Cutting Planes: Theory, Algorithms, and Applications

A Introduction and Motivation

Cutting planes are a fundamental technique for solving integer programming (IP) problems. The method iteratively refines the feasible region of the linear programming (LP) relaxation by adding valid inequalities (cuts) that exclude fractional solutions but retain all integer feasible solutions. Cutting planes strengthen the LP relaxation, making the search for integer solutions more efficient and reducing the need for extensive branching.
Consider the integer program in minimization form:

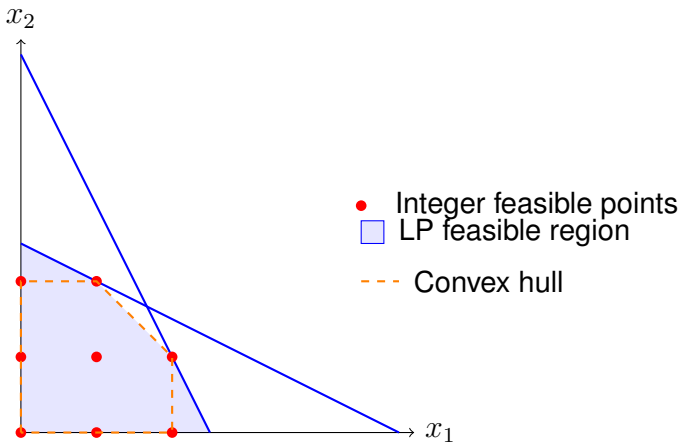
$$\begin{aligned} \text{minimize} \quad & c^T x \\ \text{subject to} \quad & Ax \geq b \\ & x \in \mathbb{Z}^n_+ \end{aligned}$$

The LP relaxation allows $x \in \mathbb{R}^n_+$. If the LP solution is not integer, a cutting plane is a valid inequality satisfied by all integer solutions but violated by the current LP solution.
Cutting planes work by iteratively shrinking the feasible region of the LP relaxation toward the convex hull of integer solutions. Each cut removes a portion of the fractional feasible region without excluding any integer feasible points. Geometrically, this process carves away the non-integral parts of the polyhedron, ideally leaving only the integer hull. In practice, a sequence of strong cuts can dramatically tighten the relaxation and reduce the need for branching.

A.1 Convex Hull

Consider the integer program:

$$\begin{aligned} \text{maximize} \quad & x_1 + x_2 \\ \text{subject to} \quad & 2x_1 + x_2 \leq 5 \\ & x_1 + 2x_2 \leq 5 \\ & x_1 \geq 0,\ x_2 \geq 0 \\ & x_1, x_2 \in \mathbb{Z} \end{aligned}$$



The red dots represent integer feasible solutions. The blue shaded triangle is the feasible region of the LP relaxation. The convex hull of the integer points is the smallest convex set containing all red dots. Note that the extreme points of the convex hull (the dashed orange polygon) are also integer feasible solutions, and the LP relaxation's feasible region is larger than the convex hull.

Definition 1 (Convex Hull)**.** *Given a set $S \subseteq \mathbb{R}^n$, the* convex hull *of S, denoted $\mathrm{conv}(S)$, is the set of all convex combinations of points in S:*

$$\mathrm{conv}(S) = \left\{ \sum_{i=1}^{k} \lambda_i x^{(i)} \;\middle|\; x^{(i)} \in S,\, \lambda_i \geq 0,\, \sum_{i=1}^{k} \lambda_i = 1,\, k \in \mathbb{N} \right\}$$

In the case of integer programming, the convex hull of the integer feasible solutions $S \subseteq \mathbb{Z}^n$ is a polytope defined by the valid inequalities (cuts) that can be derived from the constraints of the problem. The goal of cutting planes is to find these valid inequalities to tighten the LP relaxation.
Cutting planes are used to iteratively refine the LP relaxation by adding valid inequalities that exclude fractional solutions while retaining all integer feasible solutions. In theory, we will eventually reach the convex hull of the integer solutions, but in practice, we often stop when the LP solution is integer or when no more cuts can be generated. Getting the convex hull is often a computationally hard problem, and we may not be able to find all valid inequalities.

A.2 Definitions and Key Concepts

Definition 2 (Valid Inequality)**.** *A* valid inequality *is an inequality satisfied by all feasible integer solutions.*

Definition 3 (Cutting Plane)**.** *A* cutting plane *is a linear inequality satisfied by all integer feasible solutions but violated by the current LP solution.*

Contrary to valid inequalities, cutting planes are specifically designed to be violated by the current LP solution. Indeed, a valid inequality can be satisfied by the current LP solution, but a cutting plane must be violated to ensure that it effectively reduces the feasible region of the LP relaxation.

Definition 4 (Facet)**.** *A* facet *is a valid inequality that defines a face of the convex hull of integer solutions.*

A facet is a valid inequality that, when added to the LP relaxation, reduces the feasible region to a smaller polytope that still contains all integer solutions. Facets are particularly useful because they can significantly tighten the LP relaxation but they are not always easy to find.

Practice Exercise:

Consider the following integer program:

$$\begin{aligned} \text{minimize} \quad & 3x_1 + 2x_2 \\ \text{subject to} \quad & x_1 + x_2 \geq 3 \\ & 2x_1 + x_2 \leq 5 \\ & x_1, x_2 \in \mathbb{Z}_+ \end{aligned}$$

1. Solve the LP relaxation of the above integer program.
2. Identify grafically the feasible region of the LP relaxation.
3. Identify the convex hull of the integer feasible solutions.
4. Find the equation of the facets.

A.3 Implications of Efficient Convex Hull Computation

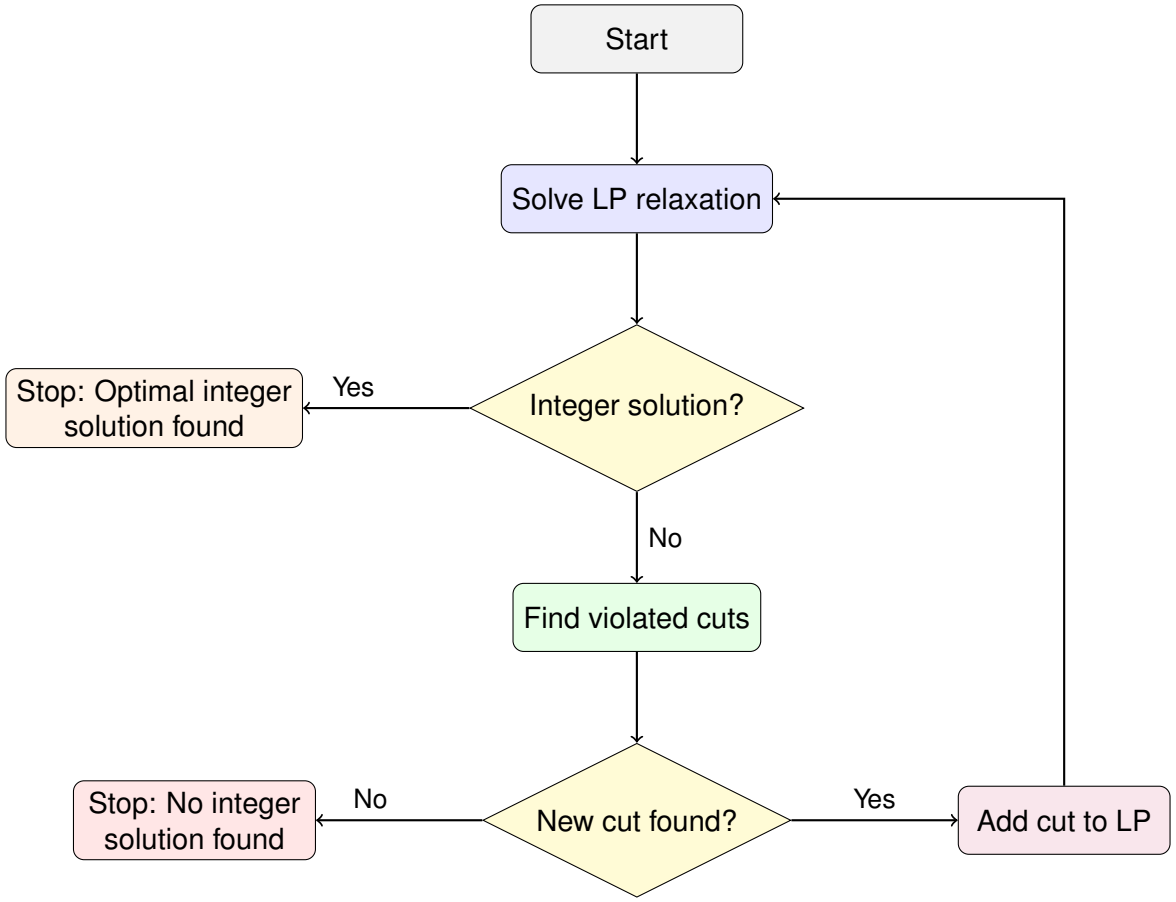
If we could efficiently compute the convex hull of the integer feasible solutions for any integer program, it would have profound consequences for the field of integer programming. The

convex hull provides the tightest possible linear relaxation of the IP: its extreme points are exactly the integer feasible solutions. This means that solving the linear program over the convex hull would yield an integer optimal solution directly, eliminating the need for branch-and-bound, cutting planes, or other combinatorial algorithms.
In complexity terms, this would imply that integer programming could be solved in polynomial time, since linear programming is polynomially solvable and the convex hull would be available explicitly. However, it is known that describing the convex hull of integer solutions can require an exponential number of inequalities in general, and that separating over the convex hull is NP-hard for many classes of IPs. Therefore, unless P=NP, there cannot exist a polynomial-time algorithm to find the convex hull for arbitrary integer programs.
In summary, efficient convex hull computation would make integer programming as tractable as linear programming, but this is believed to be impossible for general IPs due to computational complexity barriers.

B The Cutting Planes Algorithm

As shown in the figure below, the cutting planes algorithm iteratively refines the LP relaxation by solving the LP, checking for integrality, and generating cuts until an optimal integer solution is found or no more cuts can be generated. The general structure of the cutting planes algorithm can be summarized as follows:

1. Solve the LP relaxation of the integer program.
2. If the solution is integer, stop (optimal solution found).
3. Otherwise, find a violated cut (valid inequality) that excludes the current fractional solution.
4. Add the cut to the LP and resolve.
5. Repeat until an integer solution is found or no more cuts can be generated.



In practice, the effectiveness of the cutting planes algorithm depends on the strength of the generated cuts and the efficiency of the separation procedure. When combined with branch-and-bound (branch-and-cut), cutting planes are a cornerstone of modern integer programming solvers.

B.1 Finding Cuts: Separation

The core of the cutting planes algorithm is the *separation process* or *cutting plane generation*, where we identify valid inequalities that are violated by the current LP solution. This process is crucial for tightening the LP relaxation and guiding the search for an integer solution.
The separation problem is defined as follows: given a fractional solution to the LP relaxation, find a valid inequality that is violated by this solution. If such an inequality exists, it is added to the LP model, and the process continues until either an integer solution is found or no more cuts can be generated.
The separation algorithm can be thought of as a procedure that takes a fractional solution and either finds a violated cut or proves that no such cut exists within a certain class of inequalities. This is often done using combinatorial techniques, polyhedral theory, or specialized algorithms for specific classes of problems.
If the algorithm stops because no new cuts can be found, it means that our separation algorithm has exhausted the class of cuts we are considering, and we may need to either change our approach or accept that the current LP relaxation is as tight as possible given the cuts we have. Then we can either stop or continue with a different method, such as:

- Branch-and-cut: If the LP relaxation is not integer, we can branch on a fractional variable to explore the integer solution space and solve to optimality.
- Heuristic methods: If no cuts can be found, we may resort to heuristics to find a good integer solution quickly (Feasibility Pump, Local Search, etc.).
- Restarting with a different set of cuts: We can try a different class of cuts or a different separation algorithm to find cuts that were not captured by the previous method.
- Using a more powerful separation algorithm: If the current separation algorithm is not finding cuts, we can switch to a more sophisticated algorithm that are more computationally intensive but may yield better results.

B.2 Cut Acceptance Criteria

When generating cuts, it is essential to ensure that they are valid and do not exclude any integer feasible solutions. The acceptance criteria for cuts typically include:

- Validity: The cut must be satisfied by all integer feasible solutions.
- Separation: The cut must be violated by the current fractional solution.
- Minimality: The cut should be as tight as possible, meaning it should not exclude any additional integer solutions beyond the current fractional solution.

These criteria ensure that the cuts effectively tighten the LP relaxation without unnecessarily complicating the problem or excluding feasible integer solutions.
In practice, when deciding whether a cut should be accepted and added to the LP, a threshold parameter $\varepsilon > 0$ is often used to avoid adding cuts that are only marginally violated due to numerical errors or roundoff. Two common approaches are:

1. Measuring the Difference Between Left and Right Sides Given a cut in the form $a^T x \geq b$, and a current fractional solution x^*, the violation is measured as:

$$\Delta = b - a^T x^*$$

This criteria is the most used in the literature for its simplicity and effectiveness.

2. Measuring Distance to the Hyperplane Alternatively, the distance from x^* to the hyperplane $a^T x = b$ can be computed as:

$$d = \frac{b - a^T x^*}{\|a\|}$$

where $\|a\|$ is the Euclidean norm of a. If $d > \varepsilon$, the cut is violated by at least ε in normalized terms and is accepted. This approach is useful when the coefficients of the cut are large or when numerical stability is a concern, as it normalizes the violation relative to the magnitude of the coefficients.

C Gomory's Cuts

Gomory's method generates cuts from the final simplex tableau of the LP relaxation. Let B denote the set of basic variables and N the set of nonbasic variables in the optimal tableau. The variables are partitioned as $x = (x_B, x_N)$, where x_B are the basic variables and x_N are the nonbasic variables. The optimal solution to the LP relaxation is $x^* = (x_B^*, x_N^*)$.
Consider a row in the final simplex tableau corresponding to a basic variable x_{B_i} whose value $x_{B_i}^*$ is fractional:

$$x_{B_i} = b_i^* - \sum_{j \in N} a_{ij} x_j$$

where b_i^* is the right-hand side value for this row at optimality, and a_{ij} are the tableau coefficients for nonbasic variables x_j.
Let $f_i = b_i^* - \lfloor b_i^* \rfloor$ denote the fractional part of b_i^*. The Gomory fractional cut derived from this row is:

$$\sum_{j \in N} (a_{ij} - \lfloor a_{ij} \rfloor) x_j \geq f_i$$

Theorem 1 (Validity of Gomory Cuts)**.** *Every Gomory cut is a valid inequality for the integer program: it is satisfied by all integer feasible solutions and violated by the current fractional solution.*

Proof. Let x be any integer feasible solution. For each $j \in N$, x_j is integer, so $a_{ij} x_j = \lfloor a_{ij} \rfloor x_j + (a_{ij} - \lfloor a_{ij} \rfloor) x_j$. The tableau equation gives:

$$x_{B_i} = b_i^* - \sum_{j \in N} a_{ij} x_j$$

Since x_{B_i} and x_j are integer, the left side is integer, so the right side is integer. Thus,

$$b_i^* - \sum_{j \in N} a_{ij} x_j \in \mathbb{Z}$$

Rewriting $a_{ij} x_j$ as above:

$$b_i^* - \sum_{j \in N} \lfloor a_{ij} \rfloor x_j - \sum_{j \in N} (a_{ij} - \lfloor a_{ij} \rfloor) x_j \in \mathbb{Z}$$

Let $s = b_i^* - \sum_{j \in N} \lfloor a_{ij} \rfloor x_j$, which is real, and $t = \sum_{j \in N} (a_{ij} - \lfloor a_{ij} \rfloor) x_j$, which is real. Then $s - t \in \mathbb{Z}$, so $t = s - k$ for some $k \in \mathbb{Z}$.
But $s = \lfloor b_i^* - \sum_{j \in N} \lfloor a_{ij} \rfloor x_j \rfloor + f$, where f is the fractional part of $b_i^* - \sum_{j \in N} \lfloor a_{ij} \rfloor x_j$. Since b_i^* and all $\lfloor a_{ij} \rfloor x_j$ are integer, $f = f_i$.
Therefore,

$$\sum_{j \in N} (a_{ij} - \lfloor a_{ij} \rfloor) x_j = s - k \geq f_i$$

since $s - k$ is at least the fractional part f_i.
For the current fractional solution x^*, the tableau equation holds with $x_{B_i}^* = b_i^* - \sum_{j \in N} a_{ij} x_j^*$, and $x_j^* = 0$ for nonbasic variables, so

$$\sum_{j \in N} (a_{ij} - \lfloor a_{ij} \rfloor) x_j^* = 0 < f_i$$

since $0 < f_i$ (because $x_{B_i}^*$ is fractional). Thus, the cut is violated by x^*.
Therefore, the Gomory cut is valid for all integer solutions and violated by the current fractional solution. □

D Chvátal-Gomory Cuts

Chvátal-Gomory cuts generalize Gomory's approach by allowing cuts to be derived from any nonnegative linear combination of the original constraints, not just from the simplex tableau. The key idea is to take a nonnegative vector $\lambda \in \mathbb{R}_+^m$, form the linear combination $\lambda^T A x \leq \lambda^T b$, and then *round up* the right-hand side to obtain a valid inequality for the integer program.

Theorem 2 (Chvátal-Gomory (CG) Cut)**.** *Given an integer program* $\{x \in \mathbb{Z}^n : Ax \leq b\}$*, let* $\lambda \in \mathbb{R}_+^m$ *be any nonnegative vector. The inequality*

$$\lfloor \lambda^T A x \rfloor \leq \lfloor \lambda^T b \rfloor$$

is a valid inequality for the integer hull of the feasible region. This is called a Chvátal-Gomory cut*.*

Proof. Let the feasible region be the rational polyhedron:

$$P = \{x \in \mathbb{R}^n : Ax \leq b\}, \quad A \in \mathbb{Q}^{m \times n}, \, b \in \mathbb{Q}^m.$$

Then Let $\lambda \in \mathbb{R}_+^m$ be any nonnegative vector, we get for any $x \in P$:

$$\lambda^T A x \leq \lambda^T b$$

Since we are only interested in integer solutions $x \in \mathbb{Z}^n$, we can use the fact that for any real number α then $\lfloor \alpha \rfloor \leq \alpha$, and thus, for all $x \in P \cap \mathbb{Z}^n$:

$$\lfloor \lambda^T A \rfloor x \leq \lambda^T A x \leq \lambda^T b \Rightarrow \lfloor \lambda^T A \rfloor x \leq \lambda^T b$$

Since $x \in \mathbb{Z}^n$, the left-hand side $\lfloor \lambda^T A \rfloor x$ is an integer, and thus we can round the right-hand side up to obtain a valid Chvátal-Gomory cut:

$$\lfloor \lambda^T A \rfloor x \leq \lfloor \lambda^T b \rfloor$$

□

In practice, Chvátal-Gomory cuts are often derived from the constraints of the integer program or from the LP relaxation. The vector λ can be chosen based on the structure of the problem or the current solution and are generally constrained in the range $[0, 1)^m$ to ensure that the resulting cut is valid. However, the choice of λ can significantly affect the strength of the cut and they are not always easy to compute.

Relation to Gomory cuts: Chvátal-Gomory cuts can be seen as a generalization of Gomory cuts. If we take λ to be the vector corresponding to a basic variable in the simplex tableau, then the CG cut reduces to a Gomory cut. However, CG cuts can be derived from any combination of constraints, not just those in the tableau.

Note: If $A \in \mathbb{Z}^{m \times n}$ and $b \in \mathbb{Z}^m$, then valid Chvátal-Gomory cuts arise only for $\lambda \in [0, 1)^m$.

Example: Suppose we have constraints

$$4x_1 + 2x_2 \leq 7$$
$$2x_1 + 4x_2 \leq 7$$

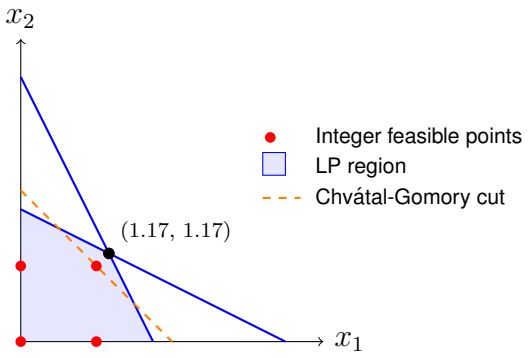
Let $\lambda = (0.2, 0.2)$. Then

$$0.2(4x_1 + 2x_2) + 0.2(2x_1 + 4x_2) = (1.2)x_1 + (1.2)x_2 \leq 0.2 \cdot 7 + 0.2 \cdot 7 = 2.8$$

Rounding down gives the Chvátal-Gomory cut:

$$x_1 + x_2 \leq 2$$

This cut is not satisfied by the fractional solution $(x_1, x_2) = (1.16, 1.16)$, but it is satisfied by all integer feasible solutions, such as $(0, 2)$, $(1, 1)$, and $(2, 0)$.



E Cover Cuts

Cover cuts are a specific type of cutting plane that arise from the concept of *covering* in combinatorial optimization. A cover is a subset of variables that, when set to 1, would violate the constraints of the integer program. Cover cuts are derived from these covers and are used to eliminate fractional solutions that do not correspond to any integer feasible solution.

Definition 5 (Cover)**.** *A* cover *is a subset of variables $C \subseteq \{1, \ldots, n\}$ such that if all variables in C are set to 1, the resulting solution violates at least one constraint of the integer program.*

Definition 6 (Cover Cut)**.** *A* cover cut *is a valid inequality derived from a cover that excludes fractional solutions while retaining all integer feasible solutions. It is typically of the form:*

$$\sum_{j \in C} x_j \leq |C| - 1$$

where $|C|$ is the size of the cover.

Cover cuts are especially powerful when applied to *knapsack constraints*, which have the form:

$$\sum_{j=1}^{n} a_j x_j \leq b, \quad x_j \in \{0,1\}$$

where $a_j > 0$ and b is a positive integer. A *cover* for this constraint is a subset $C \subseteq \{1,\ldots,n\}$ such that:

$$\sum_{j \in C} a_j > b$$

That is, if all variables in C are set to 1, the constraint is violated.
The *cover cut* associated with C is:

$$\sum_{j \in C} x_j \leq |C| - 1$$

This inequality is valid for all integer feasible solutions, since at most $|C| - 1$ variables in C can be set to 1 without violating the knapsack constraint.

Example: Consider the knapsack constraint:

$$7x_1 + 6x_2 + 4x_3 + 3x_4 \leq 14, \quad x_j \in \{0,1\}$$

Let $C = \{1,2,3\}$. Since $7+6+4 = 17 > 14$, we have a cover. The cut derived from this cover is:

$$x_1 + x_2 + x_3 \leq 2$$

This cut eliminates fractional solutions such as $x_1 = x_2 = x_3 = 0.8$, which would otherwise satisfy the original constraint but are not integer feasible.

Lifting: Cover cuts can be *lifted* to strengthen them by including variables not in C with appropriate coefficients. For example, for $j \notin C$, we can determine the largest coefficient α_j such that:

$$\sum_{i \in C} x_i + \alpha_j x_j \leq |C| - 1$$

remains valid for all integer feasible solutions. Lifting procedures can yield stronger cuts and further tighten the LP relaxation.
Let C be a minimal cover for the knapsack constraint $\sum_{i=1}^{n} a_i x_i \leq b$. For $j \notin C$, define $C' = C \cup \{j\}$. To find α_j, we can use the Balas lifting procedures. For that we need to define $S(k)$ as the sum of the k largest coefficients in $\{a_i\}_{i \in C}$, then we can set:

$$\alpha_j = k \in \mathbb{Z}_+ \text{ such that } S(k) \leq a_j \leq S(k+1)$$

This process can be repeated for each variable not in C to obtain the strongest possible cut.

Example: Consider the knapsack constraint:

$$15x_1 + 13x_2 + 9x_3 + 8x_4 + 8x_5 + 8x_6 + 5x_7 + 5x_8 + 5x_9 + 5x_{10} \leq 16, \quad x_j \in \{0,1\}$$

Then $C = \{7,8,9,10\}$ is a cover since $5+5+5+5 = 20 > 16$. The cover cut is:

$$x_7 + x_8 + x_9 + x_{10} \leq 3$$

Now, lift x_1 (not in C):
We have $S(0) = 0$, $S(1) = 5$, $S(2) = 10$, $S(3) = 15$, and $S(4) = 20$. Since $S(3) \leq a_1 \leq S(4)$, we can set $\alpha_1 = 3$. The lifted cover cut becomes:

$$x_7 + x_8 + x_9 + x_{10} + 3x_1 \leq 3$$

Note: There are many different lifting procedures to compute the coefficients α_j in the literature, each with its own strengths and computational considerations. A detailed discussion of these methods is beyond the scope of this document; interested readers are encouraged to consult the literature for further study.

F Problem Specific Cuts

In the literature, many classes of cutting planes have been developed that are tailored to specific families of integer programming problems. These *problem-specific cuts* exploit the combinatorial structure of the underlying problem to generate strong valid inequalities, often accompanied by efficient separation algorithms.

A prominent example is the **Traveling Salesman Problem (TSP)**. The TSP can be formulated as an integer program where binary variables x_{ij} indicate whether the undirected edge (i, j) is included in the tour:

$$\begin{aligned} \text{minimize} \quad & \sum_{(i,j)\in E} c_{ij} x_{ij} \\ \text{subject to} \quad & \sum_{j:(i,j)\in E} x_{ij} = 2 \quad \forall i \in V \\ & x_{ij} \in \{0,1\} \quad \forall (i,j) \in E \end{aligned}$$

The degree constraints ensure that each node has exactly two incident edges, but the LP relaxation admits solutions that correspond to multiple disconnected cycles (subtours).

To eliminate these, *Subtour Elimination Constraints (SEC)* are introduced:

$$\sum_{i,j\in S,\, i<j} x_{ij} \leq |S| - 1 \qquad \forall\, S \subset V,\ 2 \leq |S| \leq |V| - 1$$

These constraints ensure that no proper subset of nodes forms a closed cycle, thus enforcing connectivity.

The separation problem for SECs—finding a violated subtour constraint given a fractional solution—can be solved efficiently by computing minimum cuts (max flows) in the support graph of the current solution. If a subset S is found such that the sum of x_{ij} over edges in S exceeds $|S| - 1$, the corresponding SEC is violated and can be added as a cut.

Many other problem-specific cuts exist, such as clique cuts for graph problems, comb inequalities for TSP, and flow cover cuts for network design. These cuts, together with their specialized separation algorithms, are crucial for solving large-scale integer programs efficiently in practice.