InFoMM – Optimisation
Lecture 6

Raphael Hauser

Oxford Mathematical Institute

Michaelmas Term 2019

Data Specification (General B&B)

Input:

objective $x \mapsto f(x)$;
implicit description of feasible set $x \in \mathscr{F}$ of root problem $z = \max\{f(x) : x \in \mathscr{F}\}$;

Working Data:

list AN of active nodes;
global primal bound $\underline{z} \leq z$;
global dual bound $\overline{z} \geq z$;
incumbent x^*, i.e. best solution found so far;
subproblem primal bound $\underline{z}^{[j]} \leq z^{[j]} := \max\{f(x) : x \in \mathscr{F}_j\}$;
subproblem dual bound $\overline{z}^{[j]} \geq z^{[j]}$;

Outputs:

global bounds $\underline{z}$, $\overline{z}$ to provide certificate of optimality (when $\underline{z} = \overline{z}$) or approximation guarantee;
incumbent x^* as optimal solution (or solution with approximation guarantee when $\underline{z} < \overline{z}$);

Methods (General B&B)

Required Methods:

method to compute dual bounds by solving an easy problem

$$(\mathscr{R}_j) \quad \bar{z}^{[j]} = \max\{g_j(x) : x \in \mathscr{R}_j\}; \quad \texttt{//e.g., } g_j(x) \geq f(x) \texttt{ and } \mathscr{R}_j \supset \mathscr{F}_j \texttt{ (relaxation)}$$

branching rule to split $\mathscr{F}_j = \mathscr{F}_{j_1} \cup \cdots \cup \mathscr{F}_{j_k}$ so that subproblems

$$(\mathscr{F}_j) \quad \max\{f(x) : x \in \mathscr{F}_j\}$$

all fall into the same problem class as root problem $(\mathscr{F})$ to which above methods apply;

Optional Methods:

heuristic to find $y \in \mathscr{F}_j$ and compute primal bounds $\underline{z}^{[j]} = f(y)$;
selection rule for $(\mathscr{F}_j) \in$ **AN**;

Algorithm (General B&B)

```
$AN = \{(\mathscr{F})\}$; $\underline{z} = -\infty$; $\overline{z} = +\infty$; $x^* = \mathrm{NaN}$; // initialisation
while $AN \neq \emptyset$ do
    choose $(\mathscr{F}_j) \in AN$;
    if $\overline{z}^{[j]} \leq \underline{z}$ then
        $AN := (AN \setminus \{(\mathscr{F}_j\}))$; // prune by bound
    else
        solve $x^{[j]} = \arg\max\{g_j(x) : x \in \mathscr{R}_j\}$; // relaxation of $(\mathscr{F}_j)$
        $\overline{z}^{[j]} := g_j(x^{[j]})$; // $\overline{z}^{[j]} := -\infty$ if $\mathscr{F}_j = \emptyset$
        $\overline{z} := \max\{\overline{z}^{[j]} : (\mathscr{F}_j) \in AN\}$; // update global upper bound
        if $x^{[j]} \in \mathscr{F}_j$ then
            $y^{[j]} := x^{[j]}$;
        else
            attempt to find $y^{[j]} \in \mathscr{F}_j$ via heuristic; // unassigned if unsuccessful
        end
        $\underline{z}^{[j]} := f(y^{[j]})$; // set $\underline{z}^{[j]} := -\infty$ if $y^{[j]}$ unassigned
        if $\underline{z}^{[j]} > \underline{z}$ then
            $x^* := y^{[j]}$; // update incumbent
            $\underline{z} := \underline{z}^{[j]}$; // update global lower bound
        end
        if $\underline{z}^{[j]} = \overline{z}^{[j]}$ then
            $AN := (AN \setminus \{\mathscr{F}_j\})$; // prune by optimality
        else
            $AN \leftarrow (AN \setminus \{(\mathscr{F}_j)\}) \cup \{(\mathscr{F}_{j_1}), \dots, (\mathscr{F}_{j_k})\}$; // branching
            $\overline{z}^{[j_\ell]} := \overline{z}^{[j]}$, $(\ell = 1, \dots, k)$; // child nodes inherit upper bounds from parent
        end
    end
```

LP Based B&B

Data Specification (LP based B&B)

Input:
objective $x \mapsto c^{\mathbf{T}}x$;
description of $\mathscr{F}$ via formulation $\mathscr{P}$ (polyhedron);

Methods (LP based B&B)

Required Methods:
compute dual bounds $\bar{z}^{[j]} = c^{\mathbf{T}}x^{[j]}$ by solving LP relaxation

$$(\mathscr{P}_j) \quad x^{[j]} = \arg\max\{c^{\mathbf{T}}x : x \in \mathscr{P}_j\};$$

branch on fractional variable $x_i^{[j]} \notin \mathbb{Z}$

$$\mathscr{P}_{j_1} = \mathscr{P}_j \cap \left\{x : x_i \leq \lfloor x_i^{[j]} \rfloor\right\};$$
$$\mathscr{P}_{j_2} = \mathscr{P}_j \cap \left\{x : x_i \geq \lceil x_i^{[j]} \rceil\right\};$$

Adaptation to Specific Problems

Branch & Bound is a framework that can be used to build customised algorithms for specific problem classes:

- choice of algorithm to compute dual bounds,
- choice of heuristics to compute primal bounds,
- branching rules,
- node selection rules,
- preprocessing,
- combination with cutting planes (see later lectures).

Choice of Algorithm to Compute Dual Bounds

The simplex algorithm is not always the algorithm of choice in LP-based B&B.

Example (Greedy solution of knapsack LP)

Consider the 0-1 knapsack problem

$$\max\left\{\sum_{j=1}^{n} c_j x_j \, : \, \sum_{j=1}^{n} a_j x_j \leq b,\ x \in \mathbb{B}^n\right\},$$

where $a_j, c_j > 0$ for $j = 1, \dots, n$. The LP relaxation

$$\max\left\{\sum_{j=1}^{n} c_j x_j \, : \, \sum_{j=1}^{n} a_j x_j \leq b,\ 0 \leq x_j \leq 1, \quad (j = 1, \dots, n)\right\}$$

has special structure that makes it possible to solve it greedily:

Re-index the variables to that $\frac{c_1}{a_1} \geq \cdots \geq \frac{c_n}{a_n} > 0$, $\sum_{j=1}^{r-1} a_j \leq b$ and $\sum_{j=1}^{r} a_j > b$. The optimal solution of the LP relaxation is then given by $x_j = 1$ for $j = 1, \dots, r-1$, $x_r = (b - \sum_{j=1}^{r-1} a_j)/a_r$ and $x_j = 0$ for $j > r$. (See problem sheet.)

For most IPs, LP-based B&B is significantly faster when the simplex algorithm is applied to the dual of the LP relaxations $(\mathscr{P}_j)$ rather than on the primal problem.

To understand why, let us see what happens when an LP

$$
\begin{aligned}
(\mathscr{P}) \quad \max_{x\in\mathbb{R}^n}\ & c^{\mathbf{T}}x \\
\text{s.t. } & \mathbf{a}_i^{\mathbf{T}}x \leq b_i, \quad (i = 1, \dots, m) \\
& x_j \geq 0, \quad (j = 1, \dots, n),
\end{aligned}
$$

where $\mathbf{a}_i^{\mathbf{T}}$ are row vectors, is amended by introducing a new constraint:

$$
\begin{aligned}
(\mathscr{P}') \quad \max_{x\in\mathbb{R}^n}\ & c^{\mathbf{T}}x \\
\text{s.t. } & \mathbf{a}_i^{\mathbf{T}}x \leq b_i, \quad (i = 1, \dots, m) \\
& \mathbf{a}_{m+1}^{\mathbf{T}}x \leq b_{m+1}, \quad (i = 1, \dots, m) \\
& x_j \geq 0, \quad (j = 1, \dots, n).
\end{aligned}
$$

An optimal basic feasible solution (x_B, x_N) of $(\mathscr{P})$ is not necessarily feasible for $(\mathscr{P})'$, which hinders us from re-optimising the solution via primal simplex pivots.

The change in the dual

$$(\mathscr{D}) \quad \min_{y\in\mathbb{R}^m} \; b^{\mathbf{T}} y$$
$$\text{s.t.} \; \sum_{i=1}^{m} y_i \mathbf{a}_i \geq c,$$
$$y_i \geq 0, \quad (i = 1, \dots, m)$$

is more benign, as the amendment of $(\mathscr{P})$ into $(\mathscr{P})'$ corresponds to the introduction of a new variable y_{m+1},

$$(\mathscr{D}') \quad \min_{y\in\mathbb{R}^{m+1}} \; b^{\mathbf{T}} y$$
$$\text{s.t.} \; \sum_{i=1}^{m} y_i \mathbf{a}_i + y_{m+1}\mathbf{a}_{m+1} \geq c,$$
$$y_i \geq 0, \quad (i = 1, \dots, m+1)$$

If (y_B, y_N) is an optimal basic feasible solution of $(\mathscr{D})$, then setting $y_{m+1} = 0$ and adding it to the set of non-basic variables yields a basic feasible solution for $(\mathscr{D}')$ that can be re-optimised via a few extra simplex pivots.

Example (Simplex warm start of dual problem)

Suppose our dual reads

$$(\mathscr{D}) \quad \begin{aligned} \min\; & -5y_1 - 4y_2 - 3y_3 \\ \text{s.t.}\; & -2y_1 - 3y_2 - y_3 \geq -5 \\ & -4y_1 - y_2 - 2y_3 \geq -11 \\ & -3y_1 - 4y_2 - 2y_3 \geq -8 \\ & y_1, y_2, y_3 \geq 0, \end{aligned}$$

which can be written in primal form,

$$(\mathscr{D}) \quad \begin{aligned} \max\; & 5y_1 + 4y_2 + 3y_3 \\ \text{s.t.}\; & 2y_1 + 3y_2 + y_3 \leq 5 \\ & 4y_1 + y_2 + 2y_3 \leq 11 \\ & 3y_1 + 4y_2 + 2y_3 \leq 8 \\ & y_1, y_2, y_3 \geq 0. \end{aligned}$$

We already found the optimal dictionary and optimal solution $y_1^* = 2$, $y_3^* = 1$, $y_2^* = 0$ in an earlier lecture,

$$\begin{aligned} y_3 &= 1 + y_2 + 3y_4 - 2y_6 \\ y_1 &= 2 - 2y_2 - 2y_4 + y_6 \\ y_5 &= 1 + 5y_2 + 2y_4 \\ z &= 13 - 3y_2 - y_4 - y_6. \end{aligned}$$

Example (continued)

Let us now add a new variable,

$$
(\mathscr{D}') \quad \begin{aligned} \max\ & 5y_1 + 4y_2 + 3y_3 + y_4 \\ \text{s.t. } & 2y_1 + 3y_2 + y_3 - y_4 \leq 5 \\ & 4y_1 + y_2 + 2y_3 + y_4 \leq 11 \\ & 3y_1 + 4y_2 + 2y_3 - 2y_4 \leq 8 \\ & y_1, y_2, y_3, y_4 \geq 0. \end{aligned}
$$

Introducing slack variables,

$$
\begin{aligned} y_5 &= 5 - 2y_1 - 3y_2 - y_3 + y_4, \\ y_6 &= 11 - 4y_1 - y_2 - 2y_3 - y_4, \\ y_7 &= 8 - 3y_1 - 4y_2 - 2y_3 + 2y_4. \end{aligned}
$$

To find the dictionary corresponding to $(y^*, 0) = (2, 0, 1, 0)$ (or $(2, 0, 1, 0, 0, 1, 0)$ with the slack variables) we need to reformulate these equations so as to express y_1, y_3, y_6 in terms of y_2, y_4, y_5, y_7. Gaussian elimination yields

$$
\begin{aligned} y_1 &= 2 - 2y_2 - 2y_5 + y_7, \\ y_6 &= 1 + 5y_2 - 3y_4 + 2y_5, \\ y_3 &= 1 + y_2 + y_4 + 3y_5 - 2y_7. \end{aligned}
$$

Example (continued)

Expressing $z = 5y_1 + 4y_2 + 3y_3 + y_4$ in terms of y_2, y_4, y_5, y_7 by substituting from the previous equations, we find the required dictionary to warm-start the simplex method:

$$\begin{aligned} y_1 &= 2 - 2y_2 - 2y_5 + y_7 \\ y_6 &= 1 + 5y_2 - 3y_4 + 2y_5 \\ y_3 &= 1 + y_2 + y_4 + 3y_5 - 2y_7 \\ z &= 13 - 3y_2 + 4y_4 - y_5 - y_7. \end{aligned}$$

Note that the new dictionary is feasible but sub-optimal, as we can pivot on y_4.

The described method for setting up the new dictionary/tableau is just the approach we discussed in the lecture on the simplex algorithm when considering how to compute the tableau that corresponds to a particular basis choice directly.

Heuristics to Compute Primal Bounds

Example (Knapsack primal bounds)

Consider the 0-1 knapsack problem

$$z^* = \max_x \sum_{j=1}^{n} c_j x_j$$

$$\text{subject to } \sum_{j=1}^{n} a_j x_j \leq b,$$

$$x_j \in \{0,1\}, \quad (j = 1, \dots, n),$$

where a_j $(j = 1, \dots, n)$ and b are positive integers, and c_j are rational numbers.

Let z_{LP} be the optimal objective value of the LP-relaxation and z_{gh} the primal bound obtained from the greedy heuristic described on the next slide. Let $r - 1 = \max\{i : \sum_{j=1}^{i} a_j \leq b\}$ and $\xi = b - \sum_{j=1}^{r-1} a_j$.

Then the following approximation bound holds,

$$z_{LP} \geq z^* \geq z_{gh} \geq \left(1 - \frac{\xi}{b}\right) \times z_{LP}.$$

Algorithm (Greedy knapsack heuristic)

```
Re-index s.t. $c_1/a_1 \geq c_2/a_2 \geq \cdots \geq c_n/a_n$. Set $z = 0$, $v = b$; // initialisation
for $j$=1,...,$n$ do
    if $a_j \leq v$ then
        $x_j = 1$;
        $v \leftarrow v - a_j$;
        $z = z + c_j$;
    else
        $x_j = 0$;
    end
end
```

Branching Rules

Branching on most fractional variable. Let C be the set of fractional variables of the solution x^* to a LP relaxation. The *most fractional variable* approach is to branch on the variable that corresponds to the index

$$j = \arg\max_{i \in C} \min\{f_i, 1 - f_i\},$$

where $f_i = x_i^* - \lfloor x_i^* \rfloor$.

Branching by priorities. In this approach the user can indicate a priority of importance for the decision variables to be integer. The system will then branch on the fractional variable with highest priority.

Example

Fixed charge network Consider a fixed charge network problem

$$\min\{c^{\mathbf{T}}x + fy : \; Nx = b, \; x \leq uy, \; x \in \mathbb{R}^n_+, \; y \in \mathbb{Z}^n_+\},$$

where N is the node-arc incidence matrix of the network, and b is the demand vector.

Since rounding up the variables y_j corresponding to large fixed costs f_j changes the objective function more severely than rounding y_j corresponding to small fixed costs, we prioritise the y_j in order of decreasing fixed costs f_j.

GUB/SOS branching. Many IP models contain *generalised upper bound* (GUB) or *special ordered sets* (SOS) constraints of the form

$$\sum_{j=1}^{k} x_j = 1,$$

with $x_j \in \mathbb{B}$ for all j. In this case it is not good to branch on a factional variable x_i^* from the solution x^* of a LP relaxation, since the branching does not lead to balanced sets:

$$S_1 = \{x \in S : \; x_i = 1\}$$

contains only one point $x_i = 1$, $x_j = 0 \; \forall \, j \neq i$, whereas

$$S_2 = \{x \in S : \; x_i = 0\}$$

contains $|S| - 1$ points.

In these cases, a better choice of branching is given by fixing an order $j_1, \dots, j_k$ of the variables and choosing

$$S_1 = S \cap \{x : \; x_{j_i} = 0, \; i = 1, \dots, r\},$$
$$S_2 = S \cap \{x : \; x_{j_i} = 0, \; i = r+1, \dots, k\},$$

where $r = \min\{s : \; \sum_{i=1}^{s} x_{j_i}^* \geq \frac{1}{2}\}$.

Strong Branching. This is used only on difficult problems where it is worthwhile spending more time to find a good branching. In this approach one

i) chooses a set C of candidate variables, branches up and down for each $x_j \in C$,

ii) computes upper bounds z_j^U and z_j^D by solving the LP relaxations of the up and down branching corresponding to x_j,

iii) chooses the variable having the largest effect

$$j^* = \arg\min\{\max(z_j^U, z_j^D) : j \in C\}$$

as the *actual branching variable* for the branch-and-bound scheme.

That is, one explores several possible branchings and chooses to pursue the branches below only the most promising branching variable.

Node Selection

Depth-First. A *depth-first strategy* aims at finding a feasible solution quickly, by only choosing an active node that is a direct descendant of the previously processed node. It is implemented by operating a *last-in-first-out* stack for the active nodes.

Best-Node-First. To minimise the total number of nodes processed during the run of the algorithm, the optimal strategy is to always choose the node with the largest upper bound, i.e., S_j such that

$$\bar{z}^{[j]} = \max\{\bar{z}^{[i]} : S_i \in \mathrm{AN}\}.$$

Under this strategy we will never branch on a node S_t whose upper bound $\bar{z}^{[t]}$ is smaller than the optimal value z of S. This is called a *best-node-first* strategy.

The depth-first and best-node-first strategies are usually mutually contradictory, so a compromise has to be reached. Usually, depth-first is used initially until a feasible solution is found and a lower bound $\underline{z}$ is established.

Preprocessing Linear Programming Problems

LP or IP models can often be simplified by reducing the number of variables and constraints, and IP models can be tightened before any actual branch-and-bound computations are performed.

Example (Preprocessing an LP)

Consider the LP instance

$$\begin{aligned} \max\ & 2x_1 + x_2 - x_3 \\ \text{s.t.}\ & 5x_1 - 2x_2 + 8x_3 \leq 15 \\ & 8x_1 + 3x_2 - x_3 \geq 9 \\ & x_1 + x_2 + x_3 \leq 6 \\ & 0 \leq x_1 \leq 3 \\ & 0 \leq x_2 \leq 1 \\ & 1 \leq x_3. \end{aligned}$$

Tightening bounds: Isolating x_1 in the first constraint and using $x_2 \leq 1$, $-x_3 \leq -1$ yields

$$5x_1 \leq 15 + 2x_2 - 8x_3 \leq 15 + 2 \times 1 - 8 \times 1 = 9,$$

and hence, $x_1 \leq 9/5$, which tightens the bound $x_1 \leq 3$.

Likewise, isolating x_3 in the first constraint, and using the bound constraints, we find

$$8x_3 \leq 15 + 2x_2 - 5x_1 \leq 15 + 2 \times 1 - 5 \times 0 = 17.$$

This implies $x_3 \leq 17/8$ and tightens $x_3 \leq \infty$.

And finally, isolating x_2 in the first constraint,

$$2x_2 \geq 5x_1 + 8x_3 - 15 \geq 5 \times 0 + 8 \times 1 - 15 = -7$$

yields $x_2 \geq -7/2$ which does not tighten $x_2 \geq 0$.

Proceeding similarly with the second and third constraints, we obtain the tightened bound

$$8x_1 \geq 9 - 3x_2 + x_3 \geq 9 - 3 + 1 = 7,$$

yielding the improved bound $x_1 \geq 7/8$.

As some of the bounds have changed after the first sweep, we may now go back to the first constraint and tighten the bounds yet further. Isolating x_3, we obtain

$$8x_3 \leq 15 + 2x_2 - 5x_1 \leq 15 + 2 - 5 \times \frac{7}{8} = \frac{101}{8},$$

yielding the improved bound $x_3 \leq 101/64$.

Continuing the second sweep by isolating each variable in turn in each of the constraints 1–3, and using the bound constraints, several bound constraints may further tighten in general, but not in the present example.

How many sweeps of this process are needed? One can show that after two sweeps of all the constraints and variables, the *bounds cannot improve any further!*

Redundant Constraints: Using the final upper bounds in constraint 3,

$$x_1 + x_2 + x_3 \leq \frac{9}{5} + 1 + \frac{101}{64} < 6,$$

so that this constraint is redundant and can be omitted.

The remaining problem is

$$\begin{aligned}\max\ & 2x_1 + x_2 - x_3\\ & 5x_1 - 2x_2 + 8x_3 \le 15\\ & 8x_1 + 3x_2 - x_3 \ge 9\\ & \frac{7}{8} \le x_1 \le \frac{9}{5}, \quad 0 \le x_2 \le 1, \quad 1 \le x_3 \le \frac{101}{64}.\end{aligned}$$

Variable fixing:

- Increasing x_2 makes the objective function grow and loosens all constraints except $x_2 \le 1$. Therefore, in an optimal solution we must have $x_2 = 1$.
- Decreasing x_3 makes the objective function grow and loosens all constraints except $1 \le x_3$. Thus, in an optimal solution we must have $x_3 = 1$.

This leaves the trivial problem

$$\max\left\{2x_1 : \frac{7}{8} \le x_1 \le \frac{9}{5}\right\}.$$

Preprocessing Integer Programming Problems

In the preprocessing of IPs we have further possibilities:

- For all x_j with an integrality constraint $x_j \in \mathbb{Z}$ any bounds $l_j \leq x_j \leq u_j$ can be tightened to $\lceil l_j \rceil \leq x_j \leq \lfloor u_j \rfloor$.
- For binary variables new *logical* or *Boolean* constraints can be derived that tighten the formulation and hence lead to fewer branching nodes in a branch-and-bound procedure.

The latter point is illustrated in the next example:

Example (Preprocessing a Binary Programming Problems)

Consider a BIP instance whose feasible set is defined by the following constraints,

$$\begin{aligned} 7x_1 + 3x_2 - 4x_3 - 2x_4 &\leq 1 \\ -2x_1 + 7x_2 + 3x_3 + x_4 &\leq 6 \\ -2x_2 - 3x_3 - 6x_4 &\leq -5 \\ 3x_1 - 2x_3 &\geq -1 \\ x &\in \{0,1\}^4. \end{aligned}$$

Generating logical inequalities: The first constraint shows that $x_1 = 1 \Rightarrow x_3 = 1$, which can be written as $x_1 \leq x_3$. Likewise, $x_1 = 1 \Rightarrow x_4 = 1$, or equivalently, $x_1 \leq x_4$.

Finally, constraint 1 also shows that the problem is infeasible if $x_1 = x_2 = 1$. Therefore, the following constraint must hold,

$$x_1 + x_2 \leq 1.$$

We can process the remaining constraints in a similar vein:

- Constraint 2 yields the inequalities $x_2 \leq x_1$ and $x_2 + x_3 \leq 1$.
- Constraint 3 yields $x_2 + x_4 \geq 1$ and $x_3 + x_4 \geq 1$.
- Constraint 4 yields $x_1 \geq x_3$.

Although the introduction of the new logical constraints makes the problem seem more complicated, the formulation becomes tighter and thus easier to solve. Furthermore, we can now process the problem further:

Combining pairs of logical inequalities: We now consider pairs involving the same variables.

- $x_1 \leq x_3$ and $x_1 \geq x_3$ yield $x_1 = x_3$.
- $x_1 + x_2 \leq 1$ and $x_2 \leq x_1$ yield $x_2 = 0$, and then $x_2 + x_4 \geq 1$ yields $x_4 = 1$.

Simplifying: Substituting the identities $x_2 = 0$, $x_3 = x_1$ and $x_4 = 1$ we found, all four constraints become redundant.
We are left with the choice $x_1 \in \{0, 1\}$, and hence the feasible set contains only two points

$$S = \{(1, 0, 1, 1), (0, 0, 0, 1)\}.$$