LINEAR PROGRAMMING

[V. CH10]: APPLICATION: TSP

Phillip Keldenich   Ahmad Moradi

Department of Computer Science
Algorithms Department
TU Braunschweig

February 6, 2024

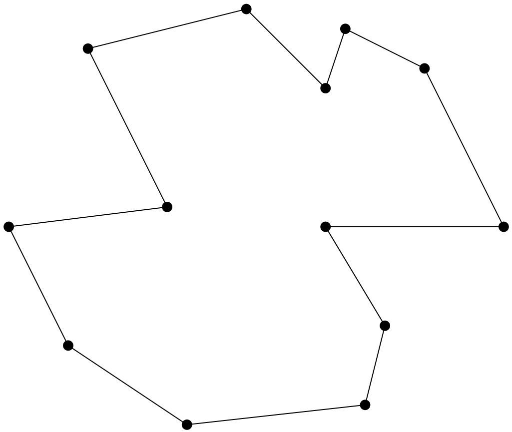
DEFINITION & MODEL

CUTTING PLANES

BRANCH, CUT & PRICE

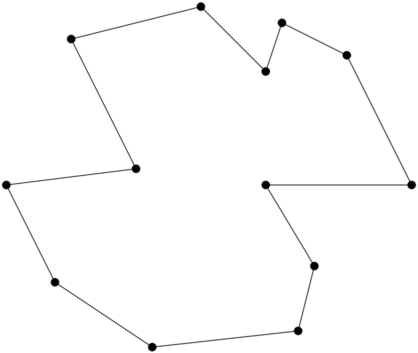
Traveling Salesman Problem

For a given set V of n cities, (sometimes also called vertices) with given costs $c_{ab} = c_{ba}$ for going from any city a to any city b, compute the shortest round trip through all cities, visiting each city exactly once.



Integer Programming Model

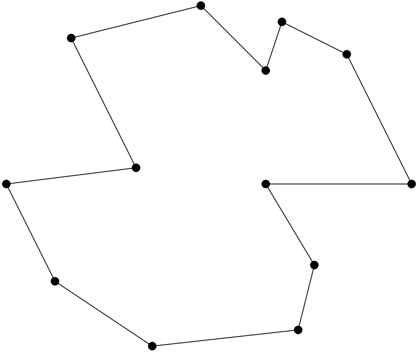
How can we model this as an integer program?



INTEGER PROGRAMMING MODEL

How can we model this as an integer program?

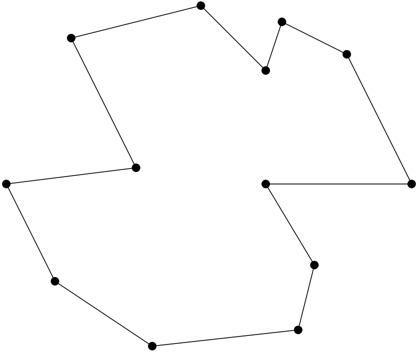
- One variable $x_{vw} \in \{0,1\}$ per undirected edge $\{v,w\} = vw \in E = \binom{V}{2}$.
- $\min \sum c_e x_e$



INTEGER PROGRAMMING MODEL

How can we model this as an integer program?

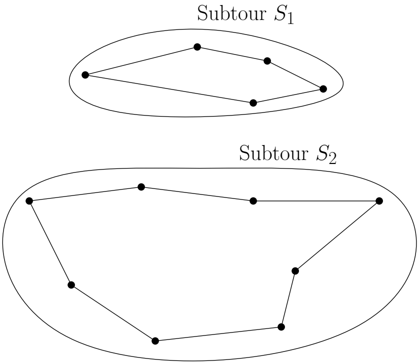
- One variable $x_{vw} \in \{0, 1\}$ per undirected edge $\{v, w\} = vw \in E = \binom{V}{2}$.
- $\min \sum c_e x_e$
- One constraint per city $v \in V$: $\sum_{e \in \delta(\{v\})} x_e = 2$.



Integer Programming Model

How can we model this as an integer program?

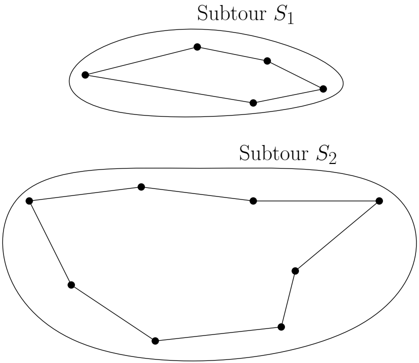
- One variable $x_{vw} \in \{0,1\}$ per undirected edge $\{v,w\} = vw \in E = \binom{V}{2}$.
- $\min \sum c_e x_e$
- One constraint per city $v \in V$: $\sum_{e \in \delta(\{v\})} x_e = 2$.



INTEGER PROGRAMMING MODEL

How can we model this as an integer program?

- One variable $x_{vw} \in \{0,1\}$ per undirected edge $\{v,w\} = vw \in E = \binom{V}{2}$.
- $\min \sum c_e x_e$
- One constraint per city $v \in V$: $\sum_{e \in \delta(\{v\})} x_e = 2$.
- *Subtour elimination* constraints: $\forall S \subsetneq V, S \neq \emptyset : \sum_{e \in \delta(S)} x_e \geq 2$.



About the Model

- This model is called the Dantzig or Dantzig-Fulkerson-Johnson formulation of the TSP.

About the Model

- This model is called the Dantzig or Dantzig-Fulkerson-Johnson formulation of the TSP.
- As written, the model has exponentially many constraints!

ABOUT THE MODEL

- This model is called the Dantzig or Dantzig-Fulkerson-Johnson formulation of the TSP.
- As written, the model has exponentially many constraints!
- We do not add the subtour elimination constraints at the beginning.

About the Model

- This model is called the Dantzig or Dantzig-Fulkerson-Johnson formulation of the TSP.
- As written, the model has exponentially many constraints!
- We do not add the subtour elimination constraints at the beginning.
- Instead, we generate them *lazily* (lazy constraints) when we find solutions that violate them.

About the Model

- This model is called the Dantzig or Dantzig-Fulkerson-Johnson formulation of the TSP.
- As written, the model has exponentially many constraints!
- We do not add the subtour elimination constraints at the beginning.
- Instead, we generate them *lazily* (lazy constraints) when we find solutions that violate them.
- There are alternative models with polynomial size, but they usually perform much worse.

ABOUT THE MODEL

- This model is called the Dantzig or Dantzig-Fulkerson-Johnson formulation of the TSP.
- As written, the model has exponentially many constraints!
- We do not add the subtour elimination constraints at the beginning.
- Instead, we generate them *lazily* (lazy constraints) when we find solutions that violate them.
- There are alternative models with polynomial size, but they usually perform much worse.
- When embedded in a Branch & Cut algorithm, lazy constraints are added after solving a linear relaxation.

About the Model

- This model is called the Dantzig or Dantzig-Fulkerson-Johnson formulation of the TSP.
- As written, the model has exponentially many constraints!
- We do not add the subtour elimination constraints at the beginning.
- Instead, we generate them *lazily* (lazy constraints) when we find solutions that violate them.
- There are alternative models with polynomial size, but they usually perform much worse.
- When embedded in a Branch & Cut algorithm, lazy constraints are added after solving a linear relaxation.
- This is very similar to cutting plane generation (which is part of the algorithm anyways).

About the Model

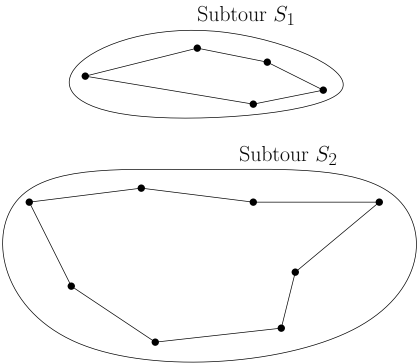
- This model is called the Dantzig or Dantzig-Fulkerson-Johnson formulation of the TSP.
- As written, the model has exponentially many constraints!
- We do not add the subtour elimination constraints at the beginning.
- Instead, we generate them *lazily* (lazy constraints) when we find solutions that violate them.
- There are alternative models with polynomial size, but they usually perform much worse.
- When embedded in a Branch & Cut algorithm, lazy constraints are added after solving a linear relaxation.
- This is very similar to cutting plane generation (which is part of the algorithm anyways).
- How hard is it to *separate* subtours?

Separating Subtour Constraints

Integral:

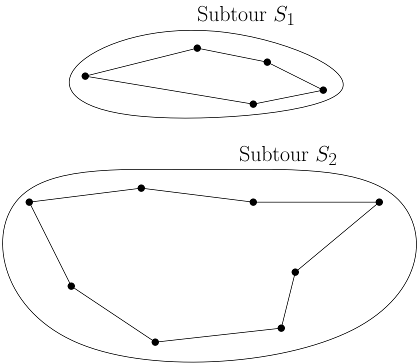
Separating Subtour Constraints

Integral:



SEPARATING SUBTOUR CONSTRAINTS

Integral: Easy (BFS/DFS): $\sum_{e\in\delta(S_1)} x_e = 0 < 2 \rightarrow$ add (violated) constraint $\sum_{e\in\delta(S_1)} x_e \geq 2$

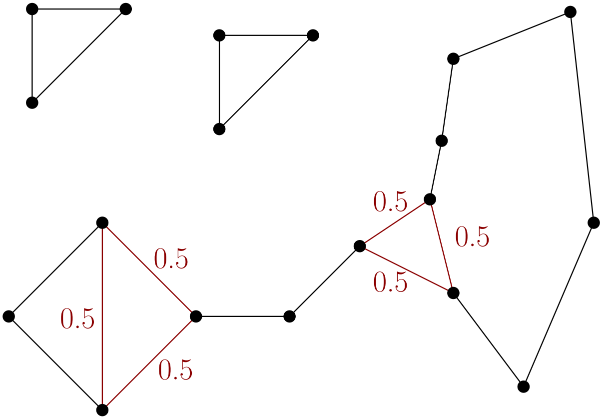


Separating Subtour Constraints

Fractional:

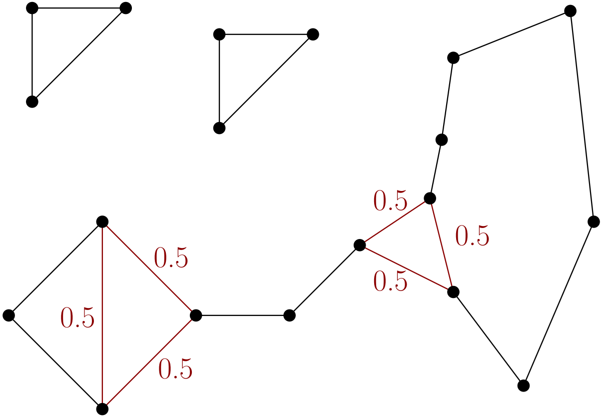
Separating Subtour Constraints

Fractional:



Separating Subtour Constraints

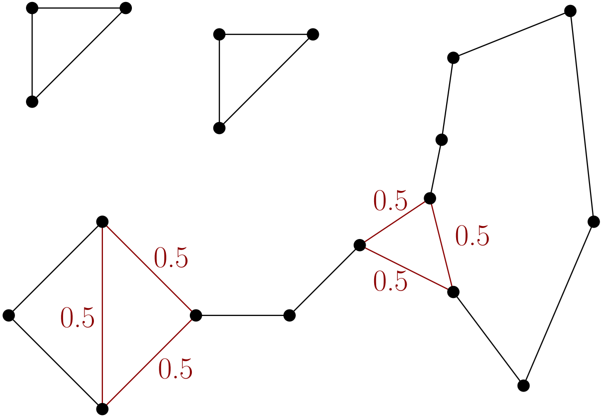
Fractional:



Algorithms: BFS/DFS connected components, biconnected components (DFS-style).

Separating Subtour Constraints

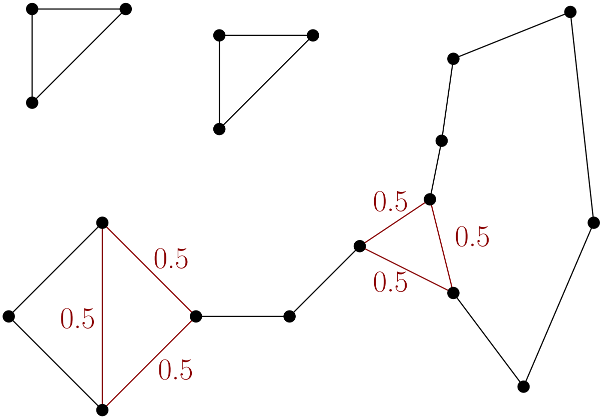
Fractional:



Algorithms: BFS/DFS connected components, biconnected components (DFS-style).
Exact separation?

Separating Subtour Constraints

Fractional:



Algorithms: BFS/DFS connected components, biconnected components (DFS-style).
Exact separation? Minimum graph cut (Stoer-Wagner algorithm).

Minimum Graph Cut

Given a graph $G = (V, E)$ with weighted undirected edges $w(e) \geq 0$, find a partition

$V = S \cup T, S \cap T = \emptyset, S, T \neq \emptyset$, which minimizes

$$\sum_{e \in \delta(S)} w(e).$$

MINIMUM GRAPH CUT

Given a graph $G = (V, E)$ with weighted undirected edges $w(e) \geq 0$, find a partition

$$V = S \cup T, S \cap T = \emptyset, S, T \neq \emptyset, \text{ which minimizes}$$

$$\sum_{e \in \delta(S)} w(e).$$

- In our case, we have $w(e) = x_e$, and we drop edges with $x_e = 0$.

MINIMUM GRAPH CUT

Given a graph $G = (V, E)$ with weighted undirected edges $w(e) \geq 0$, find a partition

$$V = S \cup T, S \cap T = \emptyset, S, T \neq \emptyset, \text{ which minimizes}$$

$$\sum_{e \in \delta(S)} w(e).$$

- In our case, we have $w(e) = x_e$, and we drop edges with $x_e = 0$.
- The algorithm of Stoer & Wagner solves this problem in $O(|V||E| + |V|^2 \log |V|)$.

Minimum Graph Cut

Given a graph $G = (V, E)$ with weighted undirected edges $w(e) \geq 0$, find a partition

$V = S \cup T, S \cap T = \emptyset, S, T \neq \emptyset$, which minimizes

$$\sum_{e \in \delta(S)} w(e).$$

- In our case, we have $w(e) = x_e$, and we drop edges with $x_e = 0$.
- The algorithm of Stoer & Wagner solves this problem in $O(|V||E| + |V|^2 \log |V|)$.
- If the minimum cut is strictly below 2, S and T are vertex sets of violated subtour constraints.

Minimum Graph Cut

Given a graph $G = (V, E)$ with weighted undirected edges $w(e) \geq 0$, find a partition

$$V = S \cup T, S \cap T = \emptyset, S, T \neq \emptyset, \text{ which minimizes}$$

$$\sum_{e \in \delta(S)} w(e).$$

- In our case, we have $w(e) = x_e$, and we drop edges with $x_e = 0$.
- The algorithm of Stoer & Wagner solves this problem in $O(|V||E| + |V|^2 \log |V|)$.
- If the minimum cut is strictly below 2, S and T are vertex sets of violated subtour constraints.
- The high running time may not be worth it — usually, at the very least, one should run the cheaper methods first.

Example Time

Interactive example at https://www.math.uwaterloo.ca/tsp/app/diy.html.

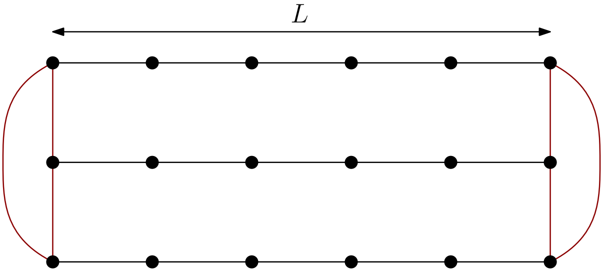
Definition & Model

Cutting Planes

Branch, Cut & Price

Cutting Planes for Tours

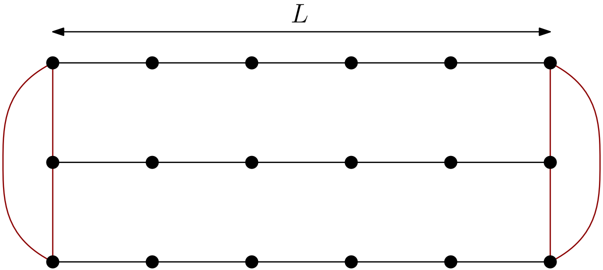
Besides subtours, what else are good cutting planes for the TSP?



Notes:

Cutting Planes for Tours

Besides subtours, what else are good cutting planes for the TSP?

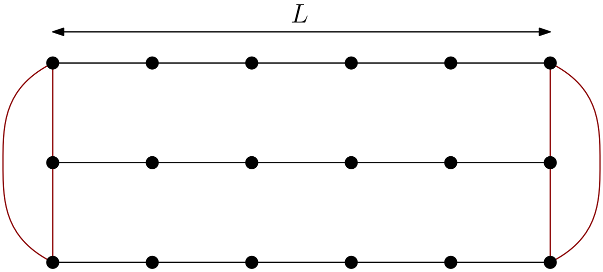


Notes:

- All subtour constraints are satisfied.

CUTTING PLANES FOR TOURS

Besides subtours, what else are good cutting planes for the TSP?

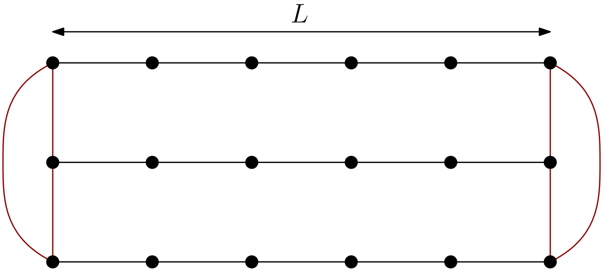


Notes:

- All subtour constraints are satisfied.
- For $L \to \infty$, the relaxation solution is $3L + 4 \approx 3L$.

Cutting Planes for Tours

Besides subtours, what else are good cutting planes for the TSP?

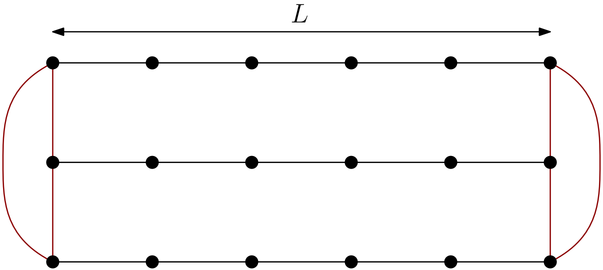


Notes:

- All subtour constraints are satisfied.
- For $L \to \infty$, the relaxation solution is $3L + 4 \approx 3L$.
- For $L \to \infty$, the optimal integral solution is $4L + \varepsilon \approx 4L$ ($\varepsilon = 0$ for even L).

Cutting Planes for Tours

Besides subtours, what else are good cutting planes for the TSP?

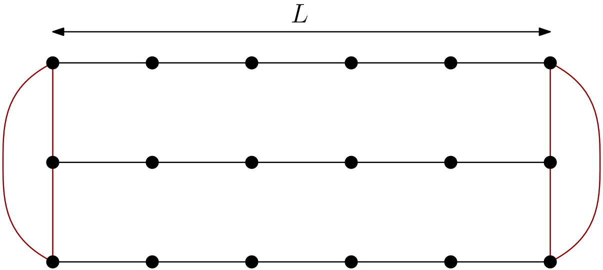


Notes:

- All subtour constraints are satisfied.
- For $L \to \infty$, the relaxation solution is $3L + 4 \approx 3L$.
- For $L \to \infty$, the optimal integral solution is $4L + \varepsilon \approx 4L$ ($\varepsilon = 0$ for even L).
- The *(asymptotic) integrality gap* of this TSP formulation (with all subtours) is at least $4/3$.

CUTTING PLANES FOR TOURS

Besides subtours, what else are good cutting planes for the TSP?

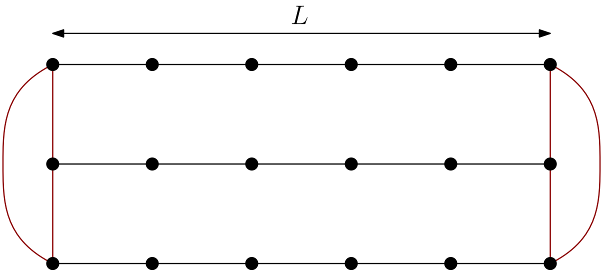


Notes:

- All subtour constraints are satisfied.
- For $L \to \infty$, the relaxation solution is $3L + 4 \approx 3L$.
- For $L \to \infty$, the optimal integral solution is $4L + \varepsilon \approx 4L$ ($\varepsilon = 0$ for even L).
- The *(asymptotic) integrality gap* of this TSP formulation (with all subtours) is at least 4/3.
- 4/3-conjecture: This is actually the integrality gap, i.e., there are no worse instances than this.

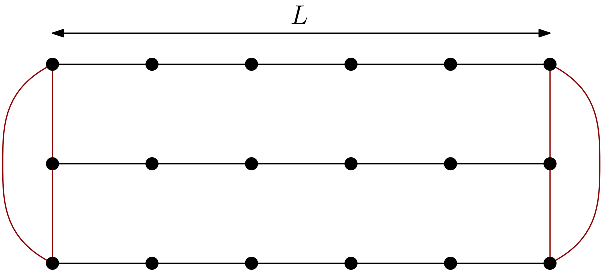
Comb Inequalities

How do we find cuts for this solution?



Comb Inequalities

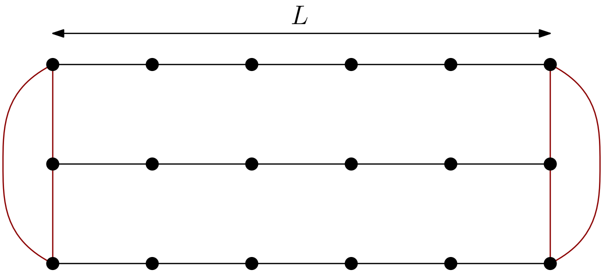
How do we find cuts for this solution?



Intuition: We have to leave each side 2 or 4 times, not 3 times!

Comb Inequalities

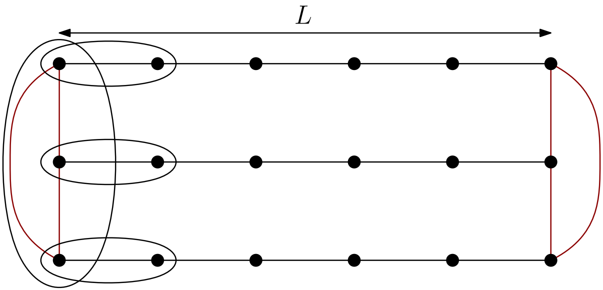
How do we find cuts for this solution?



Intuition: We have to leave each side 2 or 4 times, not 3 times!
Translation to a *single valid inequality*: not easy to see!

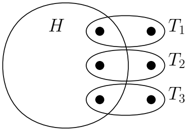
COMB INEQUALITIES

How do we find cuts for this solution?



Intuition: We have to leave each side 2 or 4 times, not 3 times!
Translation to a *single valid inequality*: not easy to see!

Comb Inequalities



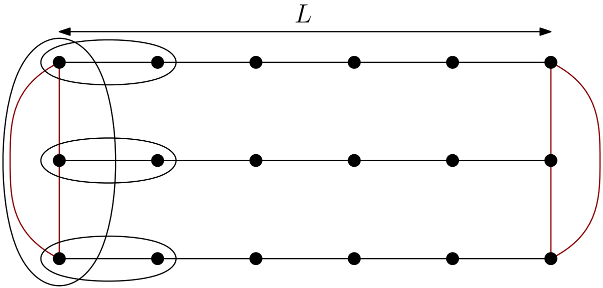
Suppose we have $H, T_1, \ldots, T_k \subset V$:

- $\forall i \in \{1, \ldots, k\} : H \cap T_i \neq \emptyset$ (handle meets each tooth),
- $\forall i \in \{1, \ldots, k\} : T_i \setminus H \neq \emptyset$ (teeth have vertex outside handle),
- $\forall i \neq j \in \{1, \ldots, k\} : T_i \cap T_j = \emptyset$ (teeth are disjoint),
- k is odd,

then every valid tour has

$$\sum_{e \in \delta(H)} x_e + \sum_{i=1}^{k} \sum_{e \in \delta(T_i)} x_e \geq 3k + 1.$$

Comb Inequalities: Our Example



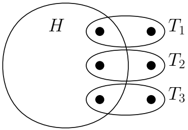
Here, we have:

$$\sum_{e\in\delta(H)} x_e = 3, \sum_{e\in\delta(T_i)} x_e = 2,$$

$$\sum_{e\in\delta(H)} x_e + \sum_{i=1}^{k} \sum_{e\in\delta(T_i)} x_e = 3 + 3\cdot 2 = 9 < 10 = 3k+1,$$

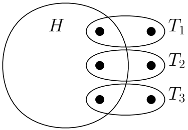
thus this comb inequality is a violated cutting plane!

Comb Inequalities: Correctness Proof



Let $H, T_1, \dots, T_k$ be a comb, and let $\mathcal{S} = \sum_{e \in \delta(H)} x_e + \sum_{i=1}^{k} \sum_{e \in \delta(T_i)} x_e$.

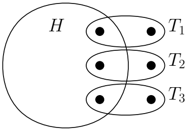
Comb Inequalities: Correctness Proof



Let $H, T_1, \ldots, T_k$ be a comb, and let $\mathcal{S} = \sum_{e \in \delta(H)} x_e + \sum_{i=1}^{k} \sum_{e \in \delta(T_i)} x_e$.

First, consider a single T_i and some tour R through all cities.

Comb Inequalities: Correctness Proof

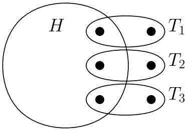


Let $H, T_1, \ldots, T_k$ be a comb, and let $\mathcal{S} = \sum_{e \in \delta(H)} x_e + \sum_{i=1}^{k} \sum_{e \in \delta(T_i)} x_e$.

First, consider a single T_i and some tour R through all cities.

- Both $H \cap T_i$ and $T_i \setminus H$ are nonempty!

Comb Inequalities: Correctness Proof

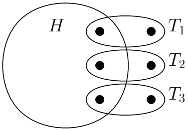


Let $H, T_1, \ldots, T_k$ be a comb, and let $\mathcal{S} = \sum_{e \in \delta(H)} x_e + \sum_{i=1}^{k} \sum_{e \in \delta(T_i)} x_e$.

First, consider a single T_i and some tour R through all cities.

- Both $H \cap T_i$ and $T_i \setminus H$ are nonempty!
- Therefore, the tour has to enter and exit each such set at least once.

COMB INEQUALITIES: CORRECTNESS PROOF

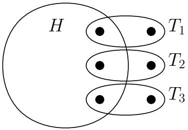


Let $H, T_1, \ldots, T_k$ be a comb, and let $\mathcal{S} = \sum_{e \in \delta(H)} x_e + \sum_{i=1}^{k} \sum_{e \in \delta(T_i)} x_e$.

First, consider a single T_i and some tour R through all cities.

- Both $H \cap T_i$ and $T_i \setminus H$ are nonempty!
- Therefore, the tour has to enter and exit each such set at least once.
- If R contains an edge e_i from $H \cap T_i$ to $H \setminus T_i$:

Comb Inequalities: Correctness Proof

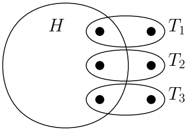


Let $H, T_1, \dots, T_k$ be a comb, and let $\mathcal{S} = \sum_{e \in \delta(H)} x_e + \sum_{i=1}^{k} \sum_{e \in \delta(T_i)} x_e$.

First, consider a single T_i and some tour R through all cities.

- Both $H \cap T_i$ and $T_i \setminus H$ are nonempty!
- Therefore, the tour has to enter and exit each such set at least once.
- If R contains an edge e_i from $H \cap T_i$ to $H \setminus T_i$:
    - Edge e_i contributes 1 to the sum $\mathcal{S}$ (crosses H).

COMB INEQUALITIES: CORRECTNESS PROOF

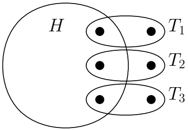


Let $H, T_1, \dots, T_k$ be a comb, and let $\mathcal{S} = \sum_{e \in \delta(H)} x_e + \sum_{i=1}^{k} \sum_{e \in \delta(T_i)} x_e$.

First, consider a single T_i and some tour R through all cities.

- Both $H \cap T_i$ and $T_i \setminus H$ are nonempty!
- Therefore, the tour has to enter and exit each such set at least once.
- If R contains an edge e_i from $H \cap T_i$ to $H \setminus T_i$:
    - Edge e_i contributes 1 to the sum $\mathcal{S}$ (crosses H).
    - Furthermore, e_i does not cross T_i, so there must be two more edges in R crossing T_i.

COMB INEQUALITIES: CORRECTNESS PROOF

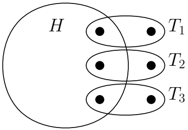


Let $H, T_1, \dots, T_k$ be a comb, and let $\mathcal{S} = \sum_{e \in \delta(H)} x_e + \sum_{i=1}^{k} \sum_{e \in \delta(T_i)} x_e$.

First, consider a single T_i and some tour R through all cities.

- Both $H \cap T_i$ and $T_i \setminus H$ are nonempty!
- Therefore, the tour has to enter and exit each such set at least once.
- If R contains an edge e_i from $H \cap T_i$ to $H \setminus T_i$:
  - Edge e_i contributes 1 to the sum $\mathcal{S}$ (crosses H).
  - Furthermore, e_i does not cross T_i, so there must be two more edges in R crossing T_i.
  - T_i contributes at least 3 to the sum $\mathcal{S}$.

COMB INEQUALITIES: CORRECTNESS PROOF

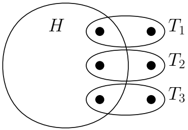


Let $H, T_1, \ldots, T_k$ be a comb, and let $\mathcal{S} = \sum_{e \in \delta(H)} x_e + \sum_{i=1}^{k} \sum_{e \in \delta(T_i)} x_e$.

First, consider a single T_i and some tour R through all cities.

- Both $H \cap T_i$ and $T_i \setminus H$ are nonempty!
- Therefore, the tour has to enter and exit each such set at least once.
- If R contains an edge e_i from $H \cap T_i$ to $H \setminus T_i$:
  - Edge e_i contributes 1 to the sum $\mathcal{S}$ (crosses H).
  - Furthermore, e_i does not cross T_i, so there must be two more edges in R crossing T_i.
  - T_i contributes at least 3 to the sum $\mathcal{S}$.
- Otherwise, R enters and leaves T_i at least twice and T_i contributes at least 4 to $\mathcal{S}$.

COMB INEQUALITIES: CORRECTNESS PROOF

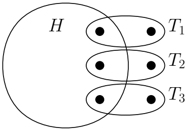


Let $H, T_1, \dots, T_k$ be a comb, and let $\mathcal{S} = \sum_{e \in \delta(H)} x_e + \sum_{i=1}^{k} \sum_{e \in \delta(T_i)} x_e$.

First, consider a single T_i and some tour R through all cities.

- Both $H \cap T_i$ and $T_i \setminus H$ are nonempty!
- Therefore, the tour has to enter and exit each such set at least once.
- If R contains an edge e_i from $H \cap T_i$ to $H \setminus T_i$:
  - Edge e_i contributes 1 to the sum $\mathcal{S}$ (crosses H).
  - Furthermore, e_i does not cross T_i, so there must be two more edges in R crossing T_i.
  - T_i contributes at least 3 to the sum $\mathcal{S}$.
- Otherwise, R enters and leaves T_i at least twice and T_i contributes at least 4 to $\mathcal{S}$.
- Because teeth are disjoint, we can sum up these contributions: $\mathcal{S} \geq 3k$.

Comb Inequalities: Correctness Proof

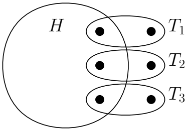


Let $H, T_1, \dots, T_k$ be a comb, and let $\mathcal{S} = \sum_{e \in \delta(H)} x_e + \sum_{i=1}^{k} \sum_{e \in \delta(T_i)} x_e$.

First, consider a single T_i and some tour R through all cities.

- Both $H \cap T_i$ and $T_i \setminus H$ are nonempty!
- Therefore, the tour has to enter and exit each such set at least once.
- If R contains an edge e_i from $H \cap T_i$ to $H \setminus T_i$:
  - Edge e_i contributes 1 to the sum $\mathcal{S}$ (crosses H).
  - Furthermore, e_i does not cross T_i, so there must be two more edges in R crossing T_i.
  - T_i contributes at least 3 to the sum $\mathcal{S}$.
- Otherwise, R enters and leaves T_i at least twice and T_i contributes at least 4 to $\mathcal{S}$.
- Because teeth are disjoint, we can sum up these contributions: $\mathcal{S} \geq 3k$.
- k is odd, so $3k$ is also odd, but $\mathcal{S}$ must be even!

Comb Inequalities: Correctness Proof



Let $H, T_1, \dots, T_k$ be a comb, and let $\mathcal{S} = \sum_{e\in\delta(H)} x_e + \sum_{i=1}^{k} \sum_{e\in\delta(T_i)} x_e$.

First, consider a single T_i and some tour R through all cities.

- Both $H \cap T_i$ and $T_i \setminus H$ are nonempty!
- Therefore, the tour has to enter and exit each such set at least once.
- If R contains an edge e_i from $H \cap T_i$ to $H \setminus T_i$:
  - Edge e_i contributes 1 to the sum $\mathcal{S}$ (crosses H).
  - Furthermore, e_i does not cross T_i, so there must be two more edges in R crossing T_i.
  - T_i contributes at least 3 to the sum $\mathcal{S}$.
- Otherwise, R enters and leaves T_i at least twice and T_i contributes at least 4 to $\mathcal{S}$.
- Because teeth are disjoint, we can sum up these contributions: $\mathcal{S} \geq 3k$.
- k is odd, so $3k$ is also odd, but $\mathcal{S}$ must be even!

$\Rightarrow$ $\mathcal{S} \geq 3k + 1$.

Example Time: Combs

Interactive example at https://www.math.uwaterloo.ca/tsp/app/diy.html.
Random seed: 1234, 50 cities.
Optimal solution (through comb and subtour cuts only): 51991.

Comb Separation

It is not known whether separating combs is NP-hard or in P.

COMB SEPARATION

It is not known whether separating combs is NP-hard or in P.

In practice, this problem is usually solved with heuristics, e.g., based on components spanned by fractional edges (for handles) and edges crossing them (for teeth), like we did in the example.

Comb Separation

It is not known whether separating combs is NP-hard or in P.

In practice, this problem is usually solved with heuristics, e.g., based on components spanned by fractional edges (for handles) and edges crossing them (for teeth), like we did in the example.

For more, see
David L. Applegate, Robert E. Bixby, Vašek Chvatál and William J. Cook.
The Traveling Salesman Problem: A Computational Study.
Princeton Series in Applied Mathematics (2006), Princeton University Press.

Definition & Model

Cutting Planes

Branch, Cut & Price

Pricing: Motivation

Solving TSP instances with this approach works well for relatively large instances.

Pricing: Motivation

Solving TSP instances with this approach works well for relatively large instances.

Going up to 100k cities however runs into one *big* problem:
even keeping a list of all edges takes too much memory!
And we need more space than that per edge!

PRICING: MOTIVATION

Solving TSP instances with this approach works well for relatively large instances.

Going up to 100k cities however runs into one *big* problem:
even keeping a list of all edges takes too much memory!
And we need more space than that per edge!

Solution: Do not consider all edges all the time — most are, after all, never useful!

PRICING: IDEA

Growing (or fully dynamic) edge set E (Branch, Cut & Price):

PRICING: IDEA

Growing (or fully dynamic) edge set E (Branch, Cut & Price):

- Initially, select a candidate set of edges E (e.g., from heuristic tours, nearest neighbors, triangulations, …).

Pricing: Idea

Growing (or fully dynamic) edge set E (Branch, Cut & Price):

- Initially, select a candidate set of edges E (e.g., from heuristic tours, nearest neighbors, triangulations, $\ldots$).
- Solve the relaxation, potentially generating cuts and repeating.

Pricing: Idea

Growing (or fully dynamic) edge set E (Branch, Cut & Price):

- Initially, select a candidate set of edges E (e.g., from heuristic tours, nearest neighbors, triangulations, …).
- Solve the relaxation, potentially generating cuts and repeating.
- *Price:* Check for edges we have missed that improve the relaxation.

Pricing: Idea

Growing (or fully dynamic) edge set E (Branch, Cut & Price):

- Initially, select a candidate set of edges E (e.g., from heuristic tours, nearest neighbors, triangulations, …).
- Solve the relaxation, potentially generating cuts and repeating.
- *Price:* Check for edges we have missed that improve the relaxation.
- Potentially delete edges and cuts that have been inactive for long.

PRICING: IDEA

Growing (or fully dynamic) edge set E (Branch, Cut & Price):

- Initially, select a candidate set of edges E (e.g., from heuristic tours, nearest neighbors, triangulations, ...).
- Solve the relaxation, potentially generating cuts and repeating.
- *Price:* Check for edges we have missed that improve the relaxation.
- Potentially delete edges and cuts that have been inactive for long.

Several challenges:

- Which edges that are not in the LP could yield an improvement?

PRICING: IDEA

Growing (or fully dynamic) edge set E (Branch, Cut & Price):

- Initially, select a candidate set of edges E (e.g., from heuristic tours, nearest neighbors, triangulations, ...).
- Solve the relaxation, potentially generating cuts and repeating.
- *Price:* Check for edges we have missed that improve the relaxation.
- Potentially delete edges and cuts that have been inactive for long.

Several challenges:

- Which edges that are not in the LP could yield an improvement?
- How do cutting planes interact with this?

PRICING

Primal simplex optimality check: $z^*_{\mathcal{N}} \geq 0$.
Equivalently: Check for dual feasibility.

Pricing

Primal simplex optimality check: $z^*_{\mathcal{N}} \geq 0$.
Equivalently: Check for dual feasibility.

With an understanding of our problem and its dual, we can sometimes check for dual feasibility even of variables that were not in the LP!

PRICING

Primal simplex optimality check: $z^*_{\mathcal{N}} \geq 0$.
Equivalently: Check for dual feasibility.

With an understanding of our problem and its dual, we can sometimes check for dual feasibility even of variables that were not in the LP!

This allows us to leave variables out at first and add them only if they look necessary later on.

Pricing

Primal simplex optimality check: $z^*_{\mathcal{N}} \geq 0$.
Equivalently: Check for dual feasibility.

With an understanding of our problem and its dual, we can sometimes check for dual feasibility even of variables that were not in the LP!

This allows us to leave variables out at first and add them only if they look necessary later on.

If all these variables have satisfied dual constraints, our solution is optimal!

Pricing

Primal simplex optimality check: $z^*_{\mathcal{N}} \geq 0$.
Equivalently: Check for dual feasibility.

With an understanding of our problem and its dual, we can sometimes check for dual feasibility even of variables that were not in the LP!

This allows us to leave variables out at first and add them only if they look necessary later on.

If all these variables have satisfied dual constraints, our solution is optimal!

Otherwise, we need to find some variables with violated dual constraints and add them.

Pricing

Primal simplex optimality check: $z^*_{\mathcal{N}} \geq 0$.
Equivalently: Check for dual feasibility.

With an understanding of our problem and its dual, we can sometimes check for dual feasibility even of variables that were not in the LP!

This allows us to leave variables out at first and add them only if they look necessary later on.

If all these variables have satisfied dual constraints, our solution is optimal!

Otherwise, we need to find some variables with violated dual constraints and add them.

Of course, we have to integrate cutting planes into the dual — as dual variables.

Dual of the TSP

$$\min c^T x \text{ s.t.}$$

$$\forall v \in V : \sum_{e \in \delta(\{v\})} x_e = 2,$$

$$\forall S \subsetneq V, S \neq \emptyset : \sum_{e \in \delta(S)} x_e \geq 2.$$

$$0 \leq x_e \leq 1$$

Dualize:

Dual of the TSP

$$\min c^T x \text{ s.t.}$$
$$\forall v \in V : \sum_{e \in \delta(\{v\})} x_e = 2,$$
$$\forall S \subsetneq V, S \neq \emptyset : \sum_{e \in \delta(S)} x_e \geq 2.$$
$$0 \leq x_e \leq 1$$

Dualize:

$$\max 2 \sum_{v \in V} y_v + 2 \sum_{S} z_S - \sum_{e \in E} y_e$$
$$\forall vw \in E : y_v + y_w + \sum_{S : vw \in \delta(S)} z_S - y_e \leq c_e$$
$$y_v \text{ free}, y_e, z_S \geq 0$$

DUAL OF THE TSP

$$\min c^T x \text{ s.t.}$$

$$\forall v \in V : \sum_{e \in \delta(\{v\})} x_e = 2,$$

$$\forall S \subsetneq V, S \neq \emptyset : \sum_{e \in \delta(S)} x_e \geq 2.$$

$$0 \leq x_e \leq 1$$

Dualize:

$$\max 2 \sum_{v \in V} y_v + 2 \sum_{S} z_S - \sum_{e \in E} y_e$$

$$\forall vw \in E : y_v + y_w + \sum_{S : vw \in \delta(S)} z_S - y_e \leq c_e$$

$$y_v \text{ free}, y_e, z_S \geq 0$$

Intuition/Geometry: Zone & Moat packing — see https://www.math.uwaterloo.ca/tsp/app/diy.html.

Cut Representation for the TSP

We represent each cut as follows:

Cut Representation for the TSP

We represent each cut as follows:

- It has a set $\mathcal{F} = \{S_1, \ldots, S_j\}$ of vertex sets $S_i \subset V$.

Cut Representation for the TSP

We represent each cut as follows:

- It has a set $\mathcal{F} = \{S_1, \ldots, S_j\}$ of vertex sets $S_i \subset V$.
- It has a right-hand side value μ.

Cut Representation for the TSP

We represent each cut as follows:

- It has a set $\mathcal{F} = \{S_1, \ldots, S_j\}$ of vertex sets $S_i \subset V$.
- It has a right-hand side value μ.
- The represented inequality is $\sum_{S_i \in \mathcal{F}} \sum_{e \in \delta(\mathcal{F})} x_e \geq \mu$.

Cut Representation for the TSP

We represent each cut as follows:

- It has a set $\mathcal{F} = \{S_1, \ldots, S_j\}$ of vertex sets $S_i \subset V$.
- It has a right-hand side value μ.
- The represented inequality is $\sum_{S_i \in \mathcal{F}} \sum_{e \in \delta(\mathcal{F})} x_e \geq \mu$.

This allows to represent

- subtours: $\mathcal{F} = \{S\}, \mu = 2$,

Cut Representation for the TSP

We represent each cut as follows:

- It has a set $\mathcal{F} = \{S_1, \ldots, S_j\}$ of vertex sets $S_i \subset V$.
- It has a right-hand side value μ.
- The represented inequality is $\sum_{S_i \in \mathcal{F}} \sum_{e \in \delta(\mathcal{F})} x_e \geq \mu$.

This allows to represent

- subtours: $\mathcal{F} = \{S\}, \mu = 2$,
- combs: $\mathcal{F} = \{H, T_1, \ldots, T_k\}, \mu = 3k + 1$,

Cut Representation for the TSP

We represent each cut as follows:

- It has a set $\mathcal{F} = \{S_1, \dots, S_j\}$ of vertex sets $S_i \subset V$.
- It has a right-hand side value μ.
- The represented inequality is $\sum_{S_i \in \mathcal{F}} \sum_{e \in \delta(\mathcal{F})} x_e \geq \mu$.

This allows to represent

- subtours: $\mathcal{F} = \{S\}, \mu = 2$,
- combs: $\mathcal{F} = \{H, T_1, \dots, T_k\}, \mu = 3k + 1$,
- even more advanced cuts (Concorde uses this representation).

Cut Representation for the TSP

We represent each cut as follows:

- It has a set $\mathcal{F} = \{S_1, \dots, S_j\}$ of vertex sets $S_i \subset V$.
- It has a right-hand side value μ.
- The represented inequality is $\sum_{S_i \in \mathcal{F}} \sum_{e \in \delta(\mathcal{F})} x_e \geq \mu$.

This allows to represent

- subtours: $\mathcal{F} = \{S\}, \mu = 2$,
- combs: $\mathcal{F} = \{H, T_1, \dots, T_k\}, \mu = 3k + 1$,
- even more advanced cuts (Concorde uses this representation).

We know how to add new edges to cuts in this representation!

GENERALIZED DUAL

With cuts (including subtours) represented as set families $\mathcal{F}$ with associated $\mu_{\mathcal{F}}$:

- let $e(\mathcal{F}) = \bigcup_{S \in \mathcal{F}} \delta(S)$ be the edges crossing any set in $\mathcal{F}$,
- let $\chi(e, \mathcal{F})$ be the number of sets in $\mathcal{F}$ crossed by e.

GENERALIZED DUAL

With cuts (including subtours) represented as set families $\mathcal{F}$ with associated $\mu_{\mathcal{F}}$:

- let $e(\mathcal{F}) = \bigcup_{S \in \mathcal{F}} \delta(S)$ be the edges crossing any set in $\mathcal{F}$,
- let $\chi(e, \mathcal{F})$ be the number of sets in $\mathcal{F}$ crossed by e.

The dual becomes:

$$\max 2 \sum_{v \in V} y_v + \sum_{\mathcal{F}} \mu_{\mathcal{F}} z_{\mathcal{F}} - \sum_{e \in E} y_e$$

$$\forall vw \in E : y_v + y_w + \sum_{\mathcal{F}: vw \in e(\mathcal{F})} \chi(e, \mathcal{F}) z_{\mathcal{F}} - y_e \leq c_e$$

$$y_v \text{ free}, y_e, z_{\mathcal{F}} \geq 0$$

GENERALIZED DUAL

With cuts (including subtours) represented as set families $\mathcal{F}$ with associated $\mu_{\mathcal{F}}$:

- let $e(\mathcal{F}) = \bigcup_{S \in \mathcal{F}} \delta(S)$ be the edges crossing any set in $\mathcal{F}$,
- let $\chi(e, \mathcal{F})$ be the number of sets in $\mathcal{F}$ crossed by e.

The dual becomes:

$$\max 2 \sum_{v \in V} y_v + \sum_{\mathcal{F}} \mu_{\mathcal{F}} z_{\mathcal{F}} - \sum_{e \in E} y_e$$

$$\forall vw \in E : y_v + y_w + \sum_{\mathcal{F} : vw \in e(\mathcal{F})} \chi(e, \mathcal{F}) z_{\mathcal{F}} - y_e \leq c_e$$

$$y_v \text{ free}, y_e, z_{\mathcal{F}} \geq 0$$

So we need to find all edges with violated constraints, i.e., cases with

$$\alpha_e = c_e - y_v - y_w + y_e - \sum_{\mathcal{F} : vw \in e(\mathcal{F})} \chi(e, \mathcal{F}) z_{\mathcal{F}} < 0.$$

QUICK REDUCED COST ESTIMATE

So we need to find all edges with violated constraints, i.e., cases with

$$\alpha_e = c_e - y_v - y_w + y_e - \sum_{\mathcal{F}:vw \in e(\mathcal{F})} \chi(e, \mathcal{F}) z_{\mathcal{F}} < 0.$$

Quick Reduced Cost Estimate

So we need to find all edges with violated constraints, i.e., cases with

$$\alpha_e = c_e - y_v - y_w + y_e - \sum_{\mathcal{F}:vw \in e(\mathcal{F})} \chi(e, \mathcal{F}) z_{\mathcal{F}} < 0.$$

If we consider our solution to be a basic solution to the system with the full set of variables reached by primal simplex, all unconsidered edges will have $x_e = 0 < 1 \Rightarrow y_e = 0$ (complementarity).

$$\alpha_e = c_e - y_v - y_w - \sum_{\mathcal{F}:vw \in e(\mathcal{F})} \chi(e, \mathcal{F}) z_{\mathcal{F}} < 0.$$

QUICK REDUCED COST ESTIMATE

So we need to find all edges with violated constraints, i.e., cases with

$$\alpha_e = c_e - y_v - y_w + y_e - \sum_{\mathcal{F}:vw \in e(\mathcal{F})} \chi(e, \mathcal{F}) z_{\mathcal{F}} < 0.$$

If we consider our solution to be a basic solution to the system with the full set of variables reached by primal simplex, all unconsidered edges will have $x_e = 0 < 1 \Rightarrow y_e = 0$ (complementarity).

$$\alpha_e = c_e - y_v - y_w - \sum_{\mathcal{F}:vw \in e(\mathcal{F})} \chi(e, \mathcal{F}) z_{\mathcal{F}} < 0.$$

We have everything we need to compute α_e. But it's too slow because of the sum of cuts.

Quick Reduced Cost Estimate

So we need to find all edges with violated constraints, i.e., cases with

$$\alpha_e = c_e - y_v - y_w + y_e - \sum_{\mathcal{F}: vw \in e(\mathcal{F})} \chi(e, \mathcal{F}) z_{\mathcal{F}} < 0.$$

If we consider our solution to be a basic solution to the system with the full set of variables reached by primal simplex, all unconsidered edges will have $x_e = 0 < 1 \Rightarrow y_e = 0$ (complementarity).

$$\alpha_e = c_e - y_v - y_w - \sum_{\mathcal{F}: vw \in e(\mathcal{F})} \chi(e, \mathcal{F}) z_{\mathcal{F}} < 0.$$

We have everything we need to compute α_e. But it's too slow because of the sum of cuts.
We need a way to (quickly) underestimate $\overline{\alpha}_e \leq \alpha_e$; then check α_e only if $\overline{\alpha}_e < 0$.

Overestimating Dual Cut Sums

Observe that

$$\sum_{\mathcal{F}:vw\in e(\mathcal{F})} \chi(e,\mathcal{F}) z_{\mathcal{F}} = \sum_{\mathcal{F}} \sum_{S\in\mathcal{F}:vw\in\delta(S)} z_{\mathcal{F}}$$

$$= \sum_{\mathcal{F}} \sum_{S\in\mathcal{F}:v\in S} z_{\mathcal{F}} + \sum_{\mathcal{F}} \sum_{S\in\mathcal{F}:w\in S} z_{\mathcal{F}} - 2\sum_{\mathcal{F}} \sum_{S\in\mathcal{F}:\{v,w\}\subseteq S} z_{\mathcal{F}}.$$

$$\leq \sum_{\mathcal{F}} \sum_{S\in\mathcal{F}:v\in S} z_{\mathcal{F}} + \sum_{\mathcal{F}} \sum_{S\in\mathcal{F}:w\in S} z_{\mathcal{F}}.$$

We set

$$\overline{y}_v = y_v + \sum_{\mathcal{F}} \sum_{S\in\mathcal{F}:v\in S} z_{\mathcal{F}} \text{ and } \overline{\alpha}_e = c_e - \overline{y}_v - \overline{y}_w \leq \alpha_e.$$

This check can be done reasonably quickly, apparently even for about 10^6 cities.
Geometry can be used to speed this up even further.

Conclusion

With a clever combination and extensions of LP/IP techniques:

CONCLUSION

With a clever combination and extensions of LP/IP techniques:

- One can tackle very large TSP instances,

Conclusion

With a clever combination and extensions of LP/IP techniques:

- One can tackle very large TSP instances,
- even those for which writing down the full model is impossible,

Conclusion

With a clever combination and extensions of LP/IP techniques:

- One can tackle very large TSP instances,
- even those for which writing down the full model is impossible,
- even those for which the variable set alone is too large.

Conclusion

With a clever combination and extensions of LP/IP techniques:

- One can tackle very large TSP instances,
- even those for which writing down the full model is impossible,
- even those for which the variable set alone is too large.

A stock MIP solver (Gurobi, CPLEX, ...) cannot replicate this easily.

Conclusion

With a clever combination and extensions of LP/IP techniques:

- One can tackle very large TSP instances,
- even those for which writing down the full model is impossible,
- even those for which the variable set alone is too large.

A stock MIP solver (Gurobi, CPLEX, ...) cannot replicate this easily.

- Lazy constraints and user-generated cuts are usually possible (also in a MIP).

Conclusion

With a clever combination and extensions of LP/IP techniques:

- One can tackle very large TSP instances,
- even those for which writing down the full model is impossible,
- even those for which the variable set alone is too large.

A stock MIP solver (Gurobi, CPLEX, ...) cannot replicate this easily.

- Lazy constraints and user-generated cuts are usually possible (also in a MIP).
- Pricing on a pure LP can easily be implemented (just repeatedly solve).

Conclusion

With a clever combination and extensions of LP/IP techniques:

- One can tackle very large TSP instances,
- even those for which writing down the full model is impossible,
- even those for which the variable set alone is too large.

A stock MIP solver (Gurobi, CPLEX, ...) cannot replicate this easily.

- Lazy constraints and user-generated cuts are usually possible (also in a MIP).
- Pricing on a pure LP can easily be implemented (just repeatedly solve).
- Branch, Cut & Price can normally not be implemented on top of their MIP capabilities.

Conclusion

With a clever combination and extensions of LP/IP techniques:

- One can tackle very large TSP instances,
- even those for which writing down the full model is impossible,
- even those for which the variable set alone is too large.

A stock MIP solver (Gurobi, CPLEX, ...) cannot replicate this easily.

- Lazy constraints and user-generated cuts are usually possible (also in a MIP).
- Pricing on a pure LP can easily be implemented (just repeatedly solve).
- Branch, Cut & Price can normally not be implemented on top of their MIP capabilities.
- That would need access to the automatically generated cuts (and may not work with them).

CONCLUSION

With a clever combination and extensions of LP/IP techniques:

- One can tackle very large TSP instances,
- even those for which writing down the full model is impossible,
- even those for which the variable set alone is too large.

A stock MIP solver (Gurobi, CPLEX, ...) cannot replicate this easily.

- Lazy constraints and user-generated cuts are usually possible (also in a MIP).
- Pricing on a pure LP can easily be implemented (just repeatedly solve).
- Branch, Cut & Price can normally not be implemented on top of their MIP capabilities.
- That would need access to the automatically generated cuts (and may not work with them).
- Some toolkits such as SCIP (or a manual Branch & Bound implementation) allow this.

Conclusion

With a clever combination and extensions of LP/IP techniques:

- One can tackle very large TSP instances,
- even those for which writing down the full model is impossible,
- even those for which the variable set alone is too large.

A stock MIP solver (Gurobi, CPLEX, . . .) cannot replicate this easily.

- Lazy constraints and user-generated cuts are usually possible (also in a MIP).
- Pricing on a pure LP can easily be implemented (just repeatedly solve).
- Branch, Cut & Price can normally not be implemented on top of their MIP capabilities.
- That would need access to the automatically generated cuts (and may not work with them).
- Some toolkits such as SCIP (or a manual Branch & Bound implementation) allow this.
- Overall approach in Concorde is much more cut-y and much less branch-y.