

B6.3 Integer Programming

Raphael Hauser

Mathematical Institute
University Of Oxford

Michaelmas Term 2018

1 The Branch & Cut Framework

2 Cover Inequalities

The Branch & Cut Framework

Data Specification (Branch & Cut)

Input data:

objective $\max c^T x$;

constraints $Ax = b$ with A, b integer valued, defining feasible set $\mathcal{X} = \{x \in \mathbb{Z}^n : Ax = b, x \geq 0\}$;

Global data:

global pool of valid inequalities $A^{(0)}x \leq b^{(0)}$ for \mathcal{X} ;

global dual bound \bar{z} ;

global primal bound \underline{z} ;

list AN of active nodes;

node counter k ;

Local (node level) data:

local inequalities $A^{[j]}x \leq b^{[j]}$;

local polyhedron $\mathcal{P}^{[j]}$, defining node feasible set $\mathcal{X}^{[j]} = \mathcal{P}^{[j]} \cap \mathbb{Z}^n$;

node dual bound $\bar{z}^{[j]}$;

node primal bound $\underline{z}^{[j]}$;

Methods (Branch & Cut)

```

function  $[x^{[j]}, \mathcal{P}^{[j]}] = \text{addCuts}(x^{[j]}, \mathcal{P}^{[j]})$ 
// input  $x^{[j]} = \arg \max\{c^T x : x \in \mathcal{P}^{[j]}\}$  optimised via simplex
while  $\mathcal{P}^{[j]} \neq \emptyset$ ,  $x^{[j]}$  fractional and  $\exists$  sufficiently deep cut in pool do
    find cut  $A_i^{[0]} x^{[j]} > b_i^{[0]}$  in pool of valid inequalities;
     $\mathcal{P}^{[j]} \leftarrow \mathcal{P}^{[j]} \cap \{x : A_i^{[0]} x \leq b_i^{[0]}\}$ ;
    re-optimize  $x^{[j]}$  via dual simplex;
end
while  $\mathcal{P}^{[j]} \neq \emptyset$ ,  $x^{[j]}$  fractional and sufficiently deep new cuts generated do
    generate cut  $\alpha^T x^{[j]} > \alpha_0$  with  $\alpha^T x \leq \alpha_0$  a valid inequality for  $\mathcal{X}$ ;
     $\{A^{[0]} x \leq b^{[0]}\} \leftarrow \{A^{[0]} x \leq b^{[0]}\} \cup \{\alpha^T x \leq \alpha_0\}$ ; // add valid inequality to pool
     $\mathcal{P}^{[j]} \leftarrow \mathcal{P}^{[j]} \cap \{x : \alpha^T x \leq \alpha_0\}$ ;
    re-optimize  $x^{[j]}$  via dual simplex;
end
if  $\mathcal{P}^{[j]} = \emptyset$  then
     $x^{[j]} = \text{NaN}$ ;
end

```

Algorithm (Branch & Cut)

```

k = 1, A[1], b[1] = 0, AN = {1}, z = -∞, z̄ = +∞, x* = NaN, {A[0]x ≤ b[0]} = 0; // initialisation
while AN ≠ ∅ do
  choose j ∈ AN;
  if z̄[j] ≤ z then
    AN ← AN \ {j}; // pruning
  else
    P[j] = {x ∈ ℝn : Ax = b, A[j]x ≤ b[j], x ≥ 0};
    x[j] := arg max{cTx : x ∈ P[j]}; // simplex warmstart from parent's initial subproblem
    [x[j], P[j]] = addCuts(x[j], P[j]);
    z̄[j] := cTx[j]; // z̄[j] := -∞ if x[j] = NaN
    z̄ := max{z̄[ℓ] : ℓ ∈ AN}; // update global upper bound
    if x[j] integer valued then y[j] := x[j];
    else attempt to find y[j] ∈ X[j] via heuristic;
    z[j] := cTy[j]; // set z[j] := -∞ if y[j] unassigned
    if z[j] > z then
      x* := y[j]; // update incumbent
      z := z[j]; // update global lower bound
    end
    if ∃ xℓ[j] fractional then
      {A[k+i]x ≤ b[k+i]} := {A[j]x ≤ b[j]} ∪ {
        {x ≤ ⌊xℓ[j]⌋}, (i = 1),
        {x ≥ ⌈xℓ[j]⌉}, (i = 2)
      };
      z̄[k+i] := z̄[j], (i = 1, 2); // inherit dual bound from parent
      AN ← (AN \ {j}) ∪ {k + 1, k + 2}, k ← k + 2; // branching
    end
  end
end
end

```

Algorithm Design Steps

Adapting the branch-and-cut framework to a specific problem class involves the following steps:

- 1 Identification of structural properties of the problem class.
- 2 Use polyhedral analysis to translate the structural properties into classes of valid inequalities to be used as cuts.
- 3 For each class C of valid inequalities, identify an efficient procedure to solve the associated separation problem: Given $x^* \notin \mathcal{X}$, find a valid inequality $\alpha^T x \leq \alpha_0$ for \mathcal{X} among class C such that $\alpha^T x^* > \alpha_0$.

Example (Search index construction)

The following example is due to Fischetti:

Relational data bases use a small number of search indices to use in queries of m different types. We need to choose which among a set of candidate indices $\{1, \dots, n\}$ to build and maintain so as to minimise the expected cost. $j = 0$ represents a default index available without fixed cost:

$$\min_{\mathbf{y}, \mathbf{x}} \sum_{j=1}^n c_j y_j + \sum_{i=1}^m \sum_{j=0}^n \gamma_{ij} x_{ij}$$

$$\text{s.t. } \sum_{j=1}^n d_j y_j \leq D$$

$$\sum_{j=0}^n x_{ij} = 1, \quad (i = 1, \dots, m)$$

$$\sum_{i=1}^m x_{ij} \leq m y_j, \quad (j = 1, \dots, n)$$

$$x_{ij}, y_j \in \{0, 1\}, \quad (i = 1, \dots, m; j = 0, \dots, n).$$

For example, $n = 5$, $m = 6$, $D = 19$,

$$[\gamma_{ij}] = \begin{bmatrix} 6200 & 1300 & 6200 & 6200 & 6200 & 6200 \\ 2000 & 900 & 700 & 2000 & 2000 & 2000 \\ 800 & 800 & 800 & 800 & 800 & 800 \\ 6700 & 6700 & 6700 & 1700 & 6700 & 2700 \\ 5000 & 5000 & 5000 & 2200 & 1200 & 4200 \\ 2000 & 2000 & 2000 & 2000 & 2000 & 750 \end{bmatrix}, \quad [c_j] = \begin{bmatrix} 200 \\ 1200 \\ 400 \\ 2400 \\ 250 \end{bmatrix}, \quad [d_j] = \begin{bmatrix} 10 \\ 5 \\ 10 \\ 8 \\ 6 \end{bmatrix}$$

Example (Search index construction continued)

The optimal solution (x^*, y^*) of the LP relaxation can never give a tight dual bound, because the big-M constraint $\sum_{i=1}^m x_{ij}^* \leq m y_j^*$ and the optimality of y^* imply that

$$y_j^* = \frac{1}{m} \sum_{i=1}^m x_{ij}^*$$

is always fractional, unless all $x_{ij} = 1$ for the same index j .

Variable fixing: $\gamma_{ij} \geq \gamma_{i0} \Rightarrow x_{ij} = 0$. This removes all variables x_{ij} bar $x_{10}, \dots, x_{60}, x_{11}, x_{21}, x_{22}, x_{43}, x_{53}, x_{54}, x_{45}, x_{55}, x_{65}$, and it allows to tighten the big-M constraints to

$$\sum_{i=1}^m x_{ij}^* \leq |I_j| y_j,$$

where $|I_j| = \{i : \gamma_{ij} < \gamma_{i0}\}$.

Now solving the LP relaxation yields the primal bound 8,940 and the following non-zero values $x_{20}^* = 6/10, x_{30}^* = 1, y_1^* = 7/10, x_{11}^* = 1, x_{21}^* = 4/10, y_3^* = 1, x_{43}^* = x_{53}^* = 1, y_5^* = 1/3, x_{65}^* = 1$.

Example (Search index construction continued)

In a similar discussion of the UFL we previously noticed that the big-M constraint $\sum_{i=1}^m x_{ij} \leq |I_j|y_j$ may be replaced by the *strong formulation* given by the valid inequalities

$$\text{Class C1: } x_{ij} \leq y_j, \quad (j \in [1, n]; i \in I_j).$$

Rather than imposing all of these constraints at each node of the branch-and-cut algorithm, we only use these as cuts when needed. In our case, the valid inequality $x_{11} \leq y_1$ is a cut of x^* , because $x_{11}^* = 1 > y_1^*$.

The separation problem associated with class C1 is straightforward to solve via enumeration and scanning. In addition to the cut $x_{11} \leq y_1$, we also find the cut $x_{65} \leq y_5$.

After adding the two cuts to the formulation, we re-optimize the LP solution via dual simplex pivots and obtain the primal bound 9,900 and the following nonzero components: $x_{30}^* = 1$, $x_{60}^* = 3/4$, $y_1^* = 1$, $x_{11}^* = x_{21}^* = 1$, $y_3^* = 3/4$, $x_{43}^* = 3/4$, $x_{53}^* = 3/4$, $y_5^* = 1/4$, $x_{45}^* = x_{55}^* = x_{65}^* = 1$.

None of the inequalities of class C1 are violated. However, note that $y_1 + y_3 \leq 1$ is a valid inequality, because $d_1 + d_3 > 19$, hence this is a cut for (x^*, y^*) . More generally, we consider the following valid inequalities

$$\text{Class C2: } \sum_{j \in S} y_j \leq |S| - 1, \quad \forall S \subseteq [1, n] \text{ s.t. } \sum_{j \in S} d_j > D.$$

Example (Search index construction continued)

To solve the separation problem associated with class C2 and find the indicator vector $z \in \{0, 1\}^n$ of a set S such that

$$\sum_{j=1}^n y_j^* z_j = \sum_{j \in S} y_j^* > |S| - 1 = \sum_{j=1}^n z_j - 1,$$
$$\sum_{j=1}^n d_j z_j = \sum_{j \in S} d_j \geq D + \varepsilon$$

for some sufficiently small ε .

We can find the deepest cut from this class by solving the knapsack problem

$$w^* = \min_{\mathbf{z}} \sum_{j=1}^n (1 - y_j^*) z_j$$
$$\text{s.t. } \sum_{j=1}^n d_j z_j \geq D + \varepsilon,$$
$$z_j \in \{0, 1\}, \quad (j \in [1, n])$$

and checking if $w^* < 1$. The knapsack problem is quite easy to solve via branch-and-bound and allows for variable fixing $z_j = 1$ if $y_j = 1$, which reduces the number of variables.

Example (Search index construction continued)

Adding the cut $y_1 + y_3 \leq 1$ from class C2 to the formulation and re-optimising the LP relaxation via dual simplex iterations yields the primal bound 10,880 and non-zero variables $x_{30}^* = 1$, $y_1^* = 1$, $x_{11}^* = x_{21}^* = 1$, $y_4^* = 3/8$, $x_{54}^* = 3/8$, $y_5^* = 1$, $x_{45}^* = x_{65}^* = 1$, $x_{55}^* = 5/8$.

Class C1 does not yield any violated inequalities, but from C2 we find the cut $y_1 + y_4 + y_5 \leq 2$. Adding this inequality to the formulation and re-optimising the LP yields the primal bound 11,100 and the non-zero variables $x_{30}^* = 1$, $y_1^* = 1$, $x_{11}^* = x_{21}^* = 1$, $y_5^* = 1$, $x_{45}^* = x_{55}^* = x_{65}^* = 1$, which is integral valued and hence (IP)-optimal.

In general, using cuts from classes C1 and C2 in the branch-and-cut framework reduces the number of nodes generated by 2 or 3 orders of magnitude over straight-forward branch-and-bound when applied to larger search index construction problems.

Class C2 can be generalised to other binary programming problems, where they are known as *cover inequalities*.

Cover Inequalities

Definition (Cover Inequality)

Let $a^T x \leq r$ with $a \in \mathbb{R}_+^n$, $r \in \mathbb{R}_+$ be a constraint on binary decision variables $x_j \in \{0, 1\}$. A *cover* of this constraint is a subset $C \subseteq \{1, \dots, n\}$ such that $\sum_{j \in C} a_j > r$.

A cover is *minimal* if no strict subset $S \subsetneq C$ is a cover.

The cover inequality associated with a cover C is the inequality $\sum_{j \in C} x_j \leq |C| - 1$.

Proposition

- The cover inequality $\sum_{j \in C} x_j \leq |C| - 1$ associated with a cover C of the constraint $a^T x \leq r$ is a valid inequality for $\{x \in \{0, 1\}^n : a^T x \leq r\}$.
- If $S \subset C$ is also a cover, then the cover inequality $\sum_{j \in S} x_j \leq |S| - 1$ is stronger than the cover inequality associated with C .
- Only cover inequalities associated with minimal covers are essential.

Proof. i) Suppose to the contrary that $\exists x \in \{0, 1\}^n$ such that $a^T x \leq r$ and $\sum_{j \in C} x_j \geq |C|$. Since x is binary, this implies $x_j = 1 \forall j \in C$, and then, using $a_j \geq 0$,

$$a^T x \geq \sum_{j \in C} a_j x_j = \sum_{j \in C} a_j > r. \quad \text{!}$$

ii) We need to show that for $x \in \{0, 1\}^n$, $\sum_{j \in S} x_j \leq |S| - 1$ implies $\sum_{j \in C} x_j \leq |C| - 1$ (but not necessarily the other way round). Let $S_0^C = \{j \in C \setminus S : x_j = 0\}$ and $S_1^C = \{j \in C \setminus S : x_j = 1\}$. Then

$$\sum_{j \in C} x_j = \sum_{j \in S} x_j + \sum_{j \in S_1^C} x_j \leq |S| - 1 + |S_1^C| \leq |C| - 1.$$

iii) If the cover C is not minimal, then there exists a cover $S \subsetneq C$, and by Part ii) the cover inequality associated with S renders the inequality $\sum_{j \in C} x_j \leq |C| - 1$ redundant.

Algorithm (Minimal cover separation)

Input: $a \in \mathbb{R}_+^n$, $r \in \mathbb{R}_+$, $x^* \in \{0, 1\}^n$ such that $a^T x^* > r$;

Output: minimal cover C whose associated cover inequality is a cut for x^* ;

Initialise: $C = \emptyset$, $\ell = 1$;

compute list $I = \{j_1, \dots, j_k\} = \{j : x_j^* > 0\}$ ordered such that $a_{j_1} \geq \dots \geq a_{j_k}$;

while $\ell \leq k$ **and** $\sum_{j \in C} a_j \leq r$ **do**

$C \leftarrow C \cup \{j_\ell\}$;
 $\ell \leftarrow \ell + 1$;

end

Definition (Extended cover Inequality)

Let $a^T x^* > r$ with $a \in \mathbb{R}_+^n$, $r \in \mathbb{R}_+$, and let $\sum_{j \in C} x_j \leq |C| - 1$ be the cover inequality with the minimal cover C constructed by Algorithm (Minimal cover separation), and $a^* := a_{\mathbf{1}} = \max_{j \in C} a_j$. Let $E = \{j : a_j > a^*\}$. The associated *extended cover inequality* is defined as $\sum_{j \in C \cup E} x_j \leq |C| - 1$.

Note that, although $C \cup E$ is a cover, the extended cover inequality is not the cover inequality associated with $C \cup E$, because the right-hand side is $|C| - 1$, not $|C \cup E| - 1$.

Proposition

- i) The extended cover inequality $\sum_{j \in C \cup E} x_j \leq |C| - 1$ is a valid inequality for the set $\{x \in \{0, 1\}^n : a^T x \leq r\}$.
- ii) The extended cover inequality is stronger than the cover inequality $\sum_{j \in C} x_j \leq |C| - 1$.

Proof. i) Suppose to the contrary that $\exists x \in \{0, 1\}^n$ such that $a^T x \leq r$ and $\sum_{j \in C \cup E} x_j \geq |C|$. Then x is a binary vector with at least $|C|$ components equal to 1. Using $a_j, x_j \geq 0 \forall j$ and $a_j > a^* \forall j \in E$ by construction, we find

$$a^T x \geq \sum_{j \in C \cup E} a_j x_j \geq \sum_{j \in C} a_j > r. \quad \text{!}$$

ii) We need to show that for $x \in \mathbb{R}_+^n$, $\sum_{j \in C \cup E} x_j \leq |C| - 1$ implies $\sum_{j \in C} x_j \leq |C| - 1$ (but not necessarily the other way round). This follows trivially from the disjointness of C and E and $x_j \geq 0$.

Lifted Cover Inequalities

Lemma (Lifting valid inequalities)

Let $S := \{j_1, \dots, j_t\} \subset [1, n]$, $\alpha_0, \alpha_{j_s} \geq 0$, ($s = 1, \dots, t$), $a \in \mathbb{R}_+^n$ and $r \in \mathbb{R}_+$ be given such that $\sum_{s=1}^t \alpha_{j_s} x_{j_s} \leq \alpha_0$ is a valid inequality for the set $\{x \in \{0, 1\}^n : a^T x \leq r\}$. Let $j_{t+1} \in [1, n] \setminus S$ and

$$\zeta_{t+1} := \max \sum_{s=1}^t \alpha_{j_s} x_{j_s}$$


$$\text{s.t. } \sum_{s=1}^t a_{j_s} x_{j_s} + a_{j_{t+1}} \leq r,$$

$$x_{j_s} \in \{0, 1\}, \quad (s = 1, \dots, t).$$

Then for every $\alpha_{j_{t+1}} \in [0, \alpha_0 - \zeta_t]$ the lifted inequality $\sum_{s=1}^{t+1} \alpha_{j_s} x_{j_s} \leq \alpha_0$ is valid for the set $\{x \in \{0, 1\}^n : a^T x \leq r\}$. Furthermore, the larger α_{t+1} , the stronger the inequality.

Proof. Let $x \in \{0, 1\}^n$ be such that $a^T x \leq r$. If $x_{j_{t+1}} = 0$, then the lifted inequality follows from the validity of the unlifted inequality. If $x_{j_{t+1}} = 1$, then x satisfies the constraints of the optimisation problem, and hence, $\sum_{s=1}^t \alpha_{j_s} x_{j_s} \leq \zeta_{t+1}$, with $\zeta_{t+1} \geq 0$ because $\alpha_{j_s}, x_{j_s} \geq 0$, ($s = 1, \dots, t$). Therefore,

$$\sum_{s=1}^t \alpha_{j_s} x_{j_s} + \alpha_{j_{t+1}} x_{j_{t+1}} \leq \zeta_{t+1} + \alpha_0 - \zeta_{t+1} = \alpha_0.$$

Furthermore, if $0 \leq \alpha^{[1]} < \alpha^{[2]} \leq \alpha_0 - \zeta_t$, then for $x \in \mathbb{R}_+^n$, $\sum_{s=1}^t \alpha_{j_s} x_{j_s} + \alpha^{[2]} x_{j_{t+1}} \leq \alpha_0$ implies $\sum_{s=1}^t \alpha_{j_s} x_{j_s} + \alpha^{[1]} x_{j_{t+1}} \leq \alpha_0$, but not necessarily the other way round. 

Let $x^* \in \mathbb{R}_+^n$ be such that $a^T x^* > r$ with $a \in \mathbb{R}_+^n, r \in \mathbb{R}_+$. We would like to use a tight lifted cover inequality to cut x^* from the set $\{x \in \{0, 1\}^n : a^T x \leq r\}$. The following algorithm solves this separation problem:

Algorithm (Separation by lifted cover inequalities)

using Algorithm (Minimal cover separation), find minimal cover inequality $\sum_{j \in C} x_j \leq |C| - 1$;

fix an ordering j_1, \dots, j_p of $[1, n] \setminus C$;

for $t = 1$ to p do

 using branch & bound, solve knapsack problem

$$\zeta_t := \max \sum_{s=1}^{t-1} \alpha_{j_s} x_{j_s} + \sum_{j \in C} x_j$$

$$\text{s.t. } \sum_{s=1}^{t-1} a_{j_s} x_{j_s} + \sum_{j \in C} a_j x_j + a_{j_t} \leq r,$$

$$x_j \in \{0, 1\}, \quad (j \in C \cup \{j_s : s \in [1, t-1]\});$$

$$\alpha_{j_t} := |C| - 1 - \zeta_t;$$

end

Proof of Correctness: By construction, the cover inequality $\sum_{j \in C} x_j \leq |C| - 1$ is a cut for x^* , and applying Lemma (Lifting valid inequalities) recursively, it follows that the lifted cover inequality $\sum_{s=1}^p \alpha_{j_s} x_{j_s} + \sum_{j \in C} x_j \leq |C| - 1$ is a valid inequality stronger than the cover inequality. Hence, it is also a cut for x^* .

Example (Generalised Assignment Problem by Branch & Cut)

A fairly general class of IPs is the Generalised Assignment Problem (GAP) that takes the following form, which uses a combination of knapsack and assignment constraints:

$$(GAP) \quad \max_x \sum_{i=1}^m \sum_{j=1}^n p_{ij} x_{ij}$$

$$(1) \quad \text{s.t.} \quad \sum_{j=1}^n c_{ij} x_{ij} \leq b_i, \quad (i = 1, \dots, m)$$

$$(2) \quad \sum_{i=1}^m x_{ij} = 1, \quad (j = 1, \dots, n)$$

$$x_{ij} \in \{0, 1\}, \quad (i = 1, \dots, m; j = 1, \dots, n).$$

GUB/SOS branching (see Lecture 9) on the assignment constraints (2) ensures that the feasible sets of subproblems are balanced (children of the same parent node have approximately equal cardinality).

Cuts for optimal solutions x^* of LP relaxed subproblems can be constructed in the form of lifted cover inequalities deriving from the knapsack constraints (1).

The branch & cut algorithm solves the problem in two to three orders of magnitude fewer nodes than the branch & bound approach.