Branch-and-cut algorithms for the TSP and the ATSP

Operational Research Complements

Giovanni Righini

References

Based on:

J.E. Mitchell, *Branch-and-cut algorithms for combinatorial optimization problems*, in *Handbook of applied optimization*, P.M. Pardalos and M.G.C. Resende eds., Oxford University Press, 2000.

M. Fischetti, A. Lodi, P. Toth, *Exact methods for the asymmetric traveling salesman problem*, in *The traveling salesman problem and its variations*, G. Gutin and A. Punnen eds., Kluwer, 2002.

Branch-and-cut

Branch-and-cut algorithms arise from the combination of two main ideas:

- implicit enumeration through branch-and-bound;
- iterative improvement of the linear relaxation, through cutting planes.

Branch-and-cut algorithms provide the best known results for a large number of *NP*-hard discrete optimization problems, including the symmetric and the asymmetric traveling salesman problem (TSP and ATSP).

State-of-the-art MILP solvers (e.g. CPLEX, GuRoBi,. . .) are branch-and-cut algorithms.

Designing a branch-and-cut algorithm

Main questions in branch-and-bound algorithms design:

- How to branch?
- How to explore the branch-and-bound tree?
- How to produce primal bounds?
- How to produce dual bounds? By solving the linear relaxation (LP).

Main questions in cutting planes algorithms design:

- Which cutting planes should be used (general or problem-specific)?
- How many cuts should be generated at each round?
- Exact or heuristic separation?

Designing a branch-and-cut algorithm

Additional questions in branch-and-cut algorithms design:

- Branching or cutting?
- Which cuts should be kept in the LP at each node?
- How to make cuts valid for sub-problems different from the one where they have been generated? Lifting!

Implementation

Several techniques can improve the performance of a branch-and-cut algorithm:

- fixing variables by reduced costs
- column generation
- primal bounds by heuristics
- pre-processing
- balancing cutting and branching
- lifting valid inequalities
- interior point algorithms to solve the LP

Variable fixing

Given a relaxed binary variable x_j, assume it is non-basic with reduced cost $\overline{c}_j^* \geq 0$ in the optimal solution of the linear relaxation, x^*, whose minimum value is z^*.

Then any solution with $x_j = 1$ would have a cost not smaller than $z^* + \overline{c}_j^*$.

Hence, if $z^* + \overline{c}_j^* \geq \overline{z}$, where $\overline{z}$ is the best incumbent upper bound, then x_j can be fixed to 0.

In turn, this may trigger further variable fixing.

Column generation

Instead of working with all the variables, one can select some of them (the most promising ones, according to some criterion: the shortest edges in a graph...).

When such a restricted LP problem has been optimized, in order to verify optimality for the original problem it is necessary to check that none of the neglected edges have negative reduced cost.

The reduced cost of all neglected columns (variables) can be computed from the values of the dual variables.

If some column with negative reduced cost is found, it is inserted in the restricted problem and the procedure is iterated.

Primal bounds

Since branch-and-bound explores different regions of the feasible set, and for each sub-problem we obtain an optimal (but fractional) solution of the linear relaxation, all these fractional solutions are a good starting set for a heuristic exploration.

One needs a fast *repair heuristic*, that is a heuristic that takes in input an infeasible (fractional) solution and quickly outputs a “similar” but feasible integer solution.

This heuristic is executed at each node of the branch-and-bound tree (especially coupled with best-first-search) to possibly update the best incumbent primal bound in order to discover the optimal solution early, although it may take an additional lot of time to guarantee its optimality.

Branching vs cutting

As far as a cutting planes algorithm proceeds,

- improvements become smaller and smaller from one iteration to the next one;
- the difficulty of finding further violated inequalities increases (and the computing time, too).

Common practice: search for cuts only at some nodes or at some levels of the tree.

In some implementations, a fixed number of cutting planes iterations is performed at each node, with more iterations at the root node and fewer in the other nodes.

Special case: cut-and-branch. Cutting planes (many of them!) are generated only at the root node.

Local and global cuts

A cut generated at a sub-problem P in the branch-and-bound tree is valid only in P and its descendants: it is a local cut.

A global cut, valid throughout the tree, can be obtained by lifting the variables that have been fixed by branching between the root and P.

The time taken by lifting must be balanced against the advantage of saving memory space:

- local cuts must be stored separately for each sub-problem;
- global cuts need to be stored only once.

Cut pool

Some cuts may become redundant in some sub-problems.

Instead of deleting these cuts, that may become useful again in different branches of the branch-and-bound tree, they are put in a data-structure, called cut pool.

When the current sub-problem is not the successor of the last examined problem, the cut pool is (efficiently) scanned in search for violated cuts before separation algorithms are run.

Interior point methods

For very large problems, even solving the LP relaxation can be a hard task.

Interior point methods may be superior to simplex methods for LPs with thousands of variables.

So, the LP relaxation at the root node of the tree can be solved with an interior point method.

Sometimes IP methods can be used throughout the branch-and-bound tree.

- Pro: IP methods handle degeneracy better than the simplex method.
- Con: Restarting is harder with an IP method.

The TSP

Given a graph $G = (V, E)$ and a cost function $c : E \mapsto \Re$, the Traveling Salesman Problem (TSP) is:

$$\begin{aligned} \text{minimize } z = \sum_{e \in E} c_e x_e & & \\ \text{s.t. } \sum_{e \in \delta(i)} x_e = 2 & \qquad \forall i \in V & \\ S.E.C & & (1) \\ x_e \in \{0, 1\} & \qquad \forall e \in E & \end{aligned}$$

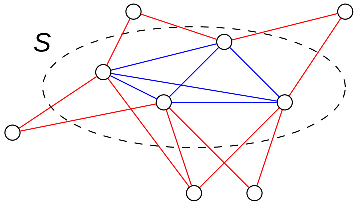
Constraints (1) are called subtour elimination constraints.

Subtour elimination constraints

$$\sum_{e \in \delta(S)} x_e \geq 2 \quad \forall S \subset V : 3 \leq |S| \leq |V|/2 \tag{2}$$

$$\sum_{e \in E(S)} x_e \leq |S| - 1 \quad \forall S \subset V : 3 \leq |S| \leq |V|/2 \tag{3}$$

where $\delta(i)$ is the subset of edges with an endpoint in $i \in V$,
$\delta(S)$ is the subset of edges with an endpoint in $S \subseteq V$,
$E(S)$ is the edge set of the subgraph induced by $S \subseteq V$.



Linear relaxation

The linear relaxation of the TSP is

$$
\begin{aligned}
\text{minimize } z = \sum_{e \in E} c_e x_e & & \\
\text{s.t. } \sum_{e \in \delta(i)} x_e = 2 & \quad \forall i \in V & (4)\\
S.E.C. & \quad \forall S \subset V : 2 \leq |S| \leq |V|/2 & (5)\\
x_e \geq 0 & \quad \forall e \in E & (6)
\end{aligned}
$$

Constraints $x_e \leq 1 \; \forall e \in E$ are equivalent to S.E.C. for $|S| = 2$.

The linear relaxation still contains an exponential number of constraints.

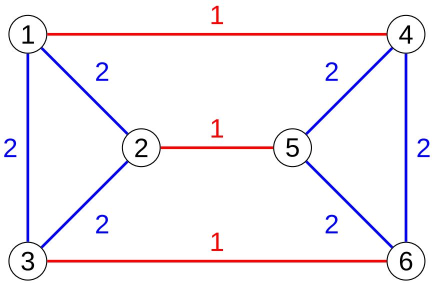
A further relaxation

We also relax the S.E.C.. The relaxation is:

$$\begin{aligned} \text{minimize } z &= \sum_{e \in E} c_e x_e \\ \text{s.t. } \sum_{e \in \delta(i)} x_e &= 2 && \forall i \in V \\ 0 \leq x_e &\leq 1 && \forall e \in E \end{aligned}$$

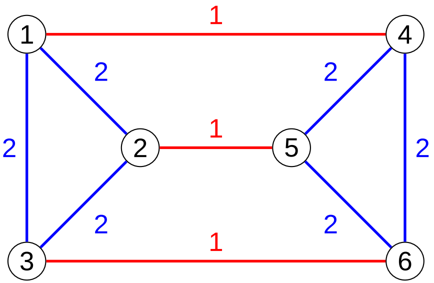
The TSP polyhedron

Unfortunately, the degree constraints (4), the S.E.C. (5) and the bounds (6) are not enough to fully describe the TSP polyhedron.



This fractional solution (blue edges: $x^* = 0.5$, red edges: $x^* = 1$) complies with constraints (4), (5) and (6).

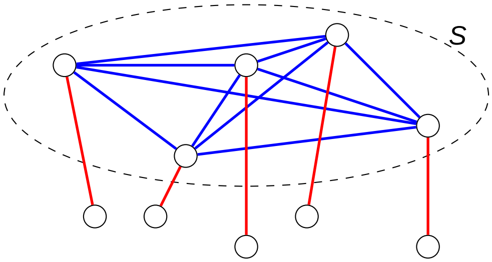
2-matching inequalities



Such an infeasible (fractional) solution is cut off by a 2-matching inequality.

$$x_{12} + x_{13} + x_{23} + x_{14} + x_{25} + x_{36} \leq 4.$$

2-matching inequalities



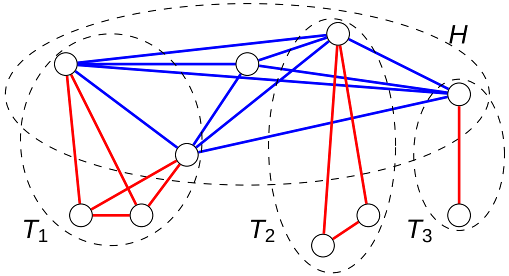
In general 2-matching inequalities (Edmonds, 1965) are facet-defining and have the following form:

$$\sum_{e \in E(S)} x_e + \sum_{e \in T} x_e \leq |S| + \left\lfloor \frac{|T|}{2} \right\rfloor$$

where $S \subset V : |S| \geq 3$ and T is an *odd* subset of *disjoint* edges with one endpoint in S.

Comb inequalities

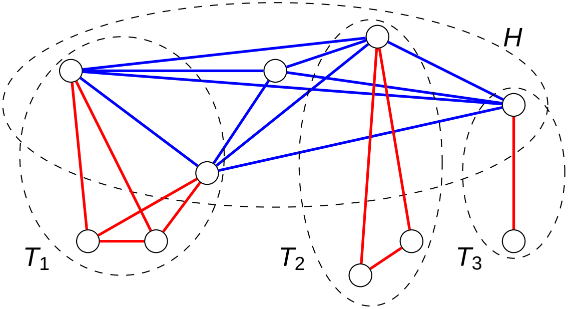
They have been generalized to comb inequalities (Groetschel and Padberg, 1979), which are also facet-defining.



A comb is made by a handle $H \subset V$ and an odd number p of teeth $T_1, \ldots, T_p \subset V$ such that:

- $T_i \cap T_j = \emptyset \ \forall i, j = 1, \ldots, p \ \ i \neq j$;
- $H \cap T_j \neq \emptyset \ \forall j = 1, \ldots, p$;
- $T_j \setminus H \neq \emptyset \ \forall j = 1, \ldots, p$.

Comb inequalities



A comb inequality is:

$$\sum_{e\in E(H)} x_e + \sum_{j=1}^{p} \sum_{e\in E(T_j)} x_e \leq |H| + \sum_{j=1}^{p} |T_j| - \left\lceil \frac{3p}{2} \right\rceil$$

Other inequalities

Other comb inequalities:

- Chvátal comb inequalities are such that $|H \cap T_j| = 1 \; \forall j = 1, \dots, p$ (Chvátal, 1973).
- Simple comb inequalities are such that $(|H \cap T_j| = 1) \vee (|T_j \backslash H| = 1) \; \forall j = 1, \dots, p$ (Letchford and Lodi, 2002).

Other inequalities have a similar structure (handle and teeth):

- Clique-tree inequalities (Groetschel and Pulleyblank, 1986)
- Path inequalities (Cornuéjols, Fonlupt and Naddef, 1985)
- Star inequalities (Fleischmann, 1988)
- Hyperstar inequalities (Fleischmann, unpublished)
- Bipartition inequalities (Boyd and Cunningham, 1991)
- Bi-nested inequalities (Naddef, 1992)

More valid inequalities

Many other families of valid inequalities have been discovered:

- Hypohamiltonian (Groetschel, 1980),
- Chain (Padberg and Hong, 1980),
- Blossom (Padberg and Rao, 1982),
- 2-handled clique tree (Padberg and Rinaldi, 1991),
- Crown (Naddef and Rinaldi, 1992),
- Ladder (Boyd et al., 1995).

This allows to solve TSP instances with thousand nodes to optimality (Padberg and Rinaldi, 1991; Groetschel and Holland, 1991; Applegate et al. 1994).

The state-of-the-art solver for the TSP is *Concorde*, based on Applegate et al. 1994.
Remark: it requires integer arc costs.

Separation

Polynomial-time algorithms for exact separation are known only for some types of inequalities:

- SECs (Crowder and Padberg, 1980)
- 2-matching inequalities (Padberg and Rao, 1982)
- clique tree inequalities with a fixed number of handles and teeth (Carr, 1997)
- certain inequalities defined by lifting (Carr, 1996, 1997).

Heuristic separation always remains an option: it does not guarantee

- to find the most violated inequality of a certain type;
- to find a violated inequality of a certain type every time one exists.

Separation of S.E.C.

Given an optimal solution x^* of the relaxation, we must check whether any S.E.C. is violated.

- Consider the graph $G = (V, E)$ as a flow network;
- Assign each edge e a capacity x_e^*;
- Replace each edge with a pair of opposite arcs with the same capacity;
- Find the all-pairs min-cut.

A cut of capacity less than 2 corresponds to a violated S.E.C.

The all-pairs min-cut can be found in polynomial time (Gomory-Hu, Padberg-Rinaldi, Hao-Orlin, . . .).

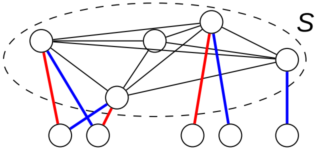
The algorithm by Nagamochi, Ono and Ibaraki (1994) does it in $O(nm + n^2 \log n)$.

Separation of 2-matching inequalities

$$\sum_{e \in E(S)} x_e + \sum_{e \in T} x_e \leq |S| + \left\lfloor \frac{|T|}{2} \right\rfloor$$

can be rewritten (Padberg and Rao, 1982) as

$$\sum_{e \in \delta(S) \setminus T} x_e + \sum_{e \in T} (1 - x_e) \geq 1.$$



Letchford, Reinelt and Theis (2003) showed how to separate these cuts with $n-1$ max-flows computations, leading to an exact separation algorithm with $O(n^2 m \log(n^2/m))$ time complexity.

$(0, \frac{1}{2})$-cuts

Given an integer program with constraints set $Ax \leq b$, where $A \in \Re^{m \times n}$ and $b \in \Re^{m}$, *mod − k* cuts are valid inequalities with the form

$$\lambda^T Ax \leq \lfloor \lambda^T b \rfloor$$

with $\lambda \in \left\{0, \frac{1}{k}, \ldots, \frac{k-1}{k}\right\}^m$ such that $\lambda^T A$ is integer.

In particular, for $k = 2$, Caprara, Fischetti, Letchford and Lodi found polynomial-time algorithms for exact separation of violated $(0, \frac{1}{2})$-cuts.

The ATSP

Given a digraph $D = (N, A)$ and a cost function $c : A \mapsto \Re$, the Asymmetric Traveling Salesman Problem is:

$$
\begin{aligned}
\text{minimize } z = \sum_{(i,j)\in A} c_{ij}x_{ij} & & & \\
\text{s.t. } \sum_{j\in N} x_{ij} = 1 & & \forall i \in N & \\
\sum_{i\in N} x_{ij} = 1 & & \forall j \in N & \\
\sum_{i\in S}\sum_{j\in \overline{S}} x_{ij} \geq 1 & & \forall S \subset N : S \neq \emptyset & \qquad (7)\\
x_{ij} \in \{0,1\} & & \forall (i,j) \in A &
\end{aligned}
$$

Constraints (7) are called subtour elimination constraints.

Subtour elimination constraints

Alternative subtour elimination constraints:

$$\sum_{i \in S} \sum_{j \in S} x_{ij} \leq |S| - 1 \;\; \forall S \subset N : S \neq \emptyset.$$

They remain exponentially many, although their number can be halved in the form

$$\sum_{i \in S} \sum_{j \in \overline{S}} x_{ij} \geq 1 \;\; \forall S \subset N : r \in S$$

$$\sum_{i \in S} \sum_{j \in S} x_{ij} \leq |S| - 1 \;\; \forall S \subset N : S \neq \emptyset, r \notin S$$

for an arbitrary chosen node $r \in N$.

Transforming an ATSP into a TSP instance

3**-node transformation** (Karp, 1972).

A complete undirected graph with $3n$ vertices is obtained from the original complete digraph with n nodes:

- add two copies, $n+i$ and $2n+i$, of each vertex $i \in N$;
- set to 0 the cost of the edges $[i, n+i]$ and $[n+i, 2n+i]$ for each $i \in N$;
- set to c_{ij} the cost of edge $[2n+i, j]$ forall $i, j \in N$;
- set to ∞ the cost of all the remaining edges, including $[i, 2n+i]$ forall $i \in N$.

Transforming an ATSP into a TSP instance

2**-node transformation** (Jonker and Volgenant, 1983).

A complete undirected graph with $2n$ vertices is obtained from the original complete directed one:

- add a copy $n + i$ of each vertex $i \in N$;
- set to 0 the cost of edge $[i, n + i]$ for each $i \in N$;
- set to $c_{ij} + M$ the cost of edge $[n + i, j]$ for each $i, j \in N$, where $M > 0$ is a “large enough” constant;
- set to ∞ the costs of all the remaining edges;
- an amount nM must be subtracted from the TSP cost to obtain the ATSP cost.

Symmetric inequalities

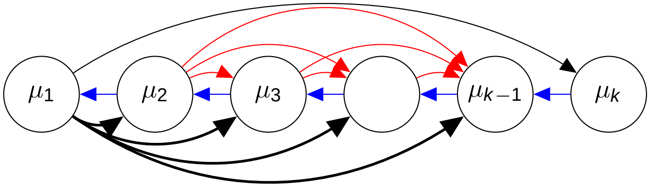
Symmetric inequality: $\alpha x \leq \alpha_0$ with $\alpha_{ij} = \alpha_{ji}\ \forall (i,j) \in A$.

If y_e indicates the binary variable of edge $e = [i,j]$ in the TSP graph, the variable substitution $y_e = x_{ij} + x_{ji}$ transforms any valid inequality for the TSP into a symmetric valid inequality for the ATSP.

D_k^+ inequalities (Groetschel and Padberg, 1985)

Consider any sequence $\mu_1, \dots, \mu_k$ of $3 \leq k \leq n-1$ nodes.

$$x_{\mu_1 \mu_k} + \sum_{h=2}^{k} x_{\mu_h \mu_{h-1}} + \mathbf{2} \sum_{\mathbf{h=2}}^{\mathbf{k-1}} \mathbf{x_{\mu_1 \mu_h}} + \sum_{h=3}^{k-1} \sum_{p=2}^{h-1} x_{\mu_p \mu_h} \leq k-1.$$



For $k = 2$ they degenerate into $x_{ij} + x_{ji} \leq 1$.
For $k = 1$ they degenerate into $x_{ii} \leq 0$.

D_k^+ inequalities

D_k^+ inequalities are facet-inducing; they can be obtained by suitably lifting the S.E.C. $\sum_{i \in S} \sum_{j \in S} x_{ij} \leq |S| - 1 \ \ \forall S \subset N : S \neq \emptyset$.

The separation problem requires to find the most violated D_k^+ inequality and it can be solved by implicit enumeration, owing to an effective and easy-to-compute bounding rule, when all SEC are satisfied.

The separation algorithm devised by Fischetti and Toth (1997) was later proven to have polynomial-time complexity.

D_k^- inequalities

D_k^- inequalities are a symmetric counterpart of D_k^+ inequalities:

$$x_{\mu_k \mu_1} + \sum_{h=2}^{k} x_{\mu_{h-1}\mu_h} + \mathbf{2}\sum_{\mathbf{h=2}}^{\mathbf{k-1}} \mathbf{x}_{\mu_\mathbf{h} \mu_\mathbf{1}} + \sum_{h=3}^{k-1}\sum_{p=2}^{h-1} x_{\mu_h \mu_p} \leq k-1.$$

D_k^- inequalities are facet-inducing; they can be obtained by suitably lifting the S.E.C..

D_k^- inequalities are obtained from D_k^+ inequalities by transposition, which is valid in general.

Defining α^T so that $\alpha_{ij}^T = \alpha_{ji}$, every inequality $\alpha^T x \leq \alpha_0$ is valid for the ATSP polyhedron if and only if $\alpha x \leq \alpha_0$ is.

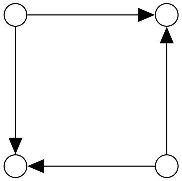
Odd Closed Alternating Trails (CAT) cuts

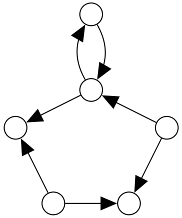
Two distinct arcs (i,j) and (u,v) are incompatible if

$$(i = u) \vee (j = v) \vee ((j = u) \wedge (i = v))$$

and compatible otherwise.

A Closed Alternating Trail (Balas, 1989) is a cyclic sequence of t distinct arcs $T = (a_1, a_2, \ldots, a_t)$ such that each arc in T is incompatible with the two adjacent arcs and compatible with all the others.





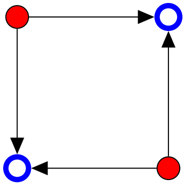
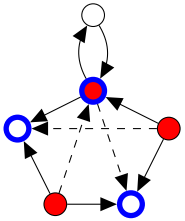
Odd Closed Alternating Trails (CAT) cuts

Indicating with $\delta^-(i)$ the in-star and with $\delta^+(i)$ the out-star of each node $i \in N$,

- a node $i \in N$ is a source ($i \in T^+$), if $|\delta^+(i) \cap T| = 2$;
- a node $i \in N$ is a sink ($i \in T^-$), if $|\delta^-(i) \cap T| = 2$.

Remark. A node can be both a source and a sink.

Let $Q = \{(i,j) \in A \backslash T : i \in T^+, j \in T^-\}$. [dashed arcs]

Odd Closed Alternating Trails (CAT) cuts

For any CAT of odd length t, the odd CAT inequality

$$\sum_{(i,j) \in T \cup Q} x_{ij} \leq \frac{|T|-1}{2}$$

is facet-defining for the ATSP polytope.

Separation: search for odd simple cycles in a suitable incompatibility graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$:

- $\mathcal{G}$ is undirected and weighted;
- $\mathcal{V}$ has a vertex for each arc (i,j) such that $x^*_{ij} > 0$;
- $\mathcal{E}$ has an edge $e = [a,b]$ linking each pair of incompatible arcs $a \in A$ and $b \in A$;
- the weight of each edge is $w_e = 1 - x^*_a - x^*_b$. If x^* satisfies all degree constraints and the trivial S.E.C. constraints $x_{ij} + x_{ji} \leq 1$, then $w_e \geq 0 \; \forall e \in \mathcal{E}$.

Odd Closed Alternating Trails (CAT) cuts

Let $\delta(v)$ be the edge set incident to $v \in \mathcal{V}$.

A cycle in $\mathcal{G}$ is subset $\mathcal{C} \subseteq \mathcal{E}$ such that $|\mathcal{C} \cap \delta(v)|$ is even for all $v \in \mathcal{V}$.

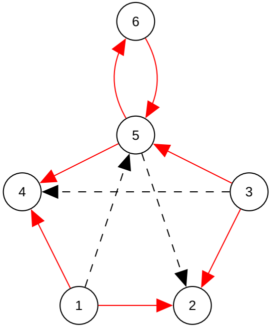
A cycle $\mathcal{C}$ is odd if $|\mathcal{C}|$ is odd.

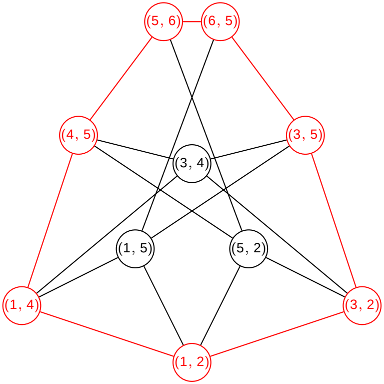
A cycle $\mathcal{C}$ is simple if $|\mathcal{C} \cap \delta(v)| \in \{0, 2\}$ $\forall v \in \mathcal{V}$.

A cycle $\mathcal{C}$ is chordless if there are no edges linking its vertices besides those in $\mathcal{C}$.

Odd Closed Alternating Trails (CAT) cuts

By construction an odd, simple and chordless cycle $\mathcal{C}$ in $\mathcal{G}$ corresponds to an odd CAT in the ATSP digraph.





Odd Closed Alternating Trails (CAT) cuts

The weight of $\mathcal{C}$ is

$$w(\mathcal{C}) = \sum_{e \in \mathcal{C}} w_e = \sum_{[a,b] \in \mathcal{C}} (1 - x_a^* - x_b^*) = |T| - 2 \sum_{a \in T} x_a^*$$

The violation of the odd CAT inequality is

$$\Phi(T) = \sum_{(i,j) \in T \cup Q} x_{ij}^* - \frac{|T| - 1}{2}.$$

Then, the following relations hold:

$$\begin{aligned} \Phi(T) &= \frac{2 \sum_{(i,j) \in T \cup Q} x_{ij}^* - (|T| - 1)}{2} \geq \\ &\geq \frac{2 \sum_{(i,j) \in T} x_{ij}^* - |T| + 1}{2} = \frac{1 - w(\mathcal{C})}{2}. \end{aligned}$$

Fischetti and Toth (1997) developed a heuristic separation algorithm that finds a minimum weight odd cycle through each edge $e \in \mathcal{E}$.

Clique lifting

Given any valid inequality $\beta y \leq \beta_0$ for the ATSP digraph $D' = (N', A')$, let define for each node $k \in N'$

$$\beta_{kk} = \max\{\beta_{ik} + \beta_{kj} - \beta_{ij} : i, j \in N' \setminus \{k\}, i \neq j\}.$$

Define a digraph $D = (N, A)$ in which each node $k \in N'$ is replaced by a clique S_k containing one or more nodes in N; assign weight β_{kk} to all arcs inside the clique S_k corresponding to node $k \in N'$; assign weight β_{ij} to all arcs of A from S_i to S_j.

Then the clique lifted inequality $\alpha y \leq \alpha_0$ with

$$\alpha_0 = \beta_0 + \sum_{k \in N'} \beta_{kk}(|S_k| - 1)$$

and $\alpha_{uv} = \beta_{ij}\ \forall u \in S_i, v \in S_j$ is valid for the ATSP defined on digraph D (Balas and Fischetti, 1993).

Furthermore, if $\beta y \leq \beta_0$ is facet-defining for D', then $\alpha x \leq \alpha_0$ is facet-defining for D.

Clique lifting and shrinking

Clique lifting is a powerful tool for extending known inequalities.

It may also simplify separation, owing to the shrinking operation, that is the opposite of lifting.

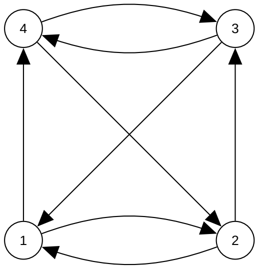
Let $S \subset N$, with $2 \leq |S| \leq |N| - 2$ and such that $\sum_{a \in A(S)} x_a = |S| - 1$, where $A(S)$ is the arc set of the subdigraph induced by S.

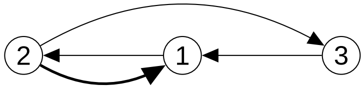
Replace S with a single node σ, updating x^* into y^* accordingly.

Every valid inequality $\beta y \leq \beta_0$ violated by y^* in the shrunken digraph corresponds to a valid inequality $\alpha x \leq \alpha_0$ violated by x^* in the original digraph and $\alpha x \leq \alpha_0$ can be obtained by clique lifting from $\beta y \leq \beta_0$ by replacing back σ with S.

Shrinking

It is not guaranteed that a violated inequality is detected if one exists: it is possible that shrinking produce a solution y^* that does not violate any inequality even if x^* does.



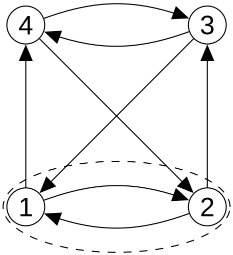


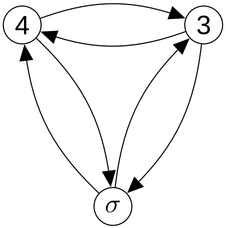
Assuming $x^* = 0.5$ for all arcs indicated on the left, x^* violates the D_3^+ inequality shown on the right:

$$x_{12} + x_{23} + x_{31} + 2x_{21} \leq 2.$$

Shrinking

Shrinking the saturated subset $\{1, 2\}$ the digraph on the right is obtained. For all arcs *a* on the right, $y^*_a = 0.5$.





The fractional solution on the right is a convex combination of two integer solutions; therefore it cannot be cut-off by any valid inequality.

Shrinking

In some cases it is guaranteed that a violated inequality $\beta y \leq \beta_0$ exists in the shrunken digraph every time a violated inequality $\alpha x \leq \alpha_0$ exists in the original digraph.

The simplest of these cases is when an arc (i, j) with $x^*_{ij} = 1$ is shrunken.

Pricing

The LP relaxation to be iteratively solved is

$$z = \min\{cx : Dx = 1, Fx \leq g, x \in \Re_+^{|A|}\},$$

where constraints $Dx = 1$ are $2n$ degree constraints and $Fx \leq g$ are possible additional valid inequalities.

The number of non-negative variables x is the number of arcs in the digraph, which grows quadratically with the number of nodes.

In order to keep the size of the LP problem limited, a restricted LP is considered, by selecting a suitable subset $\overline{A} \subseteq A$ of the variables.

All variables in $A \backslash \overline{A}$ are fixed to 0, which means that the corresponding are disregarded.

Pricing

Assuming the restricted problem

$$z_R = \min\{cx : Dx = 1, Fx \leq g, x \in \Re_+^{|\overline{A}|}\},$$

is feasible, let x^* and u^* the primal and the dual optimal solutions, respectively.

Then the reduced cost of each arc $(i,j) \in A$ is

$$\overline{c}_{ij} = c_{ij} - \sum_{r \in R_D} D_{r,(i,j)} u_r^* - \sum_{r \in R_F} F_{r,(i,j)} u_r^*,$$

where R_D and R_F are the row sets of D and F.

If $\overline{c}_{ij} \geq 0 \ \forall (i,j) \in A$, then x^* is optimal also for the unrestricted problem.
Otherwise, variables with negative reduced cost are inserted in $\overline{A}$ and the procedure is iterated.

Pricing: finding columns with negative reduced cost.

Pricing

A sensible initialization for $\overline{A}$ is to select k arcs with minimum cost incident to each node (e.g. $k = 15$).

At each pricing iteration of arbitrarily selecting arcs with negative reduced cost, one can exploit the degree constraints, solving a linear assignment problem on the ATSP digraph weighted with the reduced costs.

If the optimal assignment has zero cost, then the algorithm stops and x^* is optimal.
Otherwise the arcs in the optimal assignment are inserted in $\overline{A}$ and the procedure is iterated.