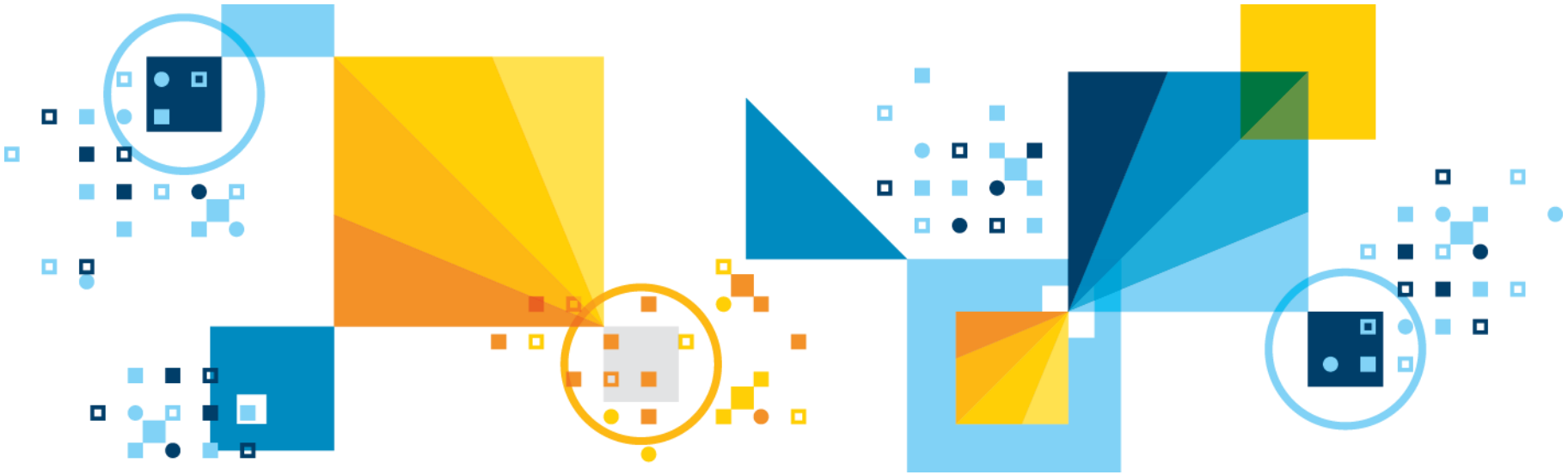
The Asymmetric Traveling Salesman Problem with Time Windows

@CPLEX school 2017, Montreal



Outline

- Introduction
  - Problem definition
  - A compact MILP formulation: the big-M formulation
  - Exercise and prerequisites for the CPLEX school
- Weakness of the big-M formulation
- Instance preprocessing
- Valid inequalities and separation algorithms
- References

The Traveling Salesman Problem with Time Windows (TSPTW)

- An important generalization of the classical TSP, where each city of the underlying graph must be visited within a given time window
- Complexity results
  - TSPTW is NP-hard (generalizes the classical TSP)
  - Finding a feasible TSPTW solution in a complete graph is NP-complete [Savelsbergh, 1985]
- Most common real-world applications: routing, scheduling, manufacturing and delivering problems
- Exact algorithms:
  - Branch-and-cut [Ascheuer et al., 2001]
  - Hybrid approach (Constraint Programming and OR techniques) [Focacci et al., 2002]
  - Extended formulation and branch-and-cut [Dash et al., 2012]
  - Column generation and dynamic programming [Baldacci et al, 2012]
- In the following we will consider the Asymmetric variant of TSPTW (ATSPTW), with asymmetric traveling costs and traveling times.

ATSPTW: problem definition

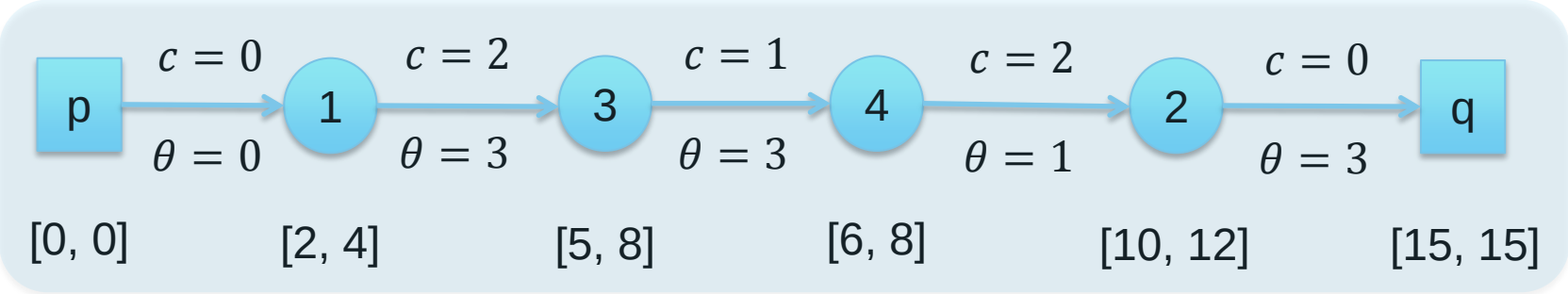
- Given a directed graph G = (V, A), with
  - A traveling cost $c_{ij} \geq 0$ associated with each arc $(i,j) \in A$
  - A traveling time $\theta_{ij} > 0$ associated with each arc $(i,j) \in A$
  - A time window $W_i = [R_i, D_i]$ associated with each node $i \in V$, where
    - R_i is the release time of node i (i.e., the earliest possible start time for node i)
    - D_i is the deadline of node i (i.e., the latest possible start time for node i)
  - Two special nodes $p, q \in V$

- The problem asks for finding a minimum cost Hamiltonian path from p to q in G that satisfies all time windows restrictions.

- Time windows restrictions:
  - Waiting times are allowed:
    - The salesman can arrive at node i at any time $t \leq D_i$ (even at time $t < R_i$)
    - But he can leave from node i only at time $t \geq R_i$

ATSPTW: example

- V = {0,1, 2, 3, 4, 5}, p = 0, q = 5
- Traveling costs, traveling times and time windows as in the picture



- Feasible solution: path (p,1, 3, 4, 2, q)
  - Waiting time at node 1
    - The salesman arrives at t = 0 but needs to wait until t = 2 before leaving the node
  - Waiting time at node 2
    - The salesman arrives at t = 9 but needs to wait until t = 10 before leaving the node

ATSPTW: A side note on the problem definition

- In some variants of the problem, processing time $\alpha_i \geq 0$ associated with the nodes are also given.
- In such a case, an equivalent ATSPTW with $\alpha_i = 0 \; \forall \; i \in V$ can be obtained by incorporating node processing times into arc travel times.

- A relevant example
  - In the instances proposed by [Ascheuer et al., 2001]
    - Travel times θ_{ij} are the same as travel costs c_{ij}
    - But processing time $\alpha_i \geq 0$ for nodes $i \in V$ are also given
  - An equivalent ATSPTW with $\alpha_i = 0 \; \forall \; i \in V$ can be obtained by setting
    - $\theta_{ij} \; := \; c_{ij} + \; \alpha_i \; \forall \; (i,j) \in A$
    - $\alpha_i \; := 0 \; \forall \; i \in V$
  - **Remark:** those are the instances we will use during the lab sessions

ATSPTW: Some more notation

- For every two sets of nodes $S, T \subseteq V$, the set of arcs from S to T is

$$\delta(S,T) \coloneqq \{(i,j) \in A : i \in S, j \in T\}$$

- For every set of nodes $S \subseteq V$, define $\overline{S} \coloneqq V \setminus S$ the complement of S. By abusing notation, we use:
  - $\delta^+(S)$ instead of $\delta(S, \overline{S})$ to denote the arcs from S to $\overline{S}$
  - $\delta^-(S)$ instead of $\delta(S, \overline{S})$ to denote the arcs from $\overline{S}$ to S

- Finally, when $S = \{i\}$ is a singleton, we use
  - $\delta^+(i)$ in place of $\delta^+(\{i\})$ to denote the arcs exiting from node i
  - $\delta^-(i)$ in place of $\delta^-(\{i\})$ to denote the arcs entering in node i

ATSPTW: The big-M MILP formulation

- Binary decision variables x_{ij} $((i,j) \in A)$ and continuous decision variables s_i $(i \in V)$

$$x_{ij} = \begin{cases} 1, \text{ if arc } (i,j) \in A \text{ is selected;} \\ 0, \text{ otherwise.} \end{cases}$$

$$s_i = \text{ start time at node } i \in V$$

- The model then reads:

$$min \sum_{(i,j) \in A} c_{ij} x_{ij} \qquad (1)$$

$$\sum_{(i,j) \in \delta^+(i)} x_{ij} = 1, \qquad \forall\, i \in V \setminus \{q\} \qquad (2)$$

$$\sum_{(j,i) \in \delta^-(i)} x_{ji} = 1, \qquad \forall\, i \in V \setminus \{p\} \qquad (3)$$

$$R_i \leq s_i \leq D_i, \qquad \forall\, i \in V \qquad (4)$$

$$x_{ij} \in \{0,1\}, \qquad \forall\, (i,j) \in A \qquad (5)$$

$$s_i + \theta_{ij} - (1 - x_{ij}) M_{ij} \leq s_j, \qquad \forall\, (i,j) \in A \qquad (6)$$

ATSPTW: The big-M constraints [Miller, Tucker and Zemlin, 1960]

- The big-M constraints (also known as MTZ-inequalities)

$$s_i + \theta_{ij} - (1 - x_{ij})M_{ij} \leq s_j, \qquad \forall (i,j) \in A \tag{6}$$

  - Are used to model the following logical implications:

$$x_{ij} = 1 \quad \Rightarrow \quad s_i + \theta_{ij} \leq s_j, \qquad \forall (i,j) \in A \tag{7}$$

  - Big-M constants M_{ij} must be big enough, in order to yield a correct formulation where all feasible solutions are preserved
  - Big-M constants M_{ij} must be as small as possible, to yield a tighter formulation
  - As shown by [Miller et al., 1960] , the best possible choice is

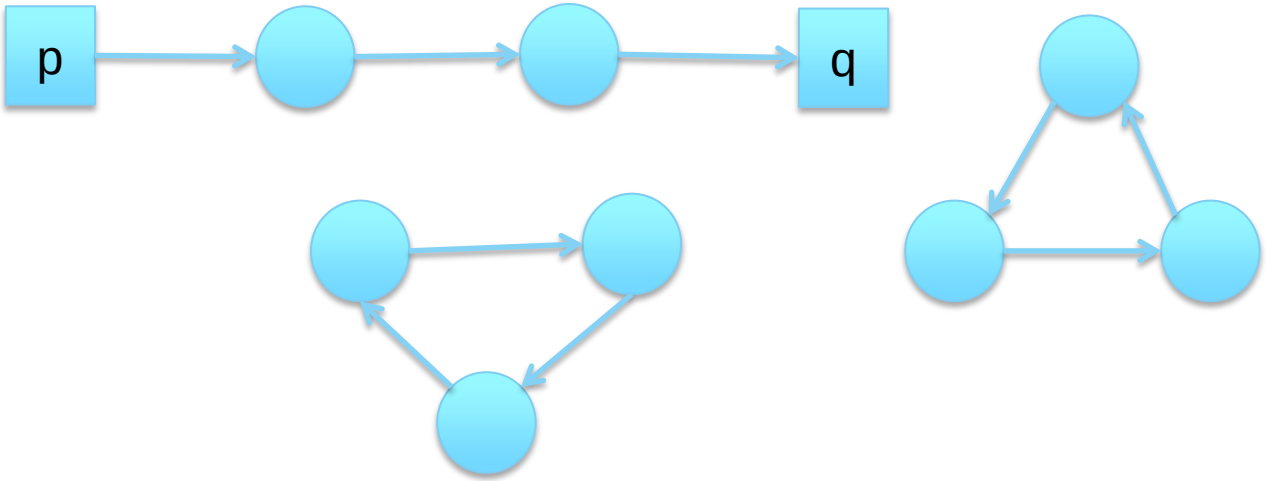
$$M_{ij} = D_i - R_j + \theta_{ij}, \qquad \forall (i,j) \in A \tag{8}$$

ATSPTW: The big-M constraints [Miller, Tucker and Zemlin, 1960]

- The big-M constraints (also known as MTZ-inequalities)

$$s_i + \theta_{ij} - (1 - x_{ij})M_{ij} \leq s_j, \qquad \forall (i,j) \in A \tag{6}$$

  - Together with the other constraints (1)-(5), enforce both time windows constraints and connectivity (i.e., they remove subcycles)
  - If removed, feasible solutions of the resulting relaxation (1)-(5) correspond to
    - One simple path from p to q,
    - Plus one or more subcycles,
    - With no time window restriction



Prerequisites for the CPLEX school

- All attendees should arrive with a laptop with
  - CPLEX 12.7.1 installed
  - Python installed
  - DOcplex installed
  - The C code provided with these slides compiling and running
  - The ipython notebook code provided with these slides properly loading

- Attendees should review the provided C and python code and try to do the exercise proposed in the next slides

Suggested reading:

S. Dash, O. Günlük, A. Lodi, A. Tramontani. A time bucket formulation for the traveling salesman problem with time windows. INFORMS J. Comput. 24, 132 – 147, 2012.

Exercise (preparation for CPLEX school)

- DOcplex and Python, complete the provided notebook:
  - Make sure that you can load everything.
  - Complete the big-M formulation.
  - Make procedures to check that a solution is indeed a tour.
  - Online DOcplex documentation available at

    http://ibmdecisionoptimization.github.io/docplex-doc/#mathematical-programming-modeling-for-python-using-docplex-mp-docplex-mp

Exercise (preparation for CPLEX school)

- CPLEX C APIs
  - Develop an application in C that
    1. Reads an ATSPTW instance from text file
    2. Builds the big-M model
    3. Solves it
    4. Checks the solution obtained
  - A source code that already implements Steps 1 and 3 is provided. Please refer to the README file given together with the source code for more details
  - Few benchmarks instances are also given together with the code
  - Other benchmark instances can be found at http://ftp.zib.de/pub/mp-testdata/tsp/atsptw/
  - Online CPLEX documentation:
    - CPLEX home page: https://www.ibm.com/support/knowledgecenter/SSSA5P_12.7.1/ilog.odms.cplex.help/CPLEX/homepages/CPLEX.html
    - CPLEX Callable Library (C API) Reference Manual: https://www.ibm.com/support/knowledgecenter/SSSA5P_12.7.1/ilog.odms.cplex.help/refcallablelibrary/homepageCrefman.html

Weakness of the big-M formulation

Weakness of the big-M formulation

- The big-M formulation (1)-(6) is compact (polynomial number of variables and constraints) and can be solved as a black box
- However, it has two main drawbacks
  - The LP relaxation is generally very weak
  - The model can be numerically unstable and can lead to incorrect results
- This is especially true if (some of) the time windows are too wide and lead to big M_{ij} values

Weakness of the big-M formulation

$$s_i + \theta_{ij} - (1 - x_{ij})M_{ij} \leq s_j, \qquad (6)$$

$$\text{with } M_{ij} = D_i - R_j + \theta_{ij},$$

$$x_{ij} = 1 \;\Rightarrow\; s_i + \theta_{ij} \leq s_j \qquad (7)$$

- Integer solutions can violate logical implications (7) modeled by the big-M constraints (6)
  - With $\boldsymbol{x_{ij} = 1 - \varepsilon}$ the constraint (6) for arc (i, j) reads

$$s_i + \theta_{ij} - \varepsilon M_{ij} \leq s_j$$

  - If $\boldsymbol{M_{ij}}$ is too big, even small values of $\boldsymbol{\varepsilon}$ that are below the integrality tolerance can lead to integer solutions that
    - satisfy constraints (6), but
    - violate the implications (7)
  - **Possible fixes**
    - set MIP integrality tolerance to 0
    - use indicator constraints to model implications (7)

Weakness of the big-M formulation

- Constraints with very large dynamism can generate numerical troubles:
  - Internally, CPLEX scales columns and rows of the constraint matrix to avoid simplex numerical issues and speed-up the computation.

$$s_i + \theta_{ij} - (1 - x_{ij})M_{ij} \leq s_j, \qquad (6)$$

$$\text{with } M_{ij} = D_i - R_j + \theta_{ij},$$

$$x_{ij} = 1 \;\Rightarrow\; s_i + \theta_{ij} \leq s_j \qquad (7)$$

  - If $\boldsymbol{M_{ij}}$ are too big, small infeasibilities in the internal scaled representation of the problem can result in large infeasibilities in the unscaled model that CPLEX might fail to recover
  - **Workarounds**
    - Adjust feasibility and optimality tolerance
    - Adjust coefficient scaling
    - Turn on numerical emphasis
  - **Fix:** use indicator constraints to model implications (7)

Weakness of the big-M formulation

- The model can be numerically unstable and can lead to incorrect results
  - Workarounds: Tuning of tolerances, numerical emphasis
  - Fix: use indicator constraints

- The LP relaxation is generally very weak
  - Exploit the problem structure to
    - Apply some ad-hoc data preprocessing (also to cure the numerical instability)
    - Derive valid inequalities to tighten the LP relaxation

Instance preprocessing

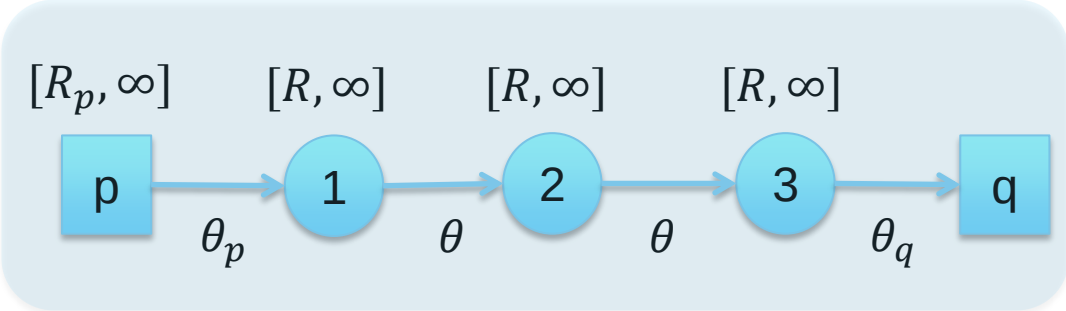
Instance preprocessing

- **Goals**
  - Cure the numerical instability of the big-M formulation
  - Tighten the LP relaxation of the big-M formulation
  - Derive precedence relationships among nodes for cutting plane separation
- Precedence notation
  - For any pair of nodes $i, j \in V$, we say that $\boldsymbol{i \prec j}$ **(i.e., $\boldsymbol{i}$ precedes $\boldsymbol{j}$)** if node i must be visited before node j in any feasible solution
  - Precedences among nodes can be inferred by exploiting the time windows
- Assumption:
  - Traveling times satisfy the triangle inequality:

$$\theta_{ik} \leq \theta_{ij} + \theta_{jk} \quad \forall\, i, j, k \in V$$

Preprocessing: Basic tightening of node deadlines

- Given any valid upper bound T on the arrival time at the last node q, the node deadlines can be easily tightened as follows
  - $D_q = \min\{D_q, T\}$, and $D_i = \min\{D_i, D_q - \theta_{iq}\}\ \forall\ i \in V \setminus \{q\}$
- A trivial upper bound T can be computed as follows
  - $V_0 \coloneqq V \setminus \{p, q\}$, with $n = |V_0|$
  - $\theta_p = max_{\{(p,j) \in A : j \in V_0\}}\ \theta_{pj}$
  - $\theta_q = max_{\{(i,q) \in A : i \in V_0\}}\ \theta_{iq}$
  - $\theta = max_{\{(i,j) \in A : i \in V_0, j\ \epsilon\ V_0\}}\ \theta_{ij}$
  - $R = max_{\{i \in V_0\}}\ R_i$
  - $\boldsymbol{T = \max\{R_p + \theta_p, R\} + (n-1)\theta + \theta_q}$



- On the instances by [Ascheuer et al., 2001], this is enough to remove insanely large time windows and $\boldsymbol{M_{ij}}$ values

Preprocessing: iterative loop [Ascheuer et al., 2001]

- A more elaborated preprocessing can be obtained by iterating over the following steps:

1. Time windows tightening
   - Example: For every $k \in V \setminus \{p\}$, compute the smallest arrival time at node k as $T_k = min_{\{(i,k) \in A\}} \{R_i + \theta_{ik}\}$. The release date of R_k can be tightened as $R_k = \max\{R_k, T_k\}$.
   - See [Ascheuer et al., 2001] for other similar tightening operations
2. Precedence identification
   - Trivial ones: $p \prec j$ for every $j \in V \setminus \{p\}$ and $i \prec q$ for every $i \in V \setminus \{q\}$
   - From time windows: $R_i + \theta_{ij} > D_j \Rightarrow j \prec i$
   - Transitive closure: $i \prec j \wedge j \prec k \Rightarrow i \prec k$
3. Arc removal
   - $i \prec j \Rightarrow \text{arc}\,(j,i)$ can be removed
   - $i \prec j \wedge j \prec k \Rightarrow \text{arc}\,(i,k)$ can be removed
   - Etc.

Valid inequalities

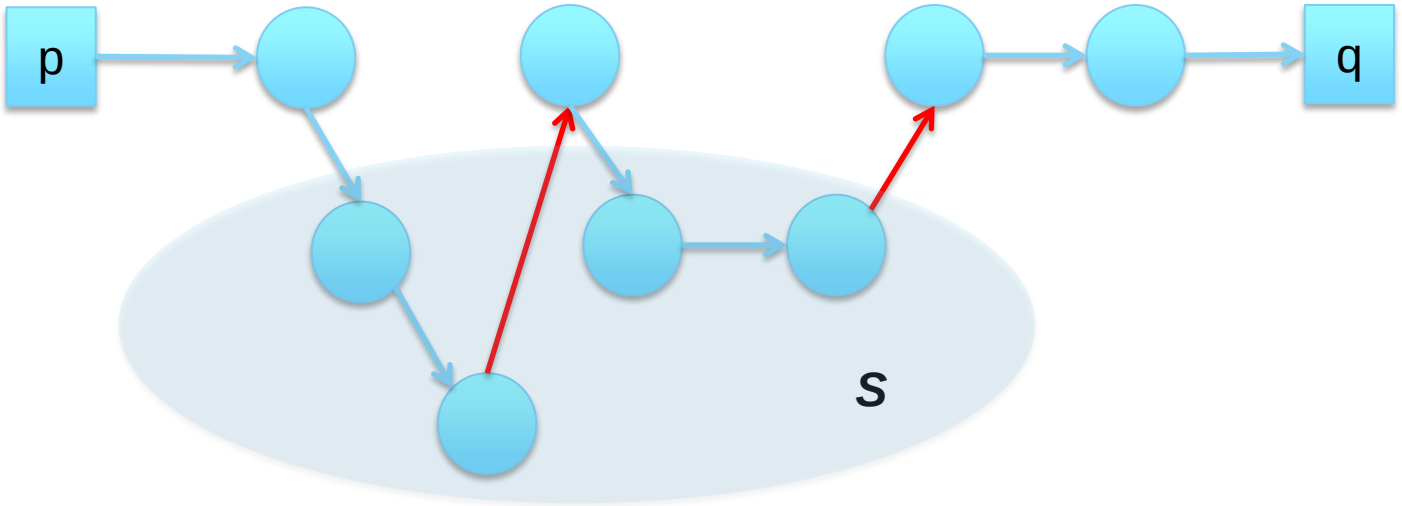
Valid inequalities

- Many valid inequalities have been proposed for TSPTW (see, e.g., [Ascheuer et al., 2000]). In this presentation we will focus on
  - Subtour Elimination Constraints (SECs)
  - Two classes of Sequential Ordering Polytope Inequalities (SOP-ineqs):
    - Predecessor inequalities (π-ineqs)
    - Successor inequalities (σ-ineqs)
- Some more notation:
  - For any subset of arcs $\boldsymbol{Q} \subseteq \boldsymbol{A}$, we define $x(Q) \coloneqq \sum_{(i,j) \in Q} x_{ij}$

Subtour Elimination Constraints (SECs)

- For every subset of nodes $S \subseteq V \setminus \{p, q\}$, the following inequality is valid for (1)-(6):

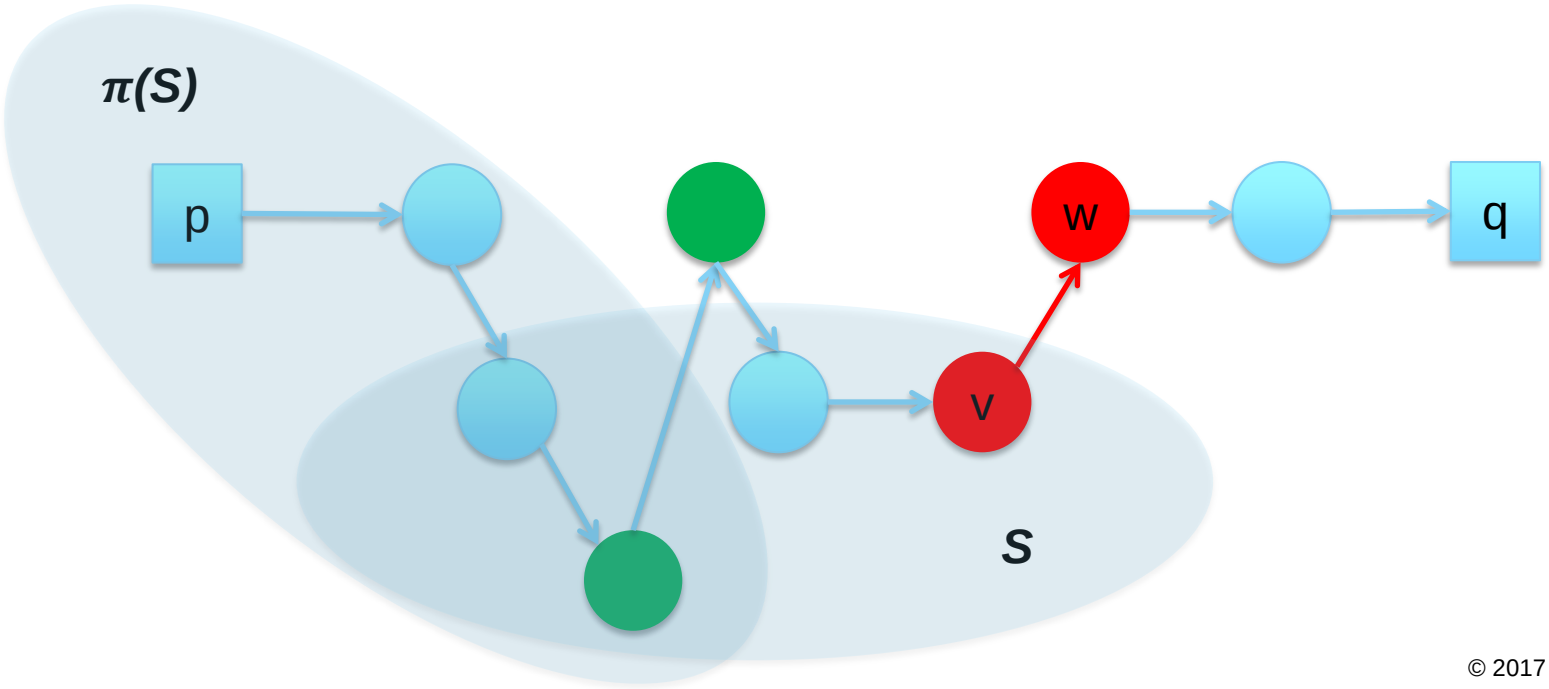
$$x(\delta^+(S)) \geq 1 \qquad (9)$$



Predecessor inequalities (π-ineqs) [Balas et al., 1995]

- For every node $i \in V$, define $\pi(i) := \{j \in V \setminus \{i\} : j \prec i\}$
- For every subset of nodes $S \subseteq V$, define $\pi(S) := \{j \in V : j \prec i \text{ for some } i \in S\}$
- For every subset of nodes $S \subseteq V \setminus \{p, q\}$, the following inequality is valid for (1)-(6):

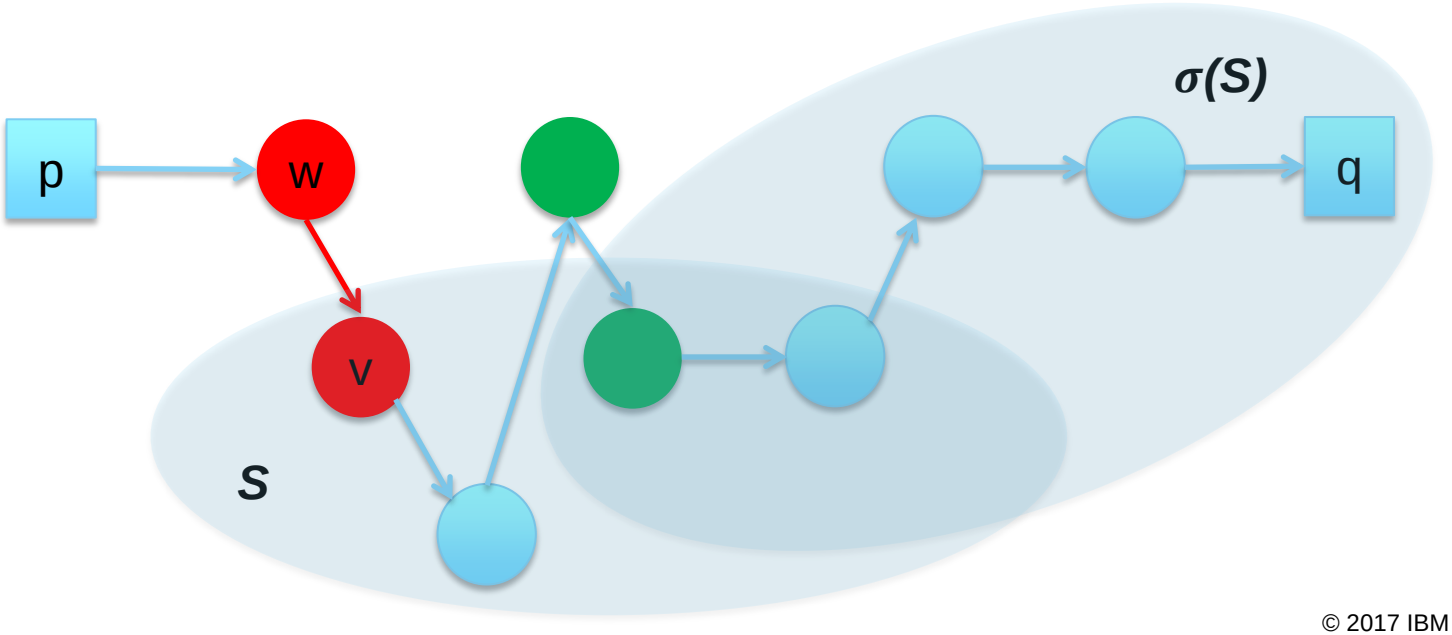
$$x\left(\delta(S \setminus \pi(S), \overline{S} \setminus \pi(S))\right) \geq 1 \qquad (10)$$



Successor inequalities (σ-ineqs) [Balas et al., 1995]

- For every node $i \in V$, define $\sigma(i) := \{j \in V \setminus \{i\} : i \prec j\}$
- For every subset of nodes $S \subseteq V$, define $\sigma(S) := \{j \in V : i \prec j \text{ for some } i \in S\}$
- For every subset of nodes $S \subseteq V \setminus \{p, q\}$, the following inequality is valid for (1)-(6):

$$x\left(\delta(\overline{S} \setminus \sigma(S), S \setminus \sigma(S))\right) \geq 1 \qquad (11)$$



Separating Subtour Elimination Constraints

Separation of SECs

- SECs are valid for ATSP (no time windows)
- Can be separated in polynomial time
  - Require the solution of $O(n)$ maxflow-mincut problems and thus can be separated in $O(n^4)$ time, see [Gomory and Hu, 1960, Padberg and Rinaldi, 1990]
- **Separation problem**:
  - Given a solution x^* of the LP relaxation of (1)-(6),
    - Find $S^* \subseteq V \setminus \{p, q\}$ with $x^*(\delta^+(S^*)) < 1$
    - Or prove that such S^* does not exist.

Separation of SECs: “Our” separation algorithm

- For every $s \in V \setminus \{p, q\}$
  - Find $S^* \subseteq V \setminus \{p, q\}$ with $\boldsymbol{s \in S^*}$, that minimizes $x^*(\delta^+(S))$
  - If $x^*(\delta^+(S^*)) < 1$, add the corresponding violated cut $x(\delta^+(S^*)) \geq 1$ to the problem

- Separation problem for a **fixed** $s \in V \setminus \{p, q\}$:
  - This is the problem of finding a mincut $\{s, t\}$ on the graph $G = (V, A)$, with
    - $t = q$
    - Capacities x^*_{ij} on the arcs
  - The mincut can be found in $O(n^3)$ time with ad-hoc algorithms
  - But we find it by solving an LP

Separating one SEC by solving a mincut problem

- Given a directed graph G = (V, A), with
  - A capacity $c_{ij} > 0$ associated with each arc $(i,j) \in A$
  - Two special nodes $s, t \in V$
- A cut in G is a set $S \subseteq V$ with $\boldsymbol{s} \in \boldsymbol{S}$, $\boldsymbol{t} \in \overline{S}$, and its capacity $z(S)$ is $c(\delta^+(S))$
- The mincut problem asks for a cut with minimum capacity

- **SEC separation problem for a fixed $\boldsymbol{s} \in V \setminus \{p, q\}$**
  1. From G = (V, A), build G* = (V, A*), where A* is obtained by removing from A
     - All arcs (i,j) with ${x_{ij}}^* = 0$
     - All arcs in $\delta^+(q)$, in $\delta^-(s)$, arc (s,q)
     - (Other reductions might be applied to further simplify the graph)
  2. Find a mincut S in G*, with arc capacities $c_{ij} := {x_{ij}}^*$
  3. If $x^*(\delta^+(S)) < 1$ in G (or, equivalent, if $z(S) + {x_{sq}}^* < 1$), then S gives a SEC violated by x^*

The mincut problem: LP formulation

- Continuous decision variables $u_i \ \ (i \in V)$ and $v_{ij} \ \ ((i,j) \in A^*)$
- The model then reads:

$$\min \ \ z = \sum_{(i,j) \in A^*} c_{ij} v_{ij} \tag{12}$$

$$-u_i + u_j + v_{ij} \geq 0, \qquad \forall \ (i,j) \ \in A^* \tag{13}$$

$$v_{ij} \geq 0, \qquad \forall \ (i,j) \ \in A^* \tag{14}$$

$$u_s = 1, \tag{15}$$

$$u_q = 0. \tag{16}$$

- **Key observation**:
  - All the vertices of the polyhedron defined by (13)-(16) are {0,1} – solutions
  - Given an optimal vertex (u^*, v^*) of (12)-(16), the corresponding mincut S can be obtained as

$$S := \{i \in V : \ {u_i}^* = 1\} \tag{17}$$

Separation of SECs: summary

- Given x^* and G = (V,A), for every fixed $s \in V \setminus \{p, q\}$
  1. Build G* = (V, A*), where A* is obtained by removing from A
     - All arcs (i, j) with $x_{ij}{}^* = 0$
     - All arcs in $\delta^+(q)$, in $\delta^-(s)$, arc (s, q)
  2. Solve (12)-(16) on G*
  3. If $z^* + x_{sq}{}^* < 1$, construct $S := \{i \in V : u_i{}^* = 1\}$ and add the corresponding violated SEC to the problem

- Trivial observation:
  - if $x_{sq}{}^* = 1$ steps 2 and 3 can be skipped as $z^* + x_{sq}{}^* \geq 1$ anyway.

Separating predecessor inequalities (π-ineqs)

Separation of predecessor inequalities (π-ineqs)

- π-ineqs exploit precedences inferred from the time windows
  - Complexity of the separation problem was never studied
  - Polynomial heuristics inspired to SECs separation are typically used [Ascheuer et al., 2001, Dash et al., 2012]
- We focus on the weak version of π-ineqs, namely:

$$x\big(\delta(S \setminus \pi(k), \overline{S} \setminus \pi(k))\big) \geq 1, \qquad k \in S \subseteq V \setminus \{p, q\} \tag{18}$$

- For (18), we can develop an exact separation algorithm that
  - Implicitly acts as an exact separation for SECs
  - Yields an effective heuristic for the general version of π-ineqs:

$$x\big(\delta(S \setminus \pi(S), \overline{S} \setminus \pi(S))\big) \geq 1, \qquad S \subseteq V \setminus \{p, q\} \tag{10}$$

Separating one weak π-ineq by solving a mincut problem

- Given a directed graph G = (V, A), with
  - A capacity $c_{ij} > 0$ associated with each arc $(i,j) \in A$
  - Two special nodes $s, t \in V$
- A cut in G is a set $S \subseteq V$ with $\boldsymbol{s \in S}$, $\boldsymbol{t} \in \overline{S}$, and its capacity $z(S)$ is $c(\delta^+(S))$
- The mincut problem asks for a cut with minimum capacity

- **Weak π-ineq separation problem for a fixed $\boldsymbol{k} \in V \setminus \{p, q\}$**
  1. From G = (V, A), build G* = (V*, A*), obtained by removing from G
     - All arcs (i,j) with $x_{ij}{}^* = 0$ (as for SEC separation)
     - All arcs in $\delta^+(q)$, in $\delta^-(k)$, arc (k,q) (as for SEC separation)
     - All **nodes in $\boldsymbol{\pi(k)}$** and all **arcs $\boldsymbol{(i,j)}$ with $\boldsymbol{i \in \pi(k)}$ OR $\boldsymbol{j \in \pi(k)}$**
  2. Find a mincut S in G*, with arc capacities $c_{ij} := x_{ij}{}^*$
  3. If $z(S) + x_{kq}{}^* < 1$, then $(S, \pi(k))$ give a weak π-ineq violated by x^*

Strengthening weak π-ineqs as general π-ineqs

- Given a node $k \in V \setminus \{p, q\}$ and a set $S \subseteq V \setminus \{p, q\}$, with $k \in S$,
  - The weak π-ineq $x\big(\delta(S \setminus \pi(k), \overline{S} \setminus \pi(k))\big) \geq 1$can always be strengthened as $x\big(\delta(S \setminus \pi(S), \overline{S} \setminus \pi(S))\big) \geq 1$
  - The strengthening is obvious as $\pi(k) \subseteq \pi(S)\ \forall\ k \in S$

- In particular, for a fixed $k \in V \setminus \{p, q\}$
  - The weak π-ineq associated with the mincut S in G* could be non-violated,
  - But the general π-ineq obtained by replacing $\pi(k)$ with $\pi(S)$ could be violated

- This gives rise to a simple heuristic for π-ineqs based on the exact separation of their weak counterpart.

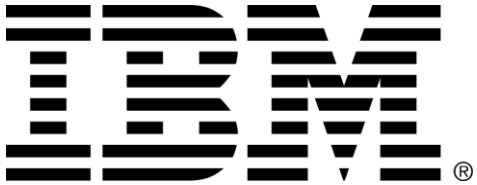
Heuristic separation of π-ineqs: summary

- Given $x^*$and G = (V,A), for every **fixed** $k \in V \setminus \{p, q\}$
  1. Build G* = (V*, A*), where G* is obtained by removing from G
     - All arcs (i, j) with $x_{ij}{}^* = 0$ (as for SEC separation)
     - All arcs in $\delta^+(q)$, in $\delta^-(k)$, arc (k, q) (as for SEC separation)
     - All **nodes in $\boldsymbol{\pi(k)}$** and all **arcs $\boldsymbol{(i,j)}$ with $\boldsymbol{i \in \pi(k)}$ $\boldsymbol{OR}$ $\boldsymbol{j \in \pi(k)}$**
  2. Solve (12)-(16) on G*
  3. Construct $S := \{i \in V : u_i{}^* = 1\}$ and construct $\pi(S)$
  4. If the corresponding π-ineq $x\big(\delta(S \setminus \pi(S), \overline{S} \setminus \pi(S))\big) \geq 1$ is violated by x^*, add it to the problem

- Trivial observations:
  - if $x_{kq}{}^* = 1$ steps 2 – 4 can be skipped (no violated cut with $k \in S$ can be found)
  - As already mentioned, the above heuristic implicitly acts as an exact separation for SECs

A few relevant references

1. E. Balas, M. Fischetti, W. Pulleyblank. The precedence constrained asymmetric traveling salesman polytope. Math. Program. 68, 241 – 265, 1995.
2. N. Ascheuer, M. Fischetti, M. Grötschel. A polyhedral study of the asymmetric traveling salesman problem with time windows. Networks 36, 69 – 79, 2000.
3. N. Ascheuer, M. Fischetti, M. Grötschel. Solving the asymmetric travelling salesman problem with time windows by branch-and-cut. Math. Prog. 90, 475 – 506, 2001.
4. F. Focacci, A. Lodi, M. Milano. A hybrid exact algorithm for the TSPTW. INFORMS J. Comput. 14, 403 – 417, 2002.
5. S. Dash, O. Günlük, A. Lodi, A. Tramontani. A time bucket formulation for the traveling salesman problem with time windows. INFORMS J. Comput. 24, 132 – 147, 2012.
6. R. Baldacci, A. Mingozzi, R. Roberti. New state-space relaxations for solving the traveling salesman problem with time windows. INFORMS J. Comput. 24, 356 – 371, 2012.



Legal Disclaimer