}_e &\geq 0 \quad \text{for all } e, \\ \sum(\overline{x}_e : u \in e) &\geq 2 \quad \text{for all } u, \\ \overline{x}(e, \overline{V} - e\} &\geq 2 \quad \text{for all } e, \end{aligned}$$

and so $\overline{x}^*$ is a convex combination of tangled tours through $\overline{V}$ if only if it is a convex combination of tangled tours through $\overline{V}$ that satisfy

$$\overline{x}_e = 0 \quad \text{for all } e \text{ such that } \overline{x}^*_e = 0, \tag{7}$$

$$\sum(\overline{x}_e : u \in e) = 2 \quad \text{for all } u \text{ such that } \sum(\overline{x}^*_e : u \in e) = 2, \tag{8}$$

$$\overline{x}(e, \overline{V} - e) = 2 \quad \text{for all } e \text{ such that } \overline{x}^*(e, \overline{V} - e\} = 2. \tag{9}$$

Concorde chooses $V_0, V_1, \ldots, V_k$ so that $x(V_i, V - V_i) = 2$ for all $i = 1, 2, \ldots, k$ (but not necessarily for $i = 0$), and so

$$\sum(\overline{x}^*_e : i \in e) = 2 \text{ for all } i \text{ in } \{1, 2, \ldots, k\}.$$

(Our machinery for processing $\overline{x}^*$ is predicated on this property of $\overline{x}^*$, but could be easily modified to handle all $\overline{x}^*$: see Sect. 7.) In this case, every solution $\overline{x}$ of (8), (9) satisfies

$$\sum(\overline{x}_e : u \in e) = 2 \quad \text{for all } u \text{ in } \{1, 2, \ldots, k\}, \tag{10}$$

$$\overline{x}_e = 1 \quad \text{for all } e \text{ such that } e \subset \{1, 2, \ldots, k\} \text{ and } \overline{x}^*_e = 1, \tag{11}$$

and so

> $\overline{x}^*$ is a convex combination of tangled tours through $\overline{V}$
> if only if it is a convex combination of tangled tours through $\overline{V}$
> that satisfy (7), (10), (11).

We will refer to tangled tours that satisfy (7), (10), (11) as *strongly constrained.*

**Delayed Generation of Strongly Constrained Tangled Tours.** Unfortunately, the strongly constrained tangled tours through $\overline{V}$ may be too numerous to be listed one by one; fortunately, $\overline{x}^*$ can be tested for membership in the convex hull of the set of strongly constrained tangled tours through $\overline{V}$ without listing these tangled tours one by one. The trick, introduced by Ford and Fulkerson [25] and by Jewell [40], is known as *delayed column generation*; we use it in Algorithm 41. This algorithm returns either the message "$\overline{x}^*$ is in the graphical traveling salesman polytope on $\overline{V}$" or a vector $a$ and a scalar $b$ such that the inequality $a^T\overline{x} \leq b$ is satisfied by all strongly constrained tangled tours $\overline{x}$ through $\overline{V}$ and violated by $\overline{x}^*$.

**Algorithm 41.** *Testing the* **if** *condition in Algorithm 31:*

```
if    there is a strongly constrained tangled tour x̄ through V̄
then  make x̄ the only specimen in a collection of
      strongly constrained tangled tours through V̄;
      repeat if    some linear inequality a^T x̄ ≤ b is
                   satisfied by all x̄ in the collection and violated by x̄*
             then find a strongly constrained tangled tour x̄ through V̄
                  that maximizes a^T x̄;
                  if    a^T x̄ ≤ b
                  then  return a and b;
                  else  add x̄ to the collection;
                  end
             else return the message
                  "x̄* is in the graphical traveling salesman polytope on V̄";
             end
      end
else  return 0 and −1;
end
```

In our illustrative example, Algorithm 41 may initialize the collection by the

- strongly constrained tangled tour 0-4-3-2-1-5-6-7-0

and then proceed through the following five iterations:

Iteration 1: Inequality

$$-\overline{x}_{15} \leq -1$$

is satisfied by all $\overline{x}$ in the collection and violated by $\overline{x}^*$. The

- strongly constrained tangled tour 0-1-2-0-4-3-5-6-7-0

maximixes $-\overline{x}_{15}$. We add this tangled tour to the collection.

Iteration 2: Inequality

$$\overline{x}_{25} \leq 0$$

is satisfied by all $\overline{x}$ in the collection and violated by $\overline{x}^*$. The

- strongly constrained tangled tour 0-1-5-2-0-4-3-6-7-0

maximixes $\overline{x}_{25}$. We add this tangled tour to the collection.

Iteration 3: Inequality

$$-\overline{x}_{15} + \overline{x}_{23} + \overline{x}_{25} \leq 0$$

is satisfied by all $\overline{x}$ in the collection and violated by $\overline{x}^*$. The

- strongly constrained tangled tour 0-1-0-4-3-2-5-6-7-0

maximixes $-\overline{x}_{15} + \overline{x}_{23} + \overline{x}_{25}$. We add this tangled tour to the collection.

Iteration 4: Inequality

$$\overline{x}_{47} \leq 0$$

is satisfied by all $\overline{x}$ in the collection and violated by $\overline{x}^*$. The

• strongly constrained tangled tour 0-1-5-6-7-4-3-2-0

maximixes $\overline{x}_{47}$. We add this tangled tour to the collection.

Iteration 5: Inequality

$$\overline{x}_{12} + \overline{x}_{25} + \overline{x}_{47} \leq 1 \tag{12}$$

is satisfied by all $\overline{x}$ in the collection and violated by $\overline{x}^*$. The

• strongly constrained tangled tour 0-4-3-2-1-5-6-7-0

maximixes $\overline{x}_{12} + \overline{x}_{25} + \overline{x}_{47}$. We conclude that (12) is satisfied by all strongly constrained tangled tours and violated by $\overline{x}^*$.

**Implementing Algorithm 41.** Let

$\overline{E}_{1/2}$ denote the set of all the edges $e$ such that
$e \subset \{1, 2, \ldots, k\}$, $\overline{x}^*_e \neq 0$, $\overline{x}^*_e \neq 1$.

The significance of $\overline{E}_{1/2}$ comes from the fact that every strongly constrained tangled tour $\overline{x}$ satisfies

$$\begin{aligned} \overline{x}_{0u} &= 2 - \sum(\overline{x}_e : e \subset \{1, 2, \ldots, k\}, u \in e) \quad \text{for all } u \text{ in } \{1, 2, \ldots, k\}, \\ \overline{x}_e &= 0 \quad \text{for all } e \text{ such that } e \subset \{1, 2, \ldots, k\} \text{ and } \overline{x}^*_e = 0, \\ \overline{x}_e &= 1 \quad \text{for all } e \text{ such that } e \subset \{1, 2, \ldots, k\} \text{ and } \overline{x}^*_e = 1, \end{aligned}$$

and so the condition

some linear inequality $a^T\overline{x} \leq b$ is
satisfied by all $\overline{x}$ in the collection and violated by $\overline{x}^*$

in Algorithm 41 is equivalent to the condition

some linear inequality $a^T\overline{x} \leq b$ with $a_e = 0$ whenever $e \notin \overline{E}_{1/2}$ is
satisfied by all $\overline{x}$ in the collection and violated by $\overline{x}^*$.

To test this condition, Concorde solves a linear programming problem. With

$\psi(\overline{x})$ standing for the restriction of $\overline{x}$
on its components indexed by elements of $\overline{E}_{1/2}$,

with $A$ the matrix whose columns $\psi(\overline{x})$ come from specimens $\overline{x}$ in the collection, and with $e$ standing – as usual – for the vector $[1, 1, \ldots, 1]^T$ whose number of components is determined by the context, this problem in variables $s, \lambda, w$ reads

$$\begin{array}{lll} \text{maximize} & s & \\ \text{subject to} & s\psi(\overline{x}^*) - A\lambda + w & = 0, \\ & -s + e^T\lambda & = 0, \\ & \lambda \geq 0, \quad -e \leq w \leq e. & \end{array} \tag{13}$$

Since its constraints can be satisfied by setting $s = 0$, $\lambda = 0$, $w = 0$, problem (13) either has an optimal solution or else it is unbounded. In the former case, simplex method applied to (13) finds also an optimal solution of its dual,

$$\begin{aligned} \text{minimize} \quad & e^T(u+v) \\ \text{subject to} \quad & -a^T A + be^T \geq 0, \\ & a^T \psi(\overline{x}^*) - b = 1, \\ & a + u - v = 0, \\ & u \geq 0, \quad v \geq 0, \end{aligned} \tag{14}$$

and this optimal solution provides the $a$ and $b$ of Algorithm 41; in fact, it

$$\text{maximizes} \quad \frac{a^T \overline{x}^* - b}{\|a\|_1}$$

subject to the constraints that $a^T \overline{x} \leq b$ for all $\overline{x}$ in the collection and that $a_e = 0$ whenever $e \notin \overline{E}_{1/2}$. In the latter case, (14) is infeasible, and so no linear inequality is satisfied by all $\overline{x}$ in the collection and violated by $\overline{x}^*$.

To find specimens $\overline{x}$ for the collection, we use a function ORACLE that, given an integer vector $c$, returns either a strongly constrained tangled tour $\overline{x}$ through $\overline{V}$ that maximizes $c^T \overline{x}$ or the message "infeasible" indicating that no tangled tour through $\overline{V}$ is strongly constrained. Concorde implements ORACLE as two algorithms in tandem: if a primitive branch-and-bound algorithm fails to solve the instance within a prescribed number of recursive calls of itself, then we switch to a more sophisticated branch-and-cut algorithm. To reconcile

- the integer arithmetic of ORACLE, which uses $a$ and $b$,

with

- the floating-point arithmetic of the simplex method, which finds $a$ and $b$,

Concorde uses the continued fraction method (see, for instance, Schrijver [66]); since ORACLE uses integer arithmetic, the cut $a^T \overline{x} \leq b$ that separates $\overline{x}^*$ from all strongly constrained tangled tours through $\overline{V}$ has an integer $a$ and an integer right-hand side $b$.

### 4.2 Separating $\overline{x}^*$ from the Graphical Traveling Salesman Polytope: The Three Phases

If $\overline{x}^*$ lies outside the graphical traveling salesman polytope, then Algorithm 41 returns a linear inequality that separates $\overline{x}^*$ from all strongly constrained tangled tours. To convert this inequality to a cut that separates $\overline{x}^*$ from all tangled tours, we proceed in in three phases:

in PHASE 1, we find a linear inequality that
separates $\overline{x}^*$ from all *moderately constrained* tangled tours and
induces a facet of the convex hull of these tangled tours,
in PHASE 2, we find a linear inequality that
separates $\overline{x}^*$ from all *weakly constrained* tangled tours and
induces a facet of the convex hull of these tangled tours,

in PHASE 3, we find a linear inequality that
separates $\overline{x}^*$ from all tangled tours and
induces a facet of the convex hull of weakly constrained tangled tours.

(In PHASE 3, we could easily find a linear inequality that separates $\overline{x}^*$ from all tangled tours and induces a facet of the convex hull of all tangled tours; however, such an inequality might not be acceptable to Concorde as a constraint of an LP relaxation of the TSP instance. We will discuss this point in Sect. 4.5.)

The intermediate classes of moderately constrained tangled tours and weakly constrained tangled tours are defined by reference to the sets of constraints

(a) $\sum(\overline{x}_e : u \in e) = 2$ for all $u$ in $\{1, 2, \ldots, k\}$;
(b) $\overline{x}_e = 0$ for all $e$ such that $e \subset \{1, 2, \ldots, k\}$ and $\overline{x}^*_e = 0$,
$\overline{x}_e = 1$ for all $e$ such that $e \subset \{1, 2, \ldots, k\}$ and $\overline{x}^*_e = 1$;
(c) $\overline{x}_{0u} = 0$ for all $u$ in $\{1, 2, \ldots, k\}$ such that $\overline{x}^*_{0u} = 0$.

Specifically,

strongly constrained tangled tours are those that satisfy (a), (b), (c);
moderately constrained tangled tours are those that satisfy (a), (b);
weakly constrained tangled tours are those that satisfy (a).

The principal tools used in PHASE 1 are a constructive proof of a a classic result of Minkowski (Theorem 1) and the Dinkelbach method of fractional programming; PHASE 2 is sequential lifting; PHASE 3 is nearly effortless. In PHASE 1 and PHASE 2, we use a function ORACLE that,

given integer vectors $c$, $\ell$, $u$ and a threshold $t$ (either an integer or $-\infty$),
returns either
a weakly constrained tangled tour $\overline{x}$ that maximizes $c^T\overline{x}$
subject to $\ell \leq \overline{x} \leq u$, $c^T\overline{x} > t$
or
the message "infeasible" indicating that
no weakly constrained tangled tour $\overline{x}$ satisfies
$\ell \leq \overline{x} \leq u$, $c^T\overline{x} > t$.

This is the same function that is used, with a fixed $\ell$ and a fixed $u$, to find items $\overline{x}$ for the collection in Algorithm 41.

### 4.3 PHASE 1: **From Strongly Constrained to Moderately Constrained Tangled Tours**

**Separating $\overline{x}^*$ from Moderately Constrained Tangled Tours.** It is easy to convert any linear inequality that separates $\overline{x}^*$ from all strongly constrained tangled tours to a linear inequality that separates $\overline{x}^*$ from all moderately constrained tangled tours. In our example, inequality

$$\overline{x}_{12} + \overline{x}_{25} + \overline{x}_{47} \leq 1$$

is satisfied by all strongly constrained tangled tours $\overline{x}$ and violated by $\overline{x}^*$; if $\overline{x}$ is a moderately constrained tangled tour, then $\psi(\overline{x})$ is a zero-one vector, and so

$$\overline{x}_{12} + \overline{x}_{25} + \overline{x}_{47} \leq 3;$$

a moderately constrained tangled tour $\overline{x}$ is strongly constrained if and only if

$$\overline{x}_{03} = 0,\ \overline{x}_{05} = 0,\ \overline{x}_{06} = 0.$$

It follows that the inequality

$$\overline{x}_{12} + \overline{x}_{25} + \overline{x}_{47} - 2(\overline{x}_{03} + \overline{x}_{05} + \overline{x}_{06}) \leq 1$$

is satisfied by all moderately constrained tangled tours $\overline{x}$ and violated by $\overline{x}^*$. In general, a moderately constrained tangled tour $\overline{x}$ is strongly constrained if and only if

$$\overline{x}_{0u} = 0 \text{ for all } u \text{ in } \{1, 2, \ldots, k\} \text{ such that } \overline{x}^*_{0u} = 0;$$

if

$$p^T \psi(\overline{x}) \leq r$$

is satisfied by all strongly constrained tangled tours $\overline{x}$ and violated by $\overline{x}^*$, then

$$p^T \psi(\overline{x}) - (||p||_1 - r) \sum(\overline{x}_{0u} : \overline{x}^*_{0u} = 0) \leq r \tag{15}$$

is satisfied by all moderately constrained tangled tours $\overline{x}$ and violated by $\overline{x}^*$ (to see that $||p||_1 - r$ is always positive, note that $r < p^T \psi(\overline{x}^*) \leq ||p||_1$); the bulk of Phase 1 is taken up by transforming this cut into a cut that induces a facet of the convex hull of all moderately constrained tangled tours.

**Dimension of the Set of Moderately Constrained Tangled Tours.** In Sect. 4.1, we defined

$$\overline{E}_{1/2} = \{e :\ e \subset \{1, 2, \ldots, k\},\ \overline{x}^*_e \neq 0,\ \overline{x}^*_e \neq 1\}.$$

In fact, we may assume that

$$\overline{E}_{1/2} = \{e : e \subset \{1, 2, \ldots, k\},\ 0 < \overline{x}^*_e < 1\} :$$

otherwise $\overline{x}^*_e > 1$ for some $e$ such that $e \subset \{1, 2, \ldots, k\}$, and so $\overline{x}^*$ is separated from all tangled tours by the subtour inequality $\overline{x}(e, \overline{V} - e) \geq 2$. (As noted by Cornuéjols, Fonlupt, and Naddef [14], subtour inequalities induce facets of the graphical traveling salesman polytope.) In addition,

with $\psi(\overline{x})$ standing for the restriction of $\overline{x}$ on $\overline{E}_{1/2}$ just as in Sect. 4.1,

we may assume that each of the vectors

$$\begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \\ \vdots \\ 0 \end{bmatrix}, \ldots, \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix}$$

with components indexed by elements of $\overline{E}_{1/2}$ equals $\psi(\overline{x})$ for some moderately constrained tangled tour $\overline{x}$ (else $\overline{x}^*$ is separated from $\mathcal{T}$ by another readily available subtour inequality), and so

$$\{\psi(\overline{x}) : \overline{x} \text{ is a moderately constrained tangled tour}\} \text{ has full dimension.}$$

Since every moderately constrained tangled tour $\overline{x}$ satisfies

$$\begin{aligned} \overline{x}_{0u} &= 2 - \sum(\overline{x}_e : e \subset \{1,2,\ldots,k\}, u \in e) \quad \text{for all } u \text{ in } \{1,2,\ldots,k\}, \\ \overline{x}_e &= 0 \quad \text{for all } e \text{ such that } e \subset \{1,2,\ldots,k\} \text{ and } \overline{x}^*_e = 0, \\ \overline{x}_e &= 1 \quad \text{for all } e \text{ such that } e \subset \{1,2,\ldots,k\} \text{ and } \overline{x}^*_e = 1, \end{aligned}$$

we conclude that

$$\text{the set of moderately constrained tangled tours has dimension } |\overline{E}_{1/2}|.$$

**From a Cut to a Facet-Inducing Cut: An Overview.** The optimal basis of problem (13) provides an affinely independent set $\mathcal{I}$ of strongly constrained tangled tours $\overline{x}$ that satisfy (15) with the sign of equality. In our illustrative example, $\mathcal{I}$ consists of

- the strongly constrained tangled tour 0-4-3-2-1-5-6-7-0,
- the strongly constrained tangled tour 0-1-2-0-4-3-5-6-7-0,
- the strongly constrained tangled tour 0-1-5-2-0-4-3-6-7-0,
- the strongly constrained tangled tour 0-1-0-4-3-2-5-6-7-0,
- the strongly constrained tangled tour 0-1-5-6-7-4-3-2-0.

In general, it may turn out that $\mathcal{I} = \emptyset$; this happens if and only if no tangled tour through $\overline{V}$ is strongly constrained, in which case (15) reads

$$-\sum(\overline{x}_{0u} : \overline{x}^*_{0u} = 0) \leq -1.$$

Writing $a^T\overline{x} \leq b$ for (15), we have an integer vector $a$, an integer $b$, and a (possibly empty) set $\mathcal{I}$ such that

(Inv1) all moderately constrained tangled tours $\overline{x}$ have $a^T\overline{x} \leq b$,
(Inv2) $\mathcal{I}$ is an affinely independent set
of moderately constrained tangled tours,
(Inv3) $a^T\overline{x} = b$ whenever $\overline{x} \in \mathcal{I}$,
(Inv4) $a^T\overline{x}^* > b$.

Concorde maintains these four invariants while adding new elements to $\mathcal{I}$ and adjusting $a$ and $b$ if necessary: when $|\mathcal{I}|$ reaches $|\overline{E}_{1/2}|$, the current cut $a^T\overline{x} \leq b$ induces a facet of the convex hull of all moderately constrained tangled tours. An outline of this process is provided in Algorithm 42.

**Algorithm 42.** *From a cut to a facet-inducing cut in Phase 1*:

**while** $|\mathcal{I}| < |\overline{E}_{1/2}|$
**do** find an integer vector $v$, an integer $w$, and
a moderately constrained tangled tour $\overline{x}^0$ such that
(A1) $v^T\overline{x} = w$ whenever $\overline{x} \in \mathcal{I}$,
(A2) some moderately constrained tangled tour $\overline{x}$ has $v^T\overline{x} \neq w$,
and either
(A3.1) $v^T\overline{x}^* \geq w$ and $v^T\overline{x} \geq w$ for all moderately constrained tangled tours
or else
(A3.2) $a^T\overline{x}^0 < b$ and $v^T\overline{x}^0 = w$;
find an integer vector $a'$, an integer $b'$, and
a moderately constrained tangled tour $\overline{x}'$ such that
(B1) all moderately constrained tangled tours $\overline{x}$ have ${a'}^T\overline{x} \leq b'$,
(B2) equation ${a'}^T\overline{x} = b'$ is a linear combination of $a^T\overline{x} = b$ and $v^T\overline{x} = w$,
(B3) ${a'}^T\overline{x}' = b'$ and $(a^T\overline{x}', v^T\overline{x}') \neq (b, w)$,
(B4) ${a'}^T\overline{x}^* > b'$;
$a = a'$, $b = b'$, $\mathcal{I} = \mathcal{I} \cup \{\overline{x}'\}$;
**end**
return $a$ and $b$;

In our illustrative example, the initial $a^T x \leq b$ reads

$$-2\overline{x}_{03} - 2\overline{x}_{05} - 2\overline{x}_{06} + \overline{x}_{12} + \overline{x}_{25} + \overline{x}_{47} \leq 1$$

and Algorithm 42 may proceed as follows.

Iteration 1: $v^T\overline{x} = \overline{x}_{03}$, $w = 0$, and $\overline{x}^0$ is an arbitrary moderately constrained tangled tour. We leave $a^T\overline{x} \leq b$ unchanged and we add to $\mathcal{I}$

- the moderately constrained tangled tour 0-1-2-5-6-7-4-3-0.

Iteration 2: $v^T\overline{x} = \overline{x}_{05}$, $w = 0$, and $\overline{x}^0$ is an arbitrary moderately constrained tangled tour. We replace $a^T\overline{x} \leq b$ by

$$-2\overline{x}_{03} - \overline{x}_{05} - 2\overline{x}_{06} + \overline{x}_{12} + \overline{x}_{25} + \overline{x}_{47} \leq 1 \tag{16}$$

and we add to $\mathcal{I}$

- the moderately constrained tangled tour 0-1-2-5-0-4-3-6-7-0.

Iteration 3: $v^T\overline{x} = \overline{x}_{06}$, $w = 0$, and $\overline{x}^0$ is an arbitrary moderately constrained tangled tour. We leave $a^T\overline{x} \leq b$ unchanged and we add to $\mathcal{I}$

- the moderately constrained tangled tour 0-1-2-5-3-4-7-6-0.

Now $|\mathcal{I}|$ has reached $|\overline{E}_{1/2}|$, and so (16) induces a facet of the convex hull of moderately constrained tangled tours.

**Finding $v$, $w$, and $\overline{x}^0$ in Algorithm 42.** In early iterations of the **while** loop in Algorithm 42, Concorde draws $v$ and $w$ from a catalog of inequalities $v^T\overline{x} \geq w$ that satisfy (A3.1) and (A2); any of these inequalities that happens to satisfy (A1) can provide the $v$ and the $w$ for use, with an arbitrary moderately constrained tangled tour $\overline{x}^0$, in the next iteration of the **while** loop. The catalog consists of

$$\begin{array}{lll} \text{all inequalities} & \overline{x}_e \geq 0 & \text{such that } e \in \overline{E}_{1/2}, \\ \text{all inequalities} & -\overline{x}_e \geq -1 & \text{such that } e \in \overline{E}_{1/2}, \\ \text{all inequalities} & \overline{x}_{0u} \geq 0 & \text{such that } u \in \{1, 2, \ldots, k\}; \end{array}$$

for all $u$ in $\{1, 2, \ldots, k\}$, Concorde's way of shrinking $x^*$ into $\overline{x}^*$ – whose description is deferred to Sect. 5 – guarantees the existence of a moderately constrained tangled tour $\overline{x}$ such that $\overline{x}_{0u} > 0$.

If no inequality $v^T\overline{x} \geq w$ in the catalog satisfies (A1) and yet $|\mathcal{I}| < |\overline{E}_{1/2}|$, then Concorde finds $v$ as a nonzero solution of the system

$$v^T\overline{x} = 0 \text{ for all } \overline{x} \text{ in } \mathcal{I}, \tag{17}$$

$$v_e = 0 \text{ for all } e \text{ outside } \overline{E}_{1/2}, \tag{18}$$

it sets $w = 0$, and it lets $\overline{x}^0$ be the moderately constrained tangled tour such that $\overline{x}^0_e = 0$ for all $e$ in $\overline{E}_{1/2}$.

For each $e$ in $\overline{E}_{1/2}$, let $\overline{x}^e$ denote the moderately constrained tangled tour such that $\overline{x}^e_e = 1$ and $\overline{x}^e_f = 0$ for all other $f$ in $\overline{E}_{1/2}$. Property (18) guarantees that $v^T\overline{x}^e = v_e$ for all $e$ in $\overline{E}_{1/2}$; since $v$ is a nonzero vector with property (18), at least one $e$ in $\overline{E}_{1/2}$ has $v_e \neq 0$; it follows that (A2) holds.

To see that $a^T\overline{x}^0 < b$, observe that

$$\sum(\overline{x}^*_e\overline{x}^e : e \in \overline{E}_{1/2}) + \left(1 - \sum(\overline{x}^*_e : e \in \overline{E}_{1/2})\right)\overline{x}^0 = \overline{x}^*,$$

and so

$$\begin{aligned} \left(1 - \sum(\overline{x}^*_e : e \in \overline{E}_{1/2})\right) a^T\overline{x}^0 &= a^T\overline{x}^* - \sum(\overline{x}^*_e a^T\overline{x}^e : e \in \overline{E}_{1/2}) \\ &> b\left(1 - \sum(\overline{x}^*_e : e \in \overline{E}_{1/2})\right). \end{aligned}$$

To see that $v^T\overline{x}^0 = 0$, note that $v_e\overline{x}^0_e = 0$ for all $e$.

**Finding $a'$, $b'$, and $\overline{x}'$ in Algorithm 42.** To add new elements to $\mathcal{I}$ and to adjust $a$ and $b$ if necessary, Concorde's implementation of Algorithm 42 uses a function TILT, which, given integer vectors $a$, $v$, integers $b$, $w$, and a moderately constrained tangled tour $\overline{x}^0$ with the property

> if all moderately constrained tangled tours $\overline{x}$ have $v^T\overline{x} \leq w$,
> then $a^T\overline{x}^0 < b$ and $v^T\overline{x}^0 = w$,

returns a nonzero integer vector $a'$, an integer $b'$, and a moderately constrained tangled tour $\overline{x}'$ with properties (B1),

(B2$^+$) inequality ${a'}^T\overline{x} \le b'$ is a nonnegative linear combination of $a^T\overline{x} \le b$ and $v^T\overline{x} \le w$,

and (B3).

In the iterations of the **while** loop in Algorithm 42 where $v$ and $w$ are drawn from the catalog, Concorde calls TILT $(a, b, v, w, \overline{x}^0)$ for $(a', b', \overline{x}')$. To see that ${a'}^T\overline{x}^* > b'$, note that there are nonnegative $\lambda$ and $\mu$ such that $a' = \lambda a + \mu v$, $b' = \lambda b + \mu w$; that $v^T\overline{x}^* \ge w$; and that $\lambda > 0$ since ${a'}^T\overline{x} \le b'$ is satisfied by all moderately constrained tangled tours but $v^T\overline{x} \le w$ is not.

In the iterations of the **while** loop in Algorithm 42 where $w = 0$ and $v$ is computed by solving the system (17), (18), Concorde computes

$$\begin{aligned}(a^+, b^+, \overline{x}^+) &= \text{TILT } (a, b, v, 0, \overline{x}^0),\\ (a^-, b^-, \overline{x}^-) &= \text{TILT } (a, b, -v, 0, \overline{x}^0)\end{aligned}$$

and then it sets

$$\begin{aligned}&a' = a^+,\ b' = b^+,\ \overline{x}' = \overline{x}^+ && \text{if } a^{+T}\overline{x}^* - b^+ \ge a^{-T}\overline{x}^* - b^-,\\ &a' = a^-,\ b' = b^-,\ \overline{x}' = \overline{x}^- && \text{otherwise.}\end{aligned}$$

To show that ${a'}^T\overline{x}^* > b'$, we are going to prove that

inequality $a^T\overline{x} \le b$ is a nonnegative linear combination of $a^{+T}\overline{x} \le b^+$ and $a^{-T}\overline{x} \le b^-$.

There are nonnegative numbers $\lambda^+$ and $\mu^+$ such that

$$a^+ = \lambda^+ a + \mu^+ v, \quad b^+ = \lambda^+ b$$

and there are nonnegative numbers $\lambda^-$ and $\mu^-$ such that

$$a^- = \lambda^- a - \mu^- v, \quad b^- = \lambda^- b.$$

Since $a^{+T}\overline{x} \le b^+$ and $a^{-T}\overline{x} \le b^-$ for all moderately constrained tangled tours $\overline{x}$ and since some moderately constrained tangled tour $\overline{x}$ has $v^T\overline{x} \ne 0$,

at least one of $\lambda^+, \lambda^-$ is positive.

If $\mu^+ = 0$, then $\lambda^+ > 0$, and so $a^T\overline{x} \le b$ is a positive multiple of $a^{+T}\overline{x} \le b^+$; if $\mu^- = 0$, then $\lambda^- > 0$, and so $a^T\overline{x} \le b$ is a positive multiple of $a^{-T}\overline{x} \le b^-$; if $\mu^+ > 0$ and $\mu^- > 0$, then $\lambda^+\mu^- + \lambda^-\mu^+ > 0$ and

$$\begin{aligned}a &= \frac{\mu^+}{\lambda^+\mu^- + \lambda^-\mu^+}\, a^- + \frac{\mu^-}{\lambda^+\mu^- + \lambda^-\mu^+}\, a^+,\\ b &= \frac{\mu^+}{\lambda^+\mu^- + \lambda^-\mu^+}\, b^- + \frac{\mu^-}{\lambda^+\mu^- + \lambda^-\mu^+}\, b^+.\end{aligned}$$

**Implementation of** TILT. TILT could be implemented by the *Dinkelbach method* of fractional programming (see, for instance, Sect. 5.6 of Craven [15] or Sect. 4.5 of Stancu-Minasian [67]) as in Algorithm 43.

**Algorithm 43.** TILT $(a, b, v, w, \overline{x}^0)$:

$\overline{x}$ = moderately constrained tangled tour that maximizes $v^T\overline{x}$;
$\lambda = v^T\overline{x} - w$, $\mu = b - a^T\overline{x}$;
**if** $\lambda = 0$
**then** return $(v, w, \overline{x}^0)$;
**else** **if** $\mu = 0$
**then** return $(a, b, \overline{x})$;
**else** return TILT $(a,\ b,\ \lambda a + \mu v,\ \lambda b + \mu w,\ \overline{x})$;
**end**
**end**

For illustration, consider ITERATION 2 in our example. Here,

$$a_{03} = -2,\ a_{05} = -2,\ a_{06} = -2,\ a_{12} = 1,\ a_{25} = 1,\ a_{47} = 1,$$

and $a_{ij} = 0$ for all other $i$ and $j$ such that $0 \le i < j \le 7$; we have $b = 1$;

$$v_{05} = 1$$

and $v_{ij} = 0$ for all other $i$ and $j$ such that $0 \le i < j \le 7$; we have $w = 0$; vector $x^0$ may be any moderately constrained tangled tour.

In the nested recursive calls of TILT $(a, b, v, w, \overline{x})$, values of $a$ and $b$ do not change, but values of $v$ and $w$ do; in our illustration, we may specify each new $v$ as

$$[\,v_{03},\ v_{05},\ v_{06},\ v_{12},\ v_{25},\ v_{47}\,]^T$$

since $v_{ij} = 0$ for all other $i$ and $j$ such that $0 \le i < j \le 7$. In this notation, a record of the computations may go as follows.

TILT $(a, b, [0, 1, 0, 0, 0, 0]^T, 0, \overline{x}^0)$:
the $\overline{x}$ returned by ORACLE is 0-1-0-2-0-3-4-0-5-0-6-7-0;
$\lambda = 2$, $\mu = 9$;
TILT $(a, b, [-4, 5, -4, 2, 2, 2]^T, 2, \overline{x})$:
the $\overline{x}$ returned by ORACLE is 0-1-2-0-4-3-6-7-0-5-0;
$\lambda = 10$, $\mu = 4$;
TILT $(a, b, [-36, 0, -36, 18, 18, 18]^T, 18, \overline{x})$:
the $\overline{x}$ returned by ORACLE is 0-1-2-5-0-4-3-6-7-0;
$\lambda = 18$, $\mu = 1$;
TILT $(a, b, [-72, -36, -72, 36, 36, 36]^T, 36, \overline{x})$:
the $\overline{x}$ returned by ORACLE is 0-1-2-5-0-4-3-6-7-0;
$\lambda = 0$, $\mu = 1$;
return $([-72, -36, -72, 36, 36, 36]^T,\ 36,$ 0-1-2-5-0-4-3-6-7-0);
return $([-72, -36, -72, 36, 36, 36]^T,\ 36,$ 0-1-2-5-0-4-3-6-7-0);

return $([-72,-36,-72,36,36,36]^T,\ 36,\ 0\text{-}1\text{-}2\text{-}5\text{-}0\text{-}4\text{-}3\text{-}6\text{-}7\text{-}0)$;
return $([-72,-36,-72,36,36,36]^T,\ 36,\ 0\text{-}1\text{-}2\text{-}5\text{-}0\text{-}4\text{-}3\text{-}6\text{-}7\text{-}0)$;

Concorde implements TILT as a modified version of Algorithm 43: before each recursive call of TILT $(a,b,v,w,\overline{x})$, it divides $w$ and all the components of $v$ by their greatest common divisor. This policy makes

TILT $(a,b,[0,1,0,0,0,0]^T,0,\overline{x}^0)$
TILT $(a,b,[-4,5,-4,2,2,2]^T,2,\overline{x})$
TILT $(a,b,[-2,0,-2,1,1,1]^T,1,\overline{x})$
TILT $(a,b,[-2,-1,-2,1,1,1]^T,1,\overline{x})$

the four invocations of TILT in our example and it makes

$$([-2,-1,-2,1,1,1]^T,\ 1,\ 0\text{-}1\text{-}2\text{-}5\text{-}0\text{-}4\text{-}3\text{-}6\text{-}7\text{-}0)$$

their common return value.

### 4.4 PHASE 2: **From Moderately Constrained to Weakly Constrained Tangled Tours**

We will write

$$E_0 = \{e:\ e \subset \{1,2,\ldots,k\}, \overline{x}^*_e = 0\},$$
$$E_1 = \{e:\ e \subset \{1,2,\ldots,k\}, \overline{x}^*_e = 1\};$$

in this notation, a weakly constrained tangled tour $\overline{x}$ is moderately constrained if and only if

$$\overline{x}_e = 0 \text{ whenever } e \in E_0 \text{ and } \overline{x}_e = 1 \text{ whenever } e \in E_1.$$

The linear inequality $a^T\overline{x} \le b$ constructed in PHASE 1 separates $\overline{x}^*$ from all moderately constrained tangled tours and induces a a facet of their convex hull; in PHASE 2, we find integers $\Delta_e$ $(e \in \overline{E}_0 \cup \overline{E}_1)$ such that the inequality

$$a^T\overline{x} + \sum(\Delta_e\overline{x}_e : e \in \overline{E}_0 \cup \overline{E}_1) \le b + \sum(\Delta_e : e \in \overline{E}_1)$$

separates $\overline{x}^*$ from all weakly constrained tangled tours and induces a a facet of their convex hull. A way of computing the $\Delta_e$ one by one originated in the work of Gomory (1969) and was elaborated by Balas (1975), Hammer, Johnson, and Peled (1975), Padberg (1973,1975), Wolsey (1975a, 1975b), and others; it is known as *sequential lifting*; its application in our context is described in Algorithm 44. Both **while** loops in this algorithm maintain the invariant

$$\left.\begin{array}{l} a^T\overline{x} \le b \text{ induces a facet of the convex hull of} \\ \text{all weakly constrained tangled tours } \overline{x} \text{ such that} \\ \overline{x}_e = 0 \text{ whenever } e \in F_0 \text{ and } \overline{x}_e = 1 \text{ whenever } e \in F_1. \end{array}\right\} \quad (19)$$

**Algorithm 44.** *Sequential lifting*

$F_0 = \overline{E}_0$ , $F_1 = \overline{E}_1$ ;
**while** $F_1 \neq \emptyset$
**do** $f$ = an edge in $F_1$;
find a weakly constrained tangled tour $\overline{x}^{\max}$ that
maximizes $a^T\overline{x}$ subject to
$\overline{x}_e = 0$ whenever $e \in F_0 \cup \{f\}$,
$\overline{x}_e = 1$ whenever $e \in F_1 - \{f\}$;
replace $a^T\overline{x} \leq b$ by $a^T\overline{x} + (a^T\overline{x}^{\max} - b)\overline{x}_f \leq a^T\overline{x}^{\max}$;
delete $f$ from $F_1$;
**end**
**while** $F_0 \neq \emptyset$
**do** $f$ = an edge in $F_0$;
find a weakly constrained tangled tour $\overline{x}^{\max}$ that
maximizes $a^T\overline{x}$ subject to
$\overline{x}_e = 0$ whenever $e \in F_0 - \{f\}$,
$\overline{x}_f = 1$;
replace $a^T\overline{x} \leq b$ by $a^T\overline{x} + (b - a^T\overline{x}^{\max})\overline{x}_f \leq b$;
delete $f$ from $F_0$;
**end**

Concorde enters Phase 2 with an inequality $a^T\overline{x} \leq b$ such that

(i) $a^T\overline{x} \leq b$ induces a facet of the convex hull of all moderately constrained tangled tours,
(ii) $a^T\overline{x}^* > b$,
(iii) $a_e = 0$ for all $e$ outside $\overline{E}_{1/2}$.

An arbitrary inequality $a^T\overline{x} \leq b$ with properties (i), (ii) can be made to satisfy (iii) as well by first substituting $2 - \sum(\overline{x}_e : 0 \notin e, u \in e)$ for all $\overline{x}_{0u}$ and then substituting 0 for all $\overline{x}_e$ such that $e \in E_0$ and substituting 1 for all $\overline{x}_e$ such that $e \in E_1$. In our illustrative example, these substitutions convert inequality

$$-2\overline{x}_{03} - \overline{x}_{05} - 2\overline{x}_{06} + \overline{x}_{12} + \overline{x}_{25} + \overline{x}_{47} \leq 1$$

with properties (i), (ii) to inequality

$$\overline{x}_{12} + \overline{x}_{15} + 2\overline{x}_{23} + 2\overline{x}_{25} + 3\overline{x}_{35} + 4\overline{x}_{36} + \overline{x}_{47} + 3\overline{x}_{56} \leq 7$$

with properties (i), (ii), and (iii).

Concorde implements Phase 2 as a streamlined version of Algorithm 44; to describe this version, we will write

$$\overline{V}^{\text{int}} = \{1, 2, \ldots, k\} \quad \text{and} \quad \overline{E}^{\text{int}} = \{e : e \subset \overline{V}^{\text{int}}, |e| = 2\}.$$

Since every weakly constrained tangled tour $\overline{x}$ satisfies

$$\overline{x}_{0u} = 2 - \sum(\overline{x}_e : e \in \overline{E}^{\text{int}}, u \in e) \text{ for all } u \text{ in } \overline{V}^{\text{int}},$$

it is determined by its restriction on $\overline{E}^{\text{int}}$. Restrictions of weakly constrained tangled tours on $\overline{E}^{\text{int}}$ are precisely the incidence vectors of the edge-sets of *path systems* – meaning graphs whose connected components are paths – with vertex-set $\overline{V}^{\text{int}}$. The set of all path systems with vertex-set $\overline{V}^{\text{int}}$ is *monotone* in the sense that the removal of an edge from a path system yields another path system.

Monotonicity of the set of all path systems with vertex-set $\overline{V}^{\text{int}}$ implies that, for all choices of subsets $R_0, R_1$ of $\overline{E}^{\text{int}}$ and for all choices of objective functions $a^T x$ such that

$$a_e = 0 \text{ for all } e \text{ outside } \overline{E}^{\text{int}}$$

(this property of $a$ is maintained by Algorithm 44), the problem of finding a weakly constrained tangled tour that

$$\begin{array}{l} \text{maximizes } a^T\overline{x} \text{ subject to} \\ \overline{x}_e = 0 \text{ whenever } e \in R_0, \\ \overline{x}_e = 1 \text{ whenever } e \in R_1 \end{array}$$

either has no feasible solution at all or else has an optimal solution such that

$$\overline{x}_e = 0 \text{ whenever } e \in \overline{E}^{\text{int}} - (R_0 \cup R_1) \text{ and } a_e = 0.$$

Concorde exploits this observation in a couple of ways: it makes the job of Oracle easier by adding constraint

$$\overline{x}_e = 0 \text{ whenever } e \in \overline{E}^{\text{int}} - (F_0 \cup F_1) \text{ and } a_e = 0$$

to the problem of finding $\overline{x}^{\max}$ and it skips certain calls of Oracle altogether.

The second of these two tricks begins with the set $\mathcal{I}$ of $|\overline{E}_{1/2}|$ moderately constrained tangled tours produced in Phase 1: for each element $\overline{x}'$ of $\mathcal{I}$, Concorde deletes from $F_0$ all the edges $uv$ such that $u$ and $v$ are endpoints of distinct paths in the path system defined by $\overline{x}'$. To see that each of these deletions preserves invariant (19), consider the weakly constrained tangled tour $\overline{x}''$ that

$$\begin{array}{l} \text{maximizes } a^T\overline{x} \text{ subject to} \\ \overline{x}_e = 0 \text{ whenever } e \in F_0 - \{uv\}, \\ \overline{x}_e = 1 \text{ whenever } e \in F_1 \cup \{uv\}. \end{array}$$

On the one hand, the path system defined by $\overline{x}'$ with edge $uv$ added shows that $a^T\overline{x}'' \geq b$; on the other hand, the path system defined by $\overline{x}''$ with edge $uv$ deleted shows that $a^T\overline{x}'' \leq b$; we conclude that $a^T\overline{x}'' = b$, and so the deletion of $uv$ from $F_0$ preserves invariant (19).

This trick applies not only to the elements of $\mathcal{I}$, but also to each weakly tangled tour found by Oracle in Phase 2: as soon as it finds a new $x^{\max}$, Concorde deletes from $F_0$ all the edges $uv$ such that $u$ and $v$ are endpoints of distinct paths in the path system defined by $x^{\max}$.

In our illustrative example, we begin with

$$F_0 = \{13, 14, 16, 17, 24, 26, 27, 37, 45, 46, 57\}, \quad F_1 = \{34, 67\}$$

and then we examine the eight path systems defined by $\mathcal{I}$:

- the path system 4-3-2-1-5-6-7 is a single path,
- the path system with components 1-2 and 4-3-5-6-7 eliminates edges 14, 17, 24, 27 from $F_0$,
- the path system with components 1-5-2 and 4-3-6-7 eliminates edges 14, 17, 24, 27 from $F_0$,
- the path system with components 1 and 4-3-2-5-6-7 eliminates edges 14, 17 from $F_0$,
- the path system 1-5-6-7-4-3-2 is a single path,
- the path system 1-2-5-6-7-4-3 is a single path,
- the path system with components 1-2-5 and 4-3-6-7 eliminates edges 14, 17, 45, 57 from $F_0$,
- the path system 1-2-5-3-4-7-6 is a single path.

The initial $a^T\overline{x} \leq b$ reads

$$\overline{x}_{12} + \overline{x}_{15} + 2\overline{x}_{23} + 2\overline{x}_{25} + 3\overline{x}_{35} + 4\overline{x}_{36} + \overline{x}_{47} + 3\overline{x}_{56} \leq 7. \tag{20}$$

The first **while** loop of Algorithm 44 may go as follows.

Iteration 1: $F_0 = \{13, 16, 26, 37, 46\},\ F_1 = \{34, 67\},\ f = 34.$

Oracle, called to find a weakly constrained tangled tour that maximizes the left-hand side $a^T\overline{x}$ of (20) subject to

$\overline{x}_{13} = \overline{x}_{14} = \overline{x}_{16} = \overline{x}_{17} = \overline{x}_{24} = \overline{x}_{26} = \overline{x}_{27} = \overline{x}_{37} = \overline{x}_{45} = \overline{x}_{46} = \overline{x}_{57} = 0,$
$\overline{x}_{34} = 0$, and $\overline{x}_{67} = 1,$

returns the $\overline{x}^{\max}$ represented by

- the path system with the single component 1-2-5-3-6-7-4;

since $a^T\overline{x}^{\max} = 11$, we replace (20) by

$$\overline{x}_{12} + \overline{x}_{15} + 2\overline{x}_{23} + 2\overline{x}_{25} + 4\overline{x}_{34} + 3\overline{x}_{35} + 4\overline{x}_{36} + \overline{x}_{47} + 3\overline{x}_{56} \leq 11. \tag{21}$$

Iteration 2: $F_0 = \{13, 16, 26, 37, 46\},\ F_1 = \{67\},\ f = 67.$

Oracle, called to find a weakly constrained tangled tour that maximizes the left-hand side $a^T\overline{x}$ of (21) subject to

$\overline{x}_{13} = \overline{x}_{14} = \overline{x}_{16} = \overline{x}_{17} = \overline{x}_{24} = \overline{x}_{26} = \overline{x}_{27} = \overline{x}_{37} = \overline{x}_{45} = \overline{x}_{46} = \overline{x}_{57} = 0,$
$\overline{x}_{67} = 0,$

returns the $\overline{x}^{\max}$ represented by

- the path system with the single component 1-2-5-6-3-4-7;

since $a^T\overline{x}^{\max} = 15$, we replace (21) by

$$\overline{x}_{12} + \overline{x}_{15} + 2\overline{x}_{23} + 2\overline{x}_{25} + 4\overline{x}_{34} + 3\overline{x}_{35} + 4\overline{x}_{36} + \overline{x}_{47} + 3\overline{x}_{56} + 4\overline{x}_{67} \leq 15. \tag{22}$$

The second **while** loop of Algorithm 44 may go as follows.

Iteration 1: $F_0 = \{13, 16, 26, 37, 46\},\ F_1 = \emptyset,\ f = 13.$

Oracle, called to find a weakly constrained tangled tour that maximizes the left-hand side $a^T\overline{x}$ of (22) subject to

$\overline{x}_{14} = \overline{x}_{16} = \overline{x}_{17} = \overline{x}_{24} = \overline{x}_{26} = \overline{x}_{27} = \overline{x}_{37} = \overline{x}_{45} = \overline{x}_{46} = \overline{x}_{57} = 0,$
$\overline{x}_{13} = 1,$

returns the $\overline{x}^{\max}$ represented by

- the path system with the single component 1-3-4-7-6-5-2;

since $a^T\overline{x}^{\max} = 14$, we replace (22) by

$$\begin{aligned}\overline{x}_{12} + \overline{x}_{13} + \overline{x}_{15} + 2\overline{x}_{23} + 2\overline{x}_{25} + 4\overline{x}_{34}\\ +3\overline{x}_{35} + 4\overline{x}_{36} + \overline{x}_{47} + 3\overline{x}_{56} + 4\overline{x}_{67} \leq 15.\end{aligned} \tag{23}$$

Iteration 2: $F_0 = \{16, 26, 37, 46\},\ F_1 = \emptyset,\ f = 16.$

Oracle, called to find a weakly constrained tangled tour that maximizes the left-hand side $a^T\overline{x}$ of (23) subject to

$\overline{x}_{14} = \overline{x}_{17} = \overline{x}_{24} = \overline{x}_{26} = \overline{x}_{27} = \overline{x}_{37} = \overline{x}_{45} = \overline{x}_{46} = \overline{x}_{57} = 0,$
$\overline{x}_{16} = 1,$

returns the $\overline{x}^{\max}$ represented by

- the path system with the single component 1-6-7-4-3-5-2;

since $a^T\overline{x}^{\max} = 14$, we replace (23) by

$$\begin{aligned}\overline{x}_{12} + \overline{x}_{13} + \overline{x}_{15} + \overline{x}_{16} + 2\overline{x}_{23} + 2\overline{x}_{25} + 4\overline{x}_{34}\\ +3\overline{x}_{35} + 4\overline{x}_{36} + \overline{x}_{47} + 3\overline{x}_{56} + 4\overline{x}_{67} \leq 15.\end{aligned} \tag{24}$$

Iteration 3: $F_0 = \{26, 37, 46\},\ F_1 = \emptyset,\ f = 26.$

Oracle, called to find a weakly constrained tangled tour that maximizes the left-hand side $a^T\overline{x}$ of (24) subject to

$\overline{x}_{14} = \overline{x}_{17} = \overline{x}_{24} = \overline{x}_{27} = \overline{x}_{37} = \overline{x}_{45} = \overline{x}_{46} = \overline{x}_{57} = 0,$
$\overline{x}_{26} = 1,$

returns the $\overline{x}^{\max}$ represented by

- the path system with the single component 1-5-3-4-7-6-2;

since $a^T\overline{x}^{\max} = 13$, we replace (24) by

$$\begin{aligned}\overline{x}_{12} + \overline{x}_{13} + \overline{x}_{15} + \overline{x}_{16} + 2\overline{x}_{23} + 2\overline{x}_{25} + 2\overline{x}_{26} + 4\overline{x}_{34}\\ +3\overline{x}_{35} + 4\overline{x}_{36} + \overline{x}_{47} + 3\overline{x}_{56} + 4\overline{x}_{67} \leq 15.\end{aligned} \tag{25}$$

Iteration 4: $F_0 = \{37, 46\},\ F_1 = \emptyset,\ f = 37.$

Oracle, called to find a weakly constrained tangled tour that maximizes the left-hand side $a^T\overline{x}$ of (25) subject to

$\overline{x}_{14} = \overline{x}_{17} = \overline{x}_{24} = \overline{x}_{27} = \overline{x}_{45} = \overline{x}_{46} = \overline{x}_{57} = 0,$
$\overline{x}_{37} = 1,$

returns the $\overline{x}^{\max}$ represented by

- the path system with the single component 1-2-5-6-7-3-4;

since $a^T\overline{x}^{\max} = 14$, we replace (25) by

$$\begin{aligned}\overline{x}_{12} + \overline{x}_{13} + \overline{x}_{15} + \overline{x}_{16} + 2\overline{x}_{23} + 2\overline{x}_{25} + 2\overline{x}_{26} + 4\overline{x}_{34}\\ +3\overline{x}_{35} + 4\overline{x}_{36} + \overline{x}_{37} + \overline{x}_{47} + 3\overline{x}_{56} + 4\overline{x}_{67} \leq 15.\end{aligned} \tag{26}$$

Iteration 5: $F_0 = \{46\},\ F_1 = \emptyset,\ f = 46$.

Oracle, called to find a weakly constrained tangled tour that maximizes the left-hand side $a^T\overline{x}$ of (26) subject to

$$\begin{aligned}&\overline{x}_{14} = \overline{x}_{17} = \overline{x}_{24} = \overline{x}_{27} = \overline{x}_{45} = \overline{x}_{57} = 0,\\ &\overline{x}_{46} = 1,\end{aligned}$$

returns the $\overline{x}^{\max}$ represented by

- the path system with the single component 1-2-5-3-4-6-7;

since $a^T\overline{x}^{\max} = 14$, we replace (26) by

$$\begin{aligned}\overline{x}_{12} + \overline{x}_{13} + \overline{x}_{15} + \overline{x}_{16} + 2\overline{x}_{23} + 2\overline{x}_{25} + 2\overline{x}_{26} + 4\overline{x}_{34}\\ +3\overline{x}_{35} + 4\overline{x}_{36} + \overline{x}_{37} + \overline{x}_{46} + +\overline{x}_{47} + 3\overline{x}_{56} + 4\overline{x}_{67} \leq 15.\end{aligned} \tag{27}$$

After this iteration, $F_0 = F_1 = \emptyset$, and so (27) induces a facet of the convex hull of all weakly constrained tangled tours.

### 4.5 Phase 3: **From Weakly Constrained to All Tangled Tours**

Phase 2 produces an inequality $a^T\overline{x} \leq b$ such that

(i) $a^T\overline{x} \leq b$ induces a facet of the convex hull of all weakly constrained tangled tours,
(ii) $a^T\overline{x}^* > b$,
(iii) $a_e = 0$ for all $e$ outside $\overline{E}^{\text{int}}$;

since the set of restrictions of weakly constrained tangled tours onto $\overline{E}^{\text{int}}$ is monotone, (i), (ii), and (iii) imply that

(iv) $a_e > 0$ for all $e$ in $\overline{E}^{\text{int}}$.

A tangled tour $\overline{x}$ is weakly constrained if and only if

$$\sum(\overline{x}_e : e \in \overline{E}^{\text{int}}, w \in e) = 2 \text{ for all } w \text{ in } \overline{V}^{\text{int}};$$

if a hypergraph inequality

$$\sum(\lambda_Q x(Q, \overline{V} - Q) : Q \in \overline{\mathcal{H}}) \geq \beta \tag{28}$$

is related to $a^T\overline{x} \leq b$ in the sense that, for some numbers $\pi_w (w \in \overline{V}^{\text{int}})$, the left-hand side of (28) is identically equal to

$$\sum(\pi_w \sum(\overline{x}_e : w \in e) : w \in \overline{V}^{\text{int}}) - 2a^T\overline{x}$$

and the right-hand side of (28) equals

$$\sum(2\pi_w : w \in \overline{V}^{\text{int}}) - 2b,$$

then (28) induces a facet of the convex hull of all weakly constrained tangled tours and is violated by $\overline{x}^*$. In Phase 3, we find a hypergraph inequality that is related to $a^T\overline{x} \leq b$ in this sense and is satisfied by all tangled tours. Algorithm 45 accomplishes this objective with a negligible amount of computations.

**Algorithm 45.** Phase 3:

construct a hypergraph $\overline{\mathcal{H}}$ on $\overline{V}^{\text{int}}$ and positive integers $\lambda_Q (Q \in \overline{\mathcal{H}})$
such that the linear form
$\sum \lambda_Q(\sum(\overline{x}_e : e \subseteq Q) : Q \in \overline{\mathcal{H}})$
is identically equal to $a^T\overline{x}$;
return the inequality
$\sum(\lambda_Q x(Q, \overline{V} - Q) : Q \in \overline{\mathcal{H}}) \geq \sum(2\lambda_Q |Q| : Q \in \overline{\mathcal{H}}) - 2b$;

Arguments used by Naddef and Rinaldi [53] show that the hypergraph inequality returned by Algorithm 45 is satisfied by all tangled tours; it is a routine matter to verify that the numbers defined by $\pi_w = \sum(\lambda_Q : w \in Q, Q \in \overline{\mathcal{H}})$ have the desired properties.

A straightforward way of constructing the $\overline{\mathcal{H}}$ and the $\lambda_Q (Q \in \overline{\mathcal{H}})$ in Algorithm 45 is to let $\overline{\mathcal{H}}$ consist of all two-point subsets $e$ such that $a_e > 0$ and to set $\lambda_e = a_e$ for all $e$ in $\overline{\mathcal{H}}$. Instead, Concorde constructs $\overline{\mathcal{H}}$ and $\lambda_Q (Q \in \overline{\mathcal{H}})$ by an iterative greedy procedure: in each iteration, it chooses a maximal (with respect to set-inclusion) subset $Q$ of $\overline{V}^{\text{int}}$ such that $a_e > 0$ for all edges $e$ with both endpoints in $Q$, it lets $\lambda_Q$ be the smallest of these positive $a_e$, it brings $Q$ into $\overline{\mathcal{H}}$, and it subtracts $\lambda_Q$ from all $a_e$ such that $e$ has both endpoints in $Q$.

In our illustrative example, the inequality $a^T\overline{x} \leq b$ produced in Phase 2 reads

$$\begin{aligned}\overline{x}_{12} + \overline{x}_{13} + \overline{x}_{15} + \overline{x}_{16} + 2\overline{x}_{23} + 2\overline{x}_{25} + 2\overline{x}_{26} + 4\overline{x}_{34} \\ +3\overline{x}_{35} + 4\overline{x}_{36} + \overline{x}_{37} + \overline{x}_{46} + +\overline{x}_{47} + 3\overline{x}_{56} + 4\overline{x}_{67} \leq 15;\end{aligned}$$

we construct the $\overline{\mathcal{H}}$ and the $\lambda_Q (Q \in \overline{\mathcal{H}})$ in Algorithm 45 as follows.

1. We choose $Q = \{1, 2, 3, 5, 6\}$, which yields $\lambda_Q = 1$ and leaves us with

$$a^T x = \overline{x}_{23} + \overline{x}_{25} + \overline{x}_{26} + 4\overline{x}_{34} + 2\overline{x}_{35} + 3\overline{x}_{36} + \overline{x}_{37} + \overline{x}_{46} + \overline{x}_{47} + 2\overline{x}_{56} + 4\overline{x}_{67}.$$

2. We choose $Q = \{2, 3, 5, 6\}$, which yields $\lambda_Q = 1$ and leaves us with
$$a^T x = 4\overline{x}_{34} + \overline{x}_{35} + 2\overline{x}_{36} + \overline{x}_{37} + \overline{x}_{46} + \overline{x}_{47} + \overline{x}_{56} + 4\overline{x}_{67}.$$
3. We choose $Q = \{3, 4, 6, 7\}$, which yields $\lambda_Q = 1$ and leaves us with
$$a^T x = 3\overline{x}_{34} + \overline{x}_{35} + \overline{x}_{36} + \overline{x}_{56} + 3\overline{x}_{67}.$$
4. We choose $Q = \{3, 4\}$, which yields $\lambda_Q = 3$ and leaves us with
$$a^T x = \overline{x}_{35} + \overline{x}_{36} + \overline{x}_{56} + 3\overline{x}_{67}.$$
5. We choose $Q = \{3, 5, 6\}$, which yields $\lambda_Q = 1$ and leaves us with
$$a^T x = 3\overline{x}_{67}.$$
6. We choose $Q = \{6, 7\}$, which yields $\lambda_Q = 3$ and leaves us with
$$a^T x = 0.$$

The resulting hypergraph inequality,

$$\begin{aligned}
&\overline{x}(\{1,2,3,5,6\},\{0,4,7\})\\
&+\overline{x}(\{2,3,5,6\},\{0,1,4,7\})\\
&+\overline{x}(\{3,4,6,7\},\{0,1,2,5\})\\
&+3\overline{x}(\{3,4\},\{0,1,2,5,6,7\})\\
&+\overline{x}(\{3,5,6\},\{0,1,2,4,7\})\\
&+3\overline{x}(\{6,7\},\{0,1,2,3,4,5\}) \geq 26\,,
\end{aligned}$$

induces a facet of the convex hull of all weakly constrained tangled tours and is violated by $\overline{x}^*$. (By the way, this inequality belongs to the class of *path inequalities* of Cornuéjols, Fonlupt, and Naddef (1985).)

Every cut produced by Algorithm 45 can be easily transformed into a cut that induces a facet of the graphical traveling salesman polytope. It can be shown that, in case the $a^T\overline{x} \leq b$ produced in Phase 2 is such that

$$\text{every } a_e \text{ with } e \in \overline{E}^{\text{int}} \text{ is a positive integer,} \tag{29}$$

this $a^T\overline{x} \leq b$ reads

$$\lambda \sum(\overline{x}_e : e \in \overline{E}^{\text{int}}) \leq \lambda(k-1)$$

for some positive $\lambda$, and so Algorithm 45 returns a positive multiple of the subtour inequality

$$\overline{x}(\{0\}, \overline{V}^{\text{int}}) \geq 2;$$

as noted in Sect. 4.3, Cornuéjols, Fonlupt, and Naddef [14] pointed out that subtour inequalities induce facets of the convex hull of $\mathcal{T}$. It can also be shown that, in case (29) fails, the following procedure transforms the cut

$$\sum(\lambda_Q x(Q, \overline{V} - Q) : Q \in \overline{\mathcal{H}}) \geq \beta$$

produced by Algorithm 45 into a cut that induces a facet of the graphical traveling salesman polytope:

STEP 1. For all choices of distinct points $u$, $v$, $w$ of $\overline{V}$, define

$$\tau(u,v,w) = \sum(\lambda_Q : Q \in \overline{\mathcal{H}}, u \in Q, v \in Q, w \notin Q) + \sum(\lambda_Q : Q \in \overline{\mathcal{H}}, u \notin Q, v \notin Q, w \in Q).$$

STEP 2. For all points $w$ of $\overline{V}^{\text{int}}$, evaluate

$$\Delta_w = \min\{\tau(u,v,w) : uv \in \overline{E}, u \neq w, v \neq w\}.$$

STEP 3. Return the inequality

$$\sum(\lambda_Q x(Q, \overline{V} - Q) : Q \in \overline{\mathcal{H}}) - \sum(\Delta_w \overline{x}(\{w\}, \overline{V} - \{w\}) : w \in \overline{V}^{\text{int}}) \geq \beta - 2\sum(\Delta_w : w \in \overline{V}^{\text{int}}).$$

Again, the arguments come from Naddef and Rinaldi [53]; in their terminology, the inequality returned in STEP 3 is *tight triangular*.

The current version of Concorde can handle only hypergraph constraints with nonnegative coefficients; it settles for the cut produced by Algorithm 45 even when this cut does not induce a facet of the graphical traveling salesman polytope. Concorde's way of choosing $\overline{\mathcal{H}}$ and $\lambda_Q(Q \in \overline{\mathcal{H}})$ aims to mitigate the effects of this carelessness by reducing the number of points $w$ such that $\Delta_w > 0$.

In our illustrative example, $\overline{\mathcal{H}}$ consists of

$$\{1,2,3,5,6\},\ \{2,3,5,6\},\ \{3,4,6,7\},\ \{3,4\},\ \{3,5,6\},\ \{6,7\};$$

since no member of $\overline{\mathcal{H}}$ contains both 1 and 4, assumption (29) is satisfied; since

$$\begin{aligned}
\Delta_1 &= \tau(0,3,1) = 0,\\
\Delta_2 &= \tau(0,3,2) = 0,\\
\Delta_3 &= \tau(4,5,3) = 0,\\
\Delta_4 &= \tau(0,3,4) = 0,\\
\Delta_5 &= \tau(0,3,5) = 0,\\
\Delta_6 &= \tau(3,7,6) = 0,\\
\Delta_7 &= \tau(0,6,7) = 0,
\end{aligned}$$

we have $\Delta_w = 0$ for all $w = 1, 2, \ldots, 7$. It follows that this path inequality is tight triangular and induces a facet of the graphical traveling salesman polytope.

## 5 Making Choices of $V_0, V_1, \ldots, V_k$

Concorde's choices of $V_0, V_1, \ldots, V_k$ in Algorithm 31 are guided by $x^*$ in a way similar to that used by Christof and Reinelt [12] in their algorithm for finding cuts that match templates from a prescribed large catalog. First, it constructs

once and for all an equivalence relation on $V$ in such a way that each equivalence class $V^\star$ of this relation satisfies

$$x^*(V^\star, V - V^\star) = 2;$$

then it makes many different choices of $V_0, V_1, \ldots, V_k$ in such a way that each of $V_1, \ldots, V_k$ is one of these equivalence classes and $V_0 = V - (V_1 \cup \ldots \cup V_k)$.

With $W$ standing for the set of the equivalence classes on $V$, the first stage amounts to preshrinking $V$ onto $W$; making each of the many different choices of $V_0, V_1, \ldots, V_k$ in the second stage means choosing a small subset of $W$ and shrinking the entire remainder of $W$ onto a single point. In terms of the preshrunk set $W$, each choice of $V_0, V_1, \ldots, V_k$ in the second stage zooms in onto a relatively small part of the problem – typically $k$ is at most thirty or so and $|W|$ may run to hundreds or thousands – and effectively discards the rest. For this reason, we developed the habit of referring to the cuts produced by Algorithm 31 as *local cuts* and referred to them by this name in Applegate et al. [2]. In terms of the original $V$, each of the sets $V_1, \ldots, V_k$ could be quite large, which makes the qualifier "local" a misnomer. Still, a crisp label for the cuts produced by Algorithm 31 is convenient to have and "local cuts" seems to be as good a name as any other that we managed to think up.

The equivalence relation is constructed by iteratively shrinking two-point sets into a single point. At each stage of this process, we have a set $W$ and a mapping $\pi : W \to 2^V$ that defines a partition of $V$ into pairwise disjoint subsets $\pi(w)$ with $w \in W$. Initially, $W = V$ and each $\pi(w)$ is the singleton $\{w\}$; as long as there are distinct elements $u, v, w$ of $W$ such that

$$x^*(\pi(u), \pi(v)) = 1 \quad \text{and} \quad x^*(\pi(u), \pi(w)) + x^*(\pi(v), \pi(w)) = 1, \tag{30}$$

we keep replacing $\pi(u)$ by $\pi(u) \cup \pi(v)$ and removing $v$ from $W$; when there are no $u, v, w$ with property (30), we stop. (During this process, we may discover pairs $u, v$ with $x^*(\pi(u), \pi(v)) > 1$, in which case $x^*$ violates the subtour inequality $x(Q, V - Q) \geq 2$ with $Q = \pi(u) \cup \pi(v)$.)

To make the many different choices of $V_1, \ldots, V_k$, we first set the value of a parameter $t$ that nearly determines the value of $k$ in the sense that $t - 3 \leq k \leq t$. Then, for each $w$ in $W$, we choose a subset $C$ of $W$ so that $w \in C$ and $t - 3 \leq |C| \leq t$; the corresponding $V_1, \ldots, V_k$ are the $\pi(v)$ with $v \in C$. The choice of $C$ is guided by the graph with vertex-set $W$, where $u$ and $v$ are adjacent if and only if $x^*(\pi(u), \pi(v)) > \varepsilon$ for some prescribed zero tolerance $\varepsilon$: starting at $w$, we carry out a breadth-first search through this graph, until we collect a set $C$ of $t - 3$ vertices. If there are any vertices $u$ outside this $C$ such that $x^*(\pi(u), \pi(v)) = 1$ for some $v$ in $C$, then we keep adding these vertices $u$ to $C$ as long as $|C| \leq t$.

It seems plausible that such a crude way of choosing $C$ can be improved. However, we found its performance satisfactory; none of the alternatives that we tried appeared to work better.

# 6 Experimental Findings

Reinelt [64] created a library named TSPLIB that contains sample instances of the TSP (and related problems) from various sources and of various types. There are 110 TSP instances in this library. Some of them arise from the task of drilling holes in printed circuit boards and others have been constructed artificially, often in the Dantzig-Fulkerson-Johnson tradition of choosing a set of actual cities and defining the cost of travel from X to Y as the distance between X and Y. None of them (with a single exception, the problem named `ts225`) is contrived to be hard and none of them is contrived to be easy; 106 of them have been solved and four have not.

The results reported in this section involve various TSP instances drawn from TSPLIB. The *default code* is Concorde 99.12.15 of Applegate et al. [3] with 99 as the random number seed (`concorde -s 99 xxx.tsp`). The *running times* are given in seconds on a Compaq XP1000 workstation with a 500 MHz EV6 Alpha processor.

## 6.1 The Easier TSPLIB Instances

Our default code solved 87 of the 110 TSPLIB instances in under 1000 seconds. These instances are listed in Table 1.

In Sect. 5, we described a way of producing a number of partititions of $V$ into nonempty sets $V_0, V_1, \ldots, V_k$ with $t-3 \leq k \leq t$ for a prescribed positive integer $t$. Concorde uses it with $t$ ranging between 8 and a prescribed integer $t_{\max}$. More precisely, the search always begins with $t = 8$. Whenever a value of $t$ is set, Concorde adds all the resulting cuts produced by Algorithm 31 to the LP relaxation of our problem and it solves the tightened LP relaxation; if the increase in the value of the relaxation is not satisfactory and $t < t_{\max}$, then the next iteration takes place with $t$ incremented by one. The default setting, $t_{\max} = 16$, was prompted by results reported in Table 2. The trends exhibited in this table are hardly surprising: increases of $t_{\max}$ yield tighter LP relaxations, but they also require additional time to construct these tighter relaxations.

Table 2 shows also that the total time to solve the 87 TSPLIB instances in Table 1 by our default code would increase to 126.8% of its original value if local cuts were turned off. This endorsement of local cuts pales in comparison with other cutting-plane routines: if we turned off a class of comb-finding procedures proposed by Padberg and Rinaldi [62] (based on the block decomposition of the graph obtained by considering only edges $e$ with $0 < x_e^* < 1$), then the running time over our default code would increase to 427.2% of its original value.

## 6.2 Three of the Harder TSPLIB Instances

Two of the harder TSPLIB instances solved by Concorde are the printed-circuit board instance `pcb3038` and the geographical instance `fnl4461`. On each of them, we have run tests similar to those of Table 2; their results are reported

**Table 1.** 87 instances from the TSPLIB
NT denotes the number of nodes in the branch-and-cut tree

| name | NT | time | name | NT | time | name | NT | time |
|---|---|---|---|---|---|---|---|---|
| burma14 | 1 | 0.06 | lin105 | 1 | 0.59 | lin318 | 1 | 9.74 |
| ulysses16 | 1 | 0.22 | pr107 | 1 | 1.03 | rd400 | 15 | 148.42 |
| gr17 | 1 | 0.08 | gr120 | 1 | 2.23 | fl417 | 5 | 57.75 |
| gr21 | 1 | 0.03 | pr124 | 1 | 3.64 | gr431 | 13 | 133.29 |
| ulysses22 | 1 | 0.53 | bier127 | 1 | 1.65 | pr439 | 15 | 216.75 |
| gr24 | 1 | 0.07 | ch130 | 1 | 2.13 | pcb442 | 9 | 49.92 |
| fri26 | 1 | 0.07 | pr136 | 1 | 3.97 | d493 | 5 | 113.32 |
| bayg29 | 1 | 0.09 | gr137 | 1 | 3.42 | att532 | 7 | 109.52 |
| bays29 | 1 | 0.13 | pr144 | 1 | 2.58 | ali535 | 3 | 53.14 |
| dantzig42 | 1 | 0.23 | ch150 | 1 | 3.03 | si535 | 3 | 43.13 |
| swiss42 | 1 | 0.13 | kroA150 | 1 | 5.00 | pa561 | 17 | 246.82 |
| att48 | 1 | 0.56 | kroB150 | 1 | 4.23 | u574 | 1 | 23.04 |
| gr48 | 1 | 0.67 | pr152 | 1 | 7.93 | rat575 | 25 | 363.07 |
| hk48 | 1 | 0.17 | u159 | 1 | 1.00 | p654 | 3 | 26.52 |
| eil51 | 1 | 0.73 | si175 | 3 | 13.09 | d657 | 13 | 260.37 |
| berlin52 | 1 | 0.29 | brg180 | 1 | 1.46 | gr666 | 3 | 49.86 |
| brazil58 | 1 | 0.68 | rat195 | 5 | 22.23 | u724 | 11 | 225.44 |
| st70 | 1 | 0.50 | d198 | 3 | 11.82 | rat783 | 1 | 37.88 |
| eil76 | 1 | 0.30 | kroA200 | 1 | 6.59 | dsj1000 | 7 | 410.32 |
| pr76 | 1 | 1.86 | kroB200 | 1 | 3.91 | pr1002 | 1 | 34.30 |
| gr96 | 1 | 6.71 | gr202 | 1 | 5.01 | si1032 | 1 | 25.47 |
| rat99 | 1 | 0.95 | ts225 | 1 | 20.52 | u1060 | 21 | 571.43 |
| kroA100 | 1 | 1.00 | tsp225 | 1 | 15.01 | vm1084 | 11 | 604.78 |
| kroB100 | 1 | 2.36 | pr226 | 1 | 4.35 | pcb1173 | 19 | 468.27 |
| kroC100 | 1 | 0.96 | gr229 | 3 | 38.61 | rl1304 | 1 | 189.20 |
| kroD100 | 1 | 1.00 | gil262 | 1 | 13.06 | nrw1379 | 19 | 578.42 |
| kroE100 | 1 | 2.44 | pr264 | 1 | 2.67 | u1432 | 3 | 223.70 |
| rd100 | 1 | 0.67 | a280 | 3 | 5.37 | d1655 | 5 | 263.03 |
| eil101 | 1 | 0.74 | pr299 | 3 | 17.49 | pr2392 | 1 | 116.86 |

in Table 3. (For the sake of uniformity, we have started each of the nine runs on each of the two instances with the value of the optimal tour as the upper bound.)

As with Table 2, the trends exhibited in Table 3 are hardly surprising, although the dependence of running time on $t_{\max}$ is not quite as neat for `fnl4461` (note its increase as $t_{\max}$ moves from 0 to 8) and it is even more erratic for `pcb3038`. One striking difference between the easier instances on the one hand and `pcb3038`, `fnl4461` on the other hand is an increase in the optimal setting of $t_{\max}$, which is 28 for `pcb3038` and 32 for `fnl4461`. These experimental results agree with the intuition that the harder instances are better off with the larger values of $t_{\max}$. Another striking difference is the increased effect of local cuts on the overall running time. With `pcb3038`, local cuts with the default $t_{\max} = 16$ reduce the running time to 50.3% of its original value and setting $t_{\max} = 28$ reduces it further to 43.8%. For `fnl4461`, the figures are more impressive: local

**Table 2.** The effect of $t_{\max}$ on the easier TSPLIB instances

| $t_{\max}$ | total time to solve the 87 instances in Table 1 | number of instances solved without branching |
|---|---|---|
| 0 | 7424.58 | 42 |
| 8 | 6679.15 | 48 |
| 10 | 6624.02 | 52 |
| 12 | 6248.12 | 54 |
| 14 | 6133.06 | 59 |
| 16 | 5900.72 | 59 |
| 18 | 6394.10 | 64 |
| 20 | 8818.98 | 65 |
| 22 | 9591.02 | 65 |
| 24 | 16519.62 | 68 |
| 26 | 23285.19 | 67 |
| 28 | 35798.49 | 64 |
| 30 | 40732.42 | 66 |

**Table 3.** The effect of $t_{\max}$ on two harder TSPLIB instances

root LP = optimal value of the LP relaxation before the first branching
NT denotes the number of nodes in the branch-and-cut tree

| `pcb3038` optimal value = 137694 | | | | `fnl4461` optimal value = 182566 | | | |
|---|---|---|---|---|---|---|---|
| $t_{\max}$ | root LP | NT | time | $t_{\max}$ | root LP | NT | time |
| 0 | 137592.61 | 665 | 145241.25 | 0 | 182471.39 | 14977 | 1990568.46 |
| 8 | 137595.10 | 659 | 162228.02 | 8 | 182487.75 | 11903 | 2286217.24 |
| 12 | 137613.44 | 383 | 105026.74 | 12 | 182506.27 | 2011 | 297019.61 |
| 16 | 137625.28 | 271 | 73085.72 | 16 | 182519.87 | 1071 | 194441.98 |
| 20 | 137637.25 | 271 | 119042.35 | 20 | 182530.12 | 417 | 78398.31 |
| 24 | 137640.66 | 155 | 65757.78 | 24 | 182541.98 | 213 | 53420.13 |
| 28 | 137644.28 | 107 | 63678.80 | 28 | 182543.60 | 137 | 49719.02 |
| 32 | 137643.42 | 127 | 129460.96 | 32 | 182546.60 | 81 | 44957.02 |
| 36 | 137651.53 | 101 | 233092.39 | 36 | 182549.72 | 59 | 56714.06 |

cuts with the default $t_{\max} = 16$ reduce the running time to 9.8% of its original value and setting $t_{\max} = 32$ reduces it further to 2.3%.

The hardest TSPLIB instances that we have solved are `usa13509` and `d15112`. Batoukov and Sørevik [6] refer to our solution of `usa13509` as one of "only a few heroic examples on successful computation on a network of workstations" and we do not propose to contradict them. This instance was solved by running an earlier version of Concorde in parallel on a network of 48 workstations, including Digital Alphas, Intel Pentium IIs and Pentium Pros, and Sun UltraSparcs; a very rough estimate of the total running time is about 4 years on our single Compaq XP1000.

Four years of CPU time is an exorbitant figure. We believe that without the use of local cuts, this exorbitant figure would grow further to a level which

would put solving `usa13509` out of our reach even if we had gathered many more workstations for the heroic purpose.

The experimental results presented in Table 4 support this sentiment. Its 29 rows correspond to 29 runs that tighten up the root LP relaxation. Each of these runs starts where the preceding run has stopped and it takes note of the gap $g$ between the length of the optimal tour and the optimal value of the current LP relaxation. Then it attempts to tighten the relaxation by additional cuts and it solves this tightened relaxation; this step gets iterated as long as the optimal value of the relaxation keeps increasing by at least a prescribed fixed percentage of $g$.

We begin with five runs of Def C00, meaning our default code with local cuts turned off. The last three of these runs yields only imperceptible improvements and they narrow the relative gap to 0.0561%. Then we follow with four runs of All C00, meaning Def C00 with additional cutting-plane routines (other than local cuts) that we found to be a hindrance in solving the easy instances from Table 1 but that might help in solving the harder instances. (These techniques include our implementation of what we understood from Naddef and Thienel [54, 55], the dominos-and-necklaces heuristic for finding violated comb inequalities that has been described in Sect. 3 of Applegate et al. [1], and other cutting-plane routines.) The last three of these runs yields only imperceptible improvements and they narrow the relative gap to 0.0435%. This is the best we can do without the use of local cuts.

Then we bring in local cuts in nineteen runs of the default code with $t_{\max}$ progressing through the sequence 8, 10, 12, ..., 44. As a result, the relative gap is narrowed to 0.0164%, less than two fifths of the previous value. Finally, an additional run of All C00 yields only an imperceptible improvement.

## 7 Generalizations

The cutting-plane method and its descendants are applicable to any problem

$$\text{minimize } c^T x \text{ subject to } x \in \mathcal{S},$$

where $\mathcal{S}$ is a finite subset of some Euclidean space $\mathbb{R}^m$, provided that an efficient algorithm to recognize points of $\mathcal{S}$ is available. The corresponding general problem of finding cuts is this:

given (by means of an efficient membership-testing oracle)
a finite subset $\mathcal{S}$ of some $\mathbb{R}^m$ and
given a point $x^*$ in $\mathbb{R}^m$ that lies outside the convex hull of $\mathcal{S}$,
find a vector $a$ and a scalar $b$ such that

$$\mathcal{S} \subset \{x : a^T x \leq b\} \text{ and } a^T x^* > b.$$

Algorithm 31, dealing with the special case case where $m = n(n-1)/2$ and $\mathcal{S}$ is the set of the incidence vectors of all the tours through a set of $n$ cities, generalizes to Algorithm 71. (In the special case, $d = (k+1)k/2$, $\phi(x) = \overline{x}$, and $\mathcal{T}$ is the set of all tangled tours through $\overline{V}$.)

**Table 4.** Local cuts and `usa13509`

| `usa13509` optimal value = 19982859 | | | |
|---|---|---|---|
| Code | root LP | gap | time |
| Def C00 | 19967864.84 | 0.0751% | 2157.08 |
| +Def C00 | 19970751.68 | 0.0606% | 2795.47 |
| +Def C00 | 19971645.49 | 0.0561% | 1140.06 |
| +Def C00 | 19971656.19 | 0.0561% | 165.04 |
| +Def C00 | 19971657.09 | 0.0561% | 155.35 |
| +All C00 | 19973413.70 | 0.0473% | 4293.28 |
| +All C00 | 19974175.97 | 0.0435% | 9455.69 |
| +All C00 | 19974177.19 | 0.0435% | 733.44 |
| +All C00 | 19974180.02 | 0.0435% | 766.91 |
| +Def C08 | 19974585.45 | 0.0414% | 2559.30 |
| +Def C10 | 19975711.27 | 0.0358% | 5131.98 |
| +Def C12 | 19976111.11 | 0.0338% | 3931.73 |
| +Def C14 | 19976520.40 | 0.0317% | 4355.10 |
| +Def C16 | 19976898.82 | 0.0298% | 4288.43 |
| +Def C18 | 19977508.51 | 0.0268% | 8964.28 |
| +Def C20 | 19978299.28 | 0.0228% | 10901.51 |
| +Def C22 | 19978577.43 | 0.0214% | 6126.33 |
| +Def C24 | 19978738.27 | 0.0206% | 5478.57 |
| +Def C26 | 19978861.69 | 0.0200% | 5556.20 |
| +Def C28 | 19979011.25 | 0.0193% | 10791.34 |
| +Def C30 | 19979133.77 | 0.0186% | 6802.63 |
| +Def C32 | 19979135.92 | 0.0186% | 1676.74 |
| +Def C34 | 19979382.66 | 0.0174% | 32465.95 |
| +Def C36 | 19979384.65 | 0.0174% | 7018.06 |
| +Def C38 | 19979507.22 | 0.0168% | 36408.27 |
| +Def C40 | 19979571.48 | 0.0165% | 25237.98 |
| +Def C42 | 19979574.20 | 0.0164% | 9143.59 |
| +Def C44 | 19979576.01 | 0.0164% | 12623.30 |
| +All C00 | 19979583.83 | 0.0164% | 1316.49 |

**Algorithm 71.** *A very general scheme for collecting cuts*

initialize an empty list $\mathcal{L}$ of cuts;
**for** selected small integers $d$, linear mappings $\phi : \mathbb{R}^m \to \mathbb{R}^d$, and finite subsets $\mathcal{T}$ of $\mathbb{R}^d$ such that $\phi(\mathcal{S}) \subseteq \mathcal{T}$
**do** **if** $\phi(x^*)$ lies outside the convex hull of $\mathcal{T}$
**then** find a vector $a$ and a scalar $b$ such that
$$\mathcal{T} \subset \{\overline{x} : a^T\overline{x} \leq b\} \text{ and } a^T\phi(x^*) > b;$$
add the cut $a^T\phi(x) \leq b$ to $\mathcal{L}$;
**end**
**end**
return $\mathcal{L}$;

The trick of trying to separate $x^*$ from $\mathcal{S}$ by separating $\phi(x^*)$ from $\mathcal{T}$ was used previously by Crowder, Johnson, and Padberg [17] in the context of integer linear programming, where $\mathcal{S}$ consists of all integer solutions of some explicitly recorded system

$$Ax = b,\ \ell \leq x \leq u \tag{31}$$

and $x^*$ satisfies (31) in place of $x$. Crowder, Johnson, and Padberg consider systems (31) such that $A$ is sparse and $\ell = 0, u = e$; for each equation $\alpha^T x = \beta$ in the system $Ax = b$, they consider the set $\mathcal{T}$ of all integer solutions of

$$\alpha^T x = \beta,\ \ 0 \leq x \leq e$$

restricted on components $x_j$ such that $\alpha_j \neq 0$; with $\phi(x)$ standing for the restriction of $x$ on these components, they try to separate $x^*$ from $\mathcal{S}$ by separating $\phi(x^*)$ from $\mathcal{T}$. In the attempt to separate $\phi(x^*)$ from $\mathcal{T}$, they use exclusively inequalities that match certain prescribed templates and they use special-purpose separation algorithms to find such cuts.

Boyd [8,9] starts out with the choices of $\phi$ and $\mathcal{T}$ made by Crowder, Johnson, and Padberg, but then he separates $\phi(x^*)$ from $\mathcal{T}$ by a general-purpose procedure. He solves the problem

$$\begin{array}{ll} \text{maximize} & z \\ \text{subject to} & z - a^T\phi(x^*) + a^T\phi(x) \leq 0 \text{ for all } x \text{ in } \mathcal{T}, \\ & ||a||_1 \leq \gamma,\ ||a||_\infty \leq 1 \end{array}$$

with a prescribed constant $\gamma$ by a method that is essentially the simplex method (in particular, the value of $z$ increases with each nondegenerate iteration); to access $\mathcal{T}$, he uses an oracle implemented by a dynamic programming algorithm. If the optimal value $z^*$ turns out to be positive, then he returns the cut

$$a^T\phi(x) \leq a^T\phi(x^*) - z^*,$$

which he calls a *Fenchel cutting plane.*

The technique described in Sect. 4.1 provides another implementation of the body of the **for** loop in Algorithm 71 for a fairly general class of sets $\mathcal{T}$: it requires only an efficient oracle that, given a vector $a$ in $\mathbb{R}^d$, returns either an $\overline{x}$ in $\mathcal{T}$ that maximizes $a^T\overline{x}$ or the message "infeasible" indicating that $\mathcal{T}$ is empty. This technique – presented as Algorithm 41 in the special case where the set of all strongly constrained tangled tours is substituted for $\mathcal{T}$ – is reviewed as Algorithm 72: function Separate, given (by means of an efficient maximization oracle) a finite subset $\mathcal{T}$ of some $\mathbb{R}^d$ and given a point $\overline{x}^*$ in $\mathbb{R}^d$, returns either the message "$\overline{x}^*$ is in the convex hull of $\mathcal{T}$" or a vector $a$ and a scalar $b$ such that

$$\mathcal{T} \subset \{\overline{x} : a^T\overline{x} \leq b\} \text{ and } a^T\overline{x}^* > b.$$

**Algorithm 72.** SEPARATE($\overline{x}^*, \mathcal{T}$)

**if** $\mathcal{T} \neq \emptyset$
**then** $A =$ any $d \times 1$ matrix whose single column is an element of $\mathcal{T}$;
  **repeat if** the linear programming problem

$$\begin{aligned} &\text{minimize } u^T e + v^T e \\ &\text{subject to } \quad -a^T A + b e^T \geq 0, \\ &\qquad\qquad\qquad a^T \overline{x}^* - b = 1, \\ &\qquad\qquad\qquad a^T + u^T - v^T = 0, \\ &\qquad\qquad\qquad\qquad u^T, v^T \geq 0 \end{aligned}$$

  has an optimal solution
  **then** find an element $\overline{x}$ of $\mathcal{T}$ that maximizes $a^T \overline{x}$;
    **if** $a^T \overline{x} \leq b$
    **then** return $a$ and $b$;
    **else** add $\overline{x}$ to $A$ as a new column;
    **end**
  **else** return the message
    "$\overline{x}^*$ is in the convex hull of $\mathcal{T}$";
  **end**
  **end**
**else** return 0 and $-1$;
**end**

One way of solving the linear programming problem in each iteration of the **repeat** loop in Algorithm 72 is to apply the simplex method to its dual with relatively few rows,

$$\begin{aligned} &\text{maximize } s \\ &\text{subject to } \quad s\overline{x}^* - A\lambda + w \qquad = 0, \\ &\qquad\qquad\quad -s + e^T \lambda \qquad\qquad = 0, \\ &\qquad\qquad\quad \lambda \geq 0, \quad -e \leq w \leq e, \end{aligned} \tag{32}$$

just as Concorde does in its implementation of Algorithm 41.

Modifications of SEPARATE($\overline{x}^*, \mathcal{T}$) used by Concorde in its implementation of Algorithm 31 may also apply to implementations of Algorithm 71. In particular, if we have at our disposal a nonempty family $\mathcal{F}$ of row vectors $[v^T, w]$ such that

$$\mathcal{T} \subset \{\overline{x} : v^T \overline{x} \geq w\} \text{ and } v^T \overline{x}^* = w,$$

then we may test the condition

$$\overline{x}^* \text{ lies outside the convex hull of } \mathcal{T}$$

in Algorithm 71 by calling SEPARATE($\overline{x}^*, \mathcal{T}^*$) with

$$\mathcal{T}^* = \{\overline{x} \in \mathcal{T} : v^T \overline{x} = w \text{ for all } [v^T, w] \text{ in } \mathcal{F}\}.$$

(In Concorde, $\mathcal{T}^*$ is the set of all strongly constrained tangled tours through $\overline{V}$.) The point of this substitution is that SEPARATE$(\overline{x}^*, \mathcal{T}^*)$ tends to run faster as the dimension of $\{\overline{x} \in \mathbb{R}^d : v^T\overline{x} = w \text{ for all } [v^T, w] \text{ in } \mathcal{F}\}$ decreases; if most of our choices of $d$, $\phi$, and $\mathcal{T}$ in Algorithm 71 place $\overline{x}^*$ in the convex hull of $\mathcal{T}$, then we may save time even if each successful call of SEPARATE$(\overline{x}^*, \mathcal{T}^*)$ is followed by a call of SEPARATE$(\overline{x}^*, \mathcal{T})$. In fact, if SEPARATE$(\overline{x}^*, \mathcal{T}^*)$ returns an inequality $a^{*T}\overline{x} \le b^*$ that separates $\overline{x}^*$ from $\mathcal{T}^*$, then there is no need to call SEPARATE$(\overline{x}^*, \mathcal{T})$: for every nonnegative $M$ such that

$$M \ \ge \ \frac{a^{*T}\overline{x} - b^*}{\sum(v^T\overline{x} - w : [v^T, w] \in \mathcal{F})} \quad \text{for all } \overline{x} \text{ in } \mathcal{T} - \mathcal{T}^*,$$

the inequality

$$a^{*T}\overline{x} - M \cdot \sum(v^T\overline{x} - w : [v^T, w] \in \mathcal{F}) \le b^*$$

separates $\overline{x}^*$ from $\mathcal{T}$.

Furthermore, as long as $\overline{x}^*$ is in the affine hull of $\mathcal{T}$, every cut separating $\overline{x}^*$ from $\mathcal{T}$ may be converted to a cut that induces a facet of the convex hull of $\mathcal{T}$ by the techniques described in Sect. 4.3, where the set of all moderately tangled tours through $\overline{V}$ plays the role of $\mathcal{T}$. Specifically, Algorithm 73, given any cut $a^T\overline{x} \le b$, returns a cut inducing a facet of the convex hull of $\mathcal{T}$. Here, a default $\mathcal{I}$ is supplied by the optimal basis of problem (32), a default $\mathcal{C}$ is supplied by all $[v^T, w]$ in $\mathcal{F}$ such that $\mathcal{T} \not\subset \{\overline{x} : v^T\overline{x} = w\}$, and TILT $(a, b, v, w, \overline{x}^0)$ is Algorithm 43 with $\mathcal{T}$ substituted for the set of all moderately tangled tours.

**Algorithm 73.** *From a cut $a^T\overline{x} \le b$ to a facet-inducing cut in general*

$\mathcal{I}$ = an affinely independent subset of $\{\overline{x} \in \mathcal{T} : a^T\overline{x} = b\}$;
$\mathcal{C}$ = a catalog of vectors $[v^T, w]$ such that $\mathcal{T} \subset \{\overline{x} : v^T\overline{x} \ge w\}$, $v^T\overline{x}^* \ge w$, and $v^T\overline{x} > w$ for some $\overline{x}$ in $\mathcal{T}$;
$\overline{x}_0$ = an arbitrary element of $\mathcal{T}$;
**for** all $[v^T, w]$ in $\mathcal{C}$
**do** remove $[v^T, w]$ from $\mathcal{C}$
  **if** $\mathcal{I} \subset \{\overline{x} : v^T\overline{x} = w\}$
  **then** $(a^+, b^+, \overline{x}^+) =$ TILT $(a, b, v, w, \overline{x}^0)$;
    $a = a^+$, $b = b^+$, $\mathcal{I} = \mathcal{I} \cup \{\overline{x}^+\}$;
  **end**
**end**
**while** $|\mathcal{I}| < \dim \mathcal{T}$
**do** $\overline{x}^0$ = an element of $\mathcal{T} - \mathcal{I}$ such that $\mathcal{I} \cup \{\overline{x}^0\}$ is affinely independent;
  **if** $a^T\overline{x}^0 = b$
  **then** replace $(a, b; \mathcal{I})$ by $(a, b; \mathcal{I} \cup \{\overline{x}^0\})$;
  **else** find a nonzero vector $v$ and a number $w$ such that
    $v^T\overline{x} - w = 0$ for all $\overline{x}$ in $\mathcal{I} \cup \{\overline{x}^0\}$,
    $v^T\overline{x} - w \ne 0$ for some $\overline{x}$ in $\mathcal{T}$;

$(a^+, b^+, \overline{x}^+) =$ TILT $(a, b, v, w, \overline{x}^0)$;
$(a^-, b^-, \overline{x}^-) =$ TILT $(a, b, -v, -w, \overline{x}^0)$;
**if** $a^{+T}\overline{x}^* - b^+ \geq a^{-T}\overline{x}^* - b^-$
**then** replace $(a, b; \mathcal{I})$ by $(a^+, b^+; \mathcal{I} \cup \{\overline{x}^+\})$;
**else** replace $(a, b; \mathcal{I})$ by $(a^-, b^-; \mathcal{I} \cup \{\overline{x}^-\})$;
**end**
**end**
**end**
return $a$ and $b$;

By the way, Algorithm 73 provides a constructive proof of the following corollary of a classic theorem of Minkowski ([50] Section 19):

**Theorem 1.** *Let $\mathcal{T}$ be a finite subset of some $\mathbb{R}^d$, let $\overline{x}^*$ be a point in the affine hull of $\mathcal{T}$, and let an inequality $a^T\overline{x} \leq b$ separate $\mathcal{T}$ from $\overline{x}^*$ in the sense that*

$$\mathcal{T} \subset \{\overline{x} : a^T\overline{x} \leq b\} \text{ and } a^T\overline{x}^* > b.$$

*Then there is an inequality $\alpha^T\overline{x} \leq \beta$ that separates $\mathcal{T}$ from $\overline{x}^*$, induces a facet of the convex hull of $\mathcal{T}$, and satisfies*

$$\{\overline{x} \in \mathcal{T} : a^T\overline{x} = b\} \subseteq \{\overline{x} \in \mathcal{T} : \alpha^T\overline{x} = \beta\}.$$

Success of Algorithm 71 hinges on the ability to make choices of $\phi$ and $\mathcal{T}$ (which may be guided by $x^*$ and $\mathcal{S}$) in such a way that

(i) chances of $\phi(x^*)$ falling outside the convex hull of $\mathcal{T}$ are reasonable and

(ii) cuts $a^T\phi(x) \leq b$ collected in $\mathcal{L}$ are not too weak.

Our way (described in Sect. 5) of making these choices in the special case where $\mathcal{S}$ is the set of all tours through a set $V$ meets both of these criteria. With respect to (i), it is adequate: typically, one out of fifty to a hundred of our choices of $\phi$ and $\mathcal{T}$ makes $\phi(x^*)$ fall outside the convex hull of $\mathcal{T}$. With respect to (ii), it could hardly be better: there is no known counterexample to the conjecture (implicit in Naddef and Rinaldi [53]) that

*$a^T\phi(x) \leq b$ induces a facet of the convex hull of $\mathcal{S}$*
*whenever $a^T\overline{x} \leq b$ induces a facet of the convex hull of $\mathcal{T}$ and $a^T\phi(x^*) > b$.*

This may be a major reason behind the success of our application of Algorithm 71 to the traveling salesman problem.

If one were to use Algorithm 71 in the context of integer linear programming, where $\mathcal{S}$ consists of all integer solutions of some explicitly recorded system

$$Ax = b, \ \ell \leq x \leq u,$$

then one would have to design a way of making choices of $\phi$ and $\mathcal{T}$. One option is to choose each $\phi$ and $\mathcal{T}$ by choosing a $d \times m$ integer matrix $P$, setting $\phi(x) = Px$, and letting $\mathcal{T}$ consist of all integer vectors $\overline{x}$ such that some $x$ satisfies

$$Ax = b,\ \ell \leq x \leq u, Px = \overline{x}.$$

Maximizing a linear function over $\mathcal{T}$ amounts to solving a mixed integer linear programming problem in $d$ integer and $m$ non-integer variables; as long as $d$ is small, this problem can be solved quickly. As long as at least one row $r^T$ of $P$ satisfies

$$\lfloor \max\{r^T x : Ax = b,\ \ell \leq x \leq u\} \rfloor < r^T x^*, \tag{33}$$

$\phi(x^*)$ falls outside the convex hull of $\mathcal{T}$; Gomory's methods mentioned in Sect. 2 provide vectors $r^T$ with property (33) at insignificant computational cost. We have not carried out any experiments with this scheme.

**Acknowledgments.** We wish to thank Adrian Bondy, Martin Farach-Colton, Denis Naddef, Yves Pochet, and Günter Rote for their helpful comments on earlier versions of this manuscript.

## References

1. Applegate, D., Bixby, R., Chvátal, V., Cook, W.: Finding cuts in the TSP (A preliminary report). DIMACS Technical Report 95-05, 1995. Available at `ftp://dimacs.rutgers.edu/pub/dimacs/TechnicalReports/TechReports/1995/`
2. Applegate, D., Bixby, R., Chvátal, V., Cook, W.: On the solution of traveling salesman problems. Documenta Mathematica **Extra Volume** (Proceedings of the International Congress of Mathematicians), 645–656, 1998. Also available at `http://www.mathematik.uni-bielefeld.de/documenta/xvol-icm/17/17.html`
3. Applegate, D., Bixby, R., Chvátal, V., Cook, W.: Concorde, 1999. Available at `http://www.math.princeton.edu/tsp/concorde.html`
4. Applegate, D., Bixby, R., Chvátal, V., Cook, W.: Solving Traveling Salesman Problems. To appear.
5. Balas, E.: Facets of the knapsack polytope. Mathematical Programming **8**, 146–164, 1975.
6. Batoukov, R., Sørevik, T.: A generic parallel branch and bound environment on a network of workstations. In: Proceedings of HiPer'99, pp. 474–483, 1999. Also available at `http://www.ii.uib.no/~tors/publications/`
7. Bock, F.: An algorithm for solving 'traveling-salesman' and related network optimization problems. Research Report, Armour Research Foundation. Presented at the Operations Research Society of America Fourteenth National Meeting, St. Louis, October 24, 1958.
8. Boyd, E.A.: Generating Fenchel cutting planes for knapsack polyhedra. SIAM Journal of Optimization **3**, 734–750, 1993.
9. Boyd, E.A.: Fenchel cutting planes for integer programs. Operations Research **42**, 53–64, 1994.

10. Carr, R.: Separating clique trees and bipartition inequalities having a fixed number of handles and teeth in polynomial time. Mathematics of Operations Research **22**, 257–265, 1997.
11. Christof, T., Reinelt, G.: Parallel cutting plane generation for the TSP. In: Parallel Programming and Applications (P. Fritzson and L. Finmo, eds.), IOS Press, pp. 163–169, 1995.
12. Christof, T., Reinelt, G.: Combinatorial optimization and small polytopes. Top **4**, 1–64, 1996.
13. Clochard, J.-M., Naddef, D.: Using path inequalities in a branch and cut code for the symmetric traveling salesman problem. In: Third IPCO Conference, (G. Rinaldi and L. Wolsey, eds.), pp. 291–311, 1993.
14. Cornuéjols, G., Fonlupt, J., Naddef, D.: The traveling salesman problem on a graph and some related integer polyhedra. Mathematical Programming **33**, 1–27, 1985.
15. Craven, B.D.: Fractional Programming. Heldermann, Berlin, 1988.
16. Croes, G.A.: A method for solving traveling-salesman problems. Operations Research **6**, 791–812, 1958.
17. Crowder, H., Johnson, E.L., Padberg, M.: Solving large-scale zero-one linear programming problems. Operations Research **31**, 803–834, 1983.
18. Crowder, H., Padberg, M.W.: Solving large-scale symmetric travelling salesman problems to optimality. Management Science **26**, 495–509, 1980.
19. Dantzig, G., Fulkerson, R., Johnson, S.: Solution of a large-scale traveling salesman problem. Operations Research **2**, 393–410, 1954.
20. Eastman, W.L.: Linear programming with pattern constraints. Ph.D. Thesis, Harvard University, 1958.
21. Edmonds, J.: Maximum matching and a polyhedron with 0,1-vertices. Journal of Research of the National Bureau of Standards **69B**, 125–130, 1965.
22. Fleischer, L.K., Tardos, É.: Separating Maximally Violated Combs in Planar Graphs. Mathematics of Operations Research **24**, 130–148, 1999.
23. Fleischmann, B.: A cutting plane procedure for the travelling salesman problem on road networks. European Journal of Operational Research **21**, 307–317, 1985.
24. Fleischmann, B.: A new class of cutting planes for the symmetric travelling salesman problem. Mathematical Programming **40**, 225–246, 1988.
25. Ford, L.R.Jr., Fulkerson, D.R.: A suggested computation for maximal multicommodity networks flows. Management Science **5**, 97–101, 1958.
26. Gomory, R.E.: Outline of an algorithm for integer solutions to linear programs. Bulletin of the American Mathematical Society **64**, 275–278, 1958.
27. Gomory, R.E.: Solving linear programs in integers. In: Combinatorial Analysis (R. E. Bellman and M. Hall, Jr., eds.), Proceedings of the Symposia on Applied Mathematics **X**, pp. 211–216, 1960.
28. Gomory, R.E.: An algorithm for integer solutions to linear programs. In: Recent Advances in Mathematical Programming (R. L. Graves and P. Wolfe, eds.), McGraw-Hill, New York, pp. 269–302, 1963.
29. Gomory, R.E.: Some polyhedra related to combinatorial problems. Linear Algebra and Its Applications **2**, 451–558, 1969.
30. Grötschel, M.: Polyedrische Charakterisierungen kombinatorischer Optimierungsprobleme, Anton Hain Verlag, Meisenheim/Glan, 1977.
31. Grötschel, M.: On the symmetric travelling salesman problem: solution of a 120-city problem. Mathematical Programming Study **12**, 61–77, 1980.
32. Grötschel, M., O. Holland.: Solution of large-scale symmetric travelling salesman problems. Mathematical Programming **51**, 141–202, 1991.

33. Grötschel, M., Jünger, M., Reinelt, G.: A cutting plane algorithm for the linear ordering problem. Operations Research **32**, 1195–1220, 1984.
34. Grötschel, M., Padberg, M.W.: On the Symmetric Travelling Salesman Problem, Report No.7536-OR, Institut für Ökonometrie und Operations Research, Universität Bonn, 1975.
35. Grötschel, M., Padberg, M.W.: On the symmetric travelling salesman problem I: Inequalities. Mathematical Programming **16**, 265–280, 1979.
36. Grötschel, M., Padberg, M.W.: On the symmetric travelling salesman problem II: Lifting theorems and facets. Mathematical Programming **16**, 281–302, 1979.
37. Grötschel, M., Pulleyblank, W.: Clique tree inequalities and the symmetric travelling salesman problem. Mathematics of Operations Research **11**, 537–569, 1986.
38. Hammer, P.L., Johnson, E.L., Peled, U.N.: Facets of regular 0-1 polytopes. Mathematical Programming **8**, 179–206, 1975.
39. Hong, S.: A linear programming approach for the traveling salesman problem, Ph.D. Thesis, The Johns Hopkins University, 1972.
40. Jewell, W.S.: Optimal flow through networks. Interim Technical Report No. 8, Massachusetts Institute of Technology, 1958.
41. Land, A.: The solution of some 100-city travelling salesman problems. Unpublished manuscript, 1979.
42. Land, A.H., Doig, A.G.: An automatic method of solving discrete programming problems. Econometrica **28**, 497–520, 1960.
43. Letchford, A.N.: Separating a superclass of comb inequalities in planar graphs. Mathematics of Operations Research **25**, 443–454, 2000.
44. Little, J.D.C., Murty, K.G., Sweeney, D.W., Karel, C.: An algorithm for the traveling salesman problem. Operations Research **11**, 972–989, 1963.
45. Martin, G.T.: An accelerated euclidean algorithm for integer linear programming. In: Recent advances in mathematical programming (R. L. Graves and P. Wolfe, eds.), McGraw-Hill, pp. 311–318, 1963.
46. Martin, G.T.: Solving the traveling salesman problem by integer linear programming. Operations Research **14** (Supplement 1), Abstract WA7.10, 1966.
47. Maurras, J.F.: Some results on the convex hull of Hamiltonian cycles of symmetric complete graphs. In: Combinatorial Programming: Methods and Applications (B. Roy, ed.), Reidel, Dordrecht, pp. 179–190, 1975.
48. Miliotis, P.: Integer programming approaches to the travelling salesman problem. Mathematical Programming **10**, 367–378, 1976.
49. Miliotis, P.: Using cutting planes to solve the symmetric travelling salesman problem. Mathematical Programming **15**, 177–188, 1978.
50. Minkowski, H.: Geometrie der Zahlen (Erste Lieferung). Teubner, Leipzig, 1896. Reprinted: Chelsea, New York, 1953.
51. Naddef, D.: Handles and teeth in the symmetric traveling salesman polytope. In: Polyhedral combinatorics (W. Cook and P. D. Seymour, eds.), DIMACS Series in Mathematics and Theoretical Computer Science **1**, American Mathematical Society, pp. 61–74, 1990.
52. Naddef, D., Rinaldi, G.: The symmetric traveling salesman polytope and its graphical relaxation: Composition of valid inequalities. Mathematical Programming **51**, 359–400, 1991.
53. Naddef, D., Rinaldi, G.: The graphical relaxation: A new framework for the symmetric traveling salesman polytope. Mathematical Programming **58**, 53–88, 1992.

54. Naddef, D., Thienel, S.: Efficient separation routines for the symmetric traveling salesman problem I: General tools and comb separation. Working paper, 1999. Available at
`http://www-id.imag.fr/Laboratoire/Membres/Naddef_Denis/perso.html`
55. Naddef, D., Thienel, S.: Efficient separation routines for the symmetric traveling salesman problem II: Separating multi handle inequalities. Working paper, 1999. Available at
`http://www-id.imag.fr/Laboratoire/Membres/Naddef_Denis/perso.html`
56. Padberg, M.W.: On the facial structure of set packing polyhedra. Mathematical Programming **5**, 199–215, 1973.
57. Padberg, M.W.: A note on zero-one programming. Operations Research **23**, 833–837, 1975.
58. Padberg, M.W., Hong, S.: On the symmetric travelling salesman problem: a computational study. Mathematical Programming Study **12**, 78–107, 1980.
59. Padberg, M.W., Rao, M.R.: Odd minimum cut-sets and $b$-matchings. Mathematics of Operations Research **7**, 67–80, 1982.
60. Padberg, M.W., Rinaldi, G.: Optimization of a 532-city symmetric traveling salesman problem by branch and cut. Operations Research Letters **6**, 1–7, 1987.
61. Padberg, M.W., Rinaldi, G.: An efficient algorithm for the minimum capacity cut problem. Mathematical Programming **47**, 1990.
62. Padberg, M.W., Rinaldi, G.: Facet identification for the symmetric traveling salesman polytope. Mathematical Programming **47**, 219–257, 1990.
63. Padberg, M.W., Rinaldi, G.: A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems. SIAM Review **33**, 60–100, 1991.
64. Reinelt, G.: TSPLIB – A traveling salesman problem library. ORSA Journal on Computing **3**, 376–384, 1991. An updated version is available at
`http://www.iwr.uni-heidelberg.de/iwr/comopt/software/ TSPLIB95/`.
65. Rossman, M.J., Twery, R.J.: A solution to the travelling salesman problem. Operations Research **6**, p.687, Abstract E3.1.3, 1958.
66. Schrijver, A.: Theory of Linear and Integer Programming. Wiley, Chichester, 1986.
67. Stancu-Minasian, I.M.: Fractional Programming. Kluwer, Dordrecht, 1997.
68. Wolsey, L.A.: Faces for a linear inequality in 0-1 variables. Mathematical Programming **8**, 165–178, 1975.
69. Wolsey, L.A.: Facets and strong valid inequalities for integer programs. Operations Research **24**, 367–372, 1975.