B6.3 Integer Programming

Raphael Hauser

Mathematical Institute
University Of Oxford

Michaelmas Term 2018

Practical Subgradient Algorithm

Theorem (Step length choice in subgradient algorithm)

i) *If* $\sum_k \mu_k \to \infty$ *and* $\sum_k \mu_k^2 \to 0$ *as* $k \to \infty$, *then* $z(u^{[k]}) \to w_{LD}$.

ii) *If* $\sum_k \mu_k \to \infty$ *and* $\mu_k \to 0$ *as* $k \to \infty$, *then* $z(u^{[k]}) \to w_{LD}$.

iii) *If* $\mu_k = \mu_0 \rho^k$ *for some fixed* $\rho \in (0,1)$ *for* μ_0 *sufficiently large and* ρ *sufficiently close to* 1, *then* $z(u^{[k]}) \to w_{LD}$.

iv) *if* $\overline{w} \geq w_{LD}$ *and*

$$\mu_k = \frac{\varepsilon_k \times (z(u^{[k]}) - \overline{w})}{\|v^{[k]}\|^2},$$

where $\varepsilon_k \in (0,2)$ *for all* k, *then either* $z(u^{[k]}) \to w_{LD}$ *for* $k \to \infty$, *or else* $\overline{w} \geq z(u^{[k]}) \geq w_{LD}$ *occurs for some finite* k.

Step length choice iv) gives the most useful step lengths in practice, but note that for μ_k to be positive, we need an upper bound $\overline{w} \in (w_{LD}, z(u^{[k]}))$. In practical applications such a bound is not available explicitly.

Note however:

- Lower bounds $\underline{w}$ of w_{LD} are available by ways of using heuristics that produce primal feasible solutions.
- If the bound $\overline{w}$ in Rule iv) is chosen too low, μ_k is positive but possibly too large. If $z(u^{[k+1]}) < z(u^{[k]})$, this does not pose a problem, as descent is achieved and the point $u^{[k+1]}$ can be accepted as the next iterate.
- If $z(u^{[k+1]}) \geq z(u^{[k]})$, the step μ_k took the iterate to a point where the objective function $z(\underline{u})$ increases again. The guess of $\overline{w}$ then needs to be increased to reduce the step length μ_k.

Algorithm (Practical Subgradient Algorithm for Solving (LD))

```
Initialise:
    fix $\varepsilon \in (0, 2)$, choose $u \in \mathbb{R}^m$ with $u_i \geq 0$, $(i \in \mathcal{I})$;
    $X^* := \arg\max\{c^T x + u^T(d - Dx) : x \in \mathscr{X}\}$, $V := \{d - Dx^* : x^* \in X^*\}$;
    find $x \in \mathscr{F} = \mathscr{X} \cap \{D_1 x \leq d_1,\ D_2 x = d_2\}$;
    set $\overline{w} := \underline{w} := c^T x$;
while $\vec{0} \notin \mathrm{conv}(V)$ do
    choose $v \in V$;
    $z^+ := +\infty$;
    while $z^+ \geq z(u)$ do
        $\overline{w} := \frac{z(u)+\overline{w}}{2}$; // this is our guess of a suitable dual bound
        $\mu := \frac{\varepsilon(z(u)-\overline{w})}{\|v\|^2}$;
        if $i \in \mathcal{I}$ then
            $u_i^+ := \max(u_i - \mu v_i,\ 0)$; // compute candidate updates
        else
            $u_i^+ := u_i - \mu v_i$; // compute candidate updates
        end
        $z^+ := z(u^+)$; // evaluate candidate updates
    end
    $u := u^+$; // accept candidate updates as actual updates
    $\overline{w} := \underline{w}$;
    $X^* := \arg\max\{c^T x + u^T(d - Dx) : x \in \mathscr{X}\}$, $V := \{d - Dx^* : x^* \in X^*\}$;
end
```

Example (STSP)

We revisit the Lagrangian Dual of the STSP from last lecture, this time without assuming that we have a priori knowledge of the optimal u.

The dualised constraints were the degree constraints

$$\sum_{e\in\delta(i)} x_e = 2 \quad (i \in V).$$

Since these are *equality constraints*, the Lagrange multipliers u are unconstrained, and the updating rule is

$$u_i^{[k+1]} = u_i^{[k]} + \mu_k \big(2 - \sum_{e\in\delta(i)} x_e^*(u^{[k]})\big).$$

The STSP being a *minimisation* problem rather than a maximisation problem, we have to replace all mins by maxes, lower bounds by upper bounds and so forth.

We step length rule iii) with $\varepsilon_k = 1$, that is,

$$\mu_k = \frac{\underline{w} - z(u^{[k]})}{\sum_{i\in V}\big(2 - \sum_{e\in\delta(i)} x_e^*(u^{[k]})\big)^2},$$

where $\underline{w}$ is a *lower bound* on w_{LD}, and where we had to invert the sign of μ_k because (LD) is a maximisation problem.

Example (STSP continued)

Initialisation: We apply the greedy heuristic and find the tour $1 \to 2 \to 3 \to 4 \to 5 \to 1$ of length 148.

As no lower bound $\underline{w}$ on w_{LD} is known at present, we use the primal (upper) bound $\overline{w} = 148$ instead and see how far this allows us to decrease $z(u^{[k]})$, knowing that at a later time we will probably have to replace $\overline{w}$ by a smaller value, as 148 is usually not a lower bound.

Iteration 0: Starting with $u^{[0]} = [0, 0, 0, 0, 0]$, the revised costs are given by

$$\bar{c}_{ij}^{[0]} = c_{ij} - u_i^{[0]} - u_j^{[0]} = c_{ij}.$$

Solving the associated min-cost 1-tree problem, we find the optimal 1-tree with edge incidence matrix

$$[x_{ij}^*(u^{[0]})] = \begin{bmatrix} - & 1 & 1 & 0 & 0 \\ - & - & 1 & 0 & 0 \\ - & - & - & 1 & 1 \\ - & - & - & - & 0 \\ - & - & - & - & - \end{bmatrix},$$

leading to the objective value $z(u^{[0]}) = 130$.

Example (STSP continued)

The subgradient of $z(u)$ at $u^{[0]}$ is

$$\Big[\big(2 - \sum_{e \in \delta(i)} x_e^*(u^{[0]})\big)_{i=1,\dots,m}\Big] = [0, 0, -2, 1, 1],$$

and as $\mu_0 = (148 - 130)/6 = 3$, we find $u^{[1]} = u^{[0]} + 3 \cdot [0, 0, -2, 1, 1] = [0, 0, -6, 3, 3]$.

Iteration 1: The new cost matrix is

$$[\bar{c}_{ij}^{[1]}] = \begin{bmatrix} - & 30 & 32 & 47 & 37 \\ - & - & 30 & 37 & 47 \\ - & - & - & 27 & 29 \\ - & - & - & - & 24 \\ - & - & - & - & - \end{bmatrix}.$$

The optimal 1-tree is found as

$$[x_{ij}^*(u^{[1]})] = \begin{bmatrix} - & 1 & 1 & 0 & 0 \\ - & - & 1 & 0 & 0 \\ - & - & - & 1 & 0 \\ - & - & - & - & 1 \\ - & - & - & - & - \end{bmatrix},$$

leading to the objective value $z(u^{[1]}) = 143 + 2\sum_i u_i^{[1]} = 143$.

Example (STSP continued)

Updating the Lagrange multipliers, we obtain

$$u^{[2]} = u^{[1]} + \frac{148 - 143}{2} \cdot [0, 0, -1, 0, 1] = \left[0, 0, -\frac{17}{2}, 3, \frac{11}{2}\right].$$

Iteration 2: The new cost matrix and optimal 1-tree are

$$[\bar{c}_{ij}^{[2]}] = \begin{bmatrix} - & 30 & 34.5 & 47 & 34.5 \\ - & - & 32.5 & 37 & 44.5 \\ - & - & - & 29.5 & 29 \\ - & - & - & - & 21.5 \\ - & - & - & - & - \end{bmatrix}, \qquad [x_{ij}^*(u^{[2]})] = \begin{bmatrix} - & 1 & 0 & 0 & 1 \\ - & - & 1 & 0 & 0 \\ - & - & - & 0 & 1 \\ - & - & - & - & 1 \\ - & - & - & - & - \end{bmatrix},$$

leading to the objective value $z(u^{[2]}) = 147.5$.

This means that 147.5 is a lower bound on the optimal value z of the STSP. But since $[c_{ij}]$ is integer valued, this implies

$$148 = \lceil 147.5 \rceil \leq z \leq \overline{w} = 148,$$

which shows that the greedy tour $1 \to 2 \to 3 \to 4 \to 5 \to 1$ was STSP-optimal!

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

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

Variable fixing:

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

This leaves the trivial problem

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

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

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

The Cutting Plane Algorithm

The above discussion of preprocessing steps shows that it is possible to derive valid inequalities for the integer feasible solutions of an IP from its polyhedral formulation. This idea can be generalised and systematically exploited in the design of algorithms.

Consider the problem

$$\begin{aligned}(\mathrm{IP})\quad & \max\ c^{\mathbf{T}} x \\ & \text{s.t. } x \in \mathscr{X} = \mathscr{P} \cap \mathbb{Z}^{\boldsymbol{n}},\end{aligned}$$

where $\mathscr{P} = \{x \in \mathbb{R}^{\boldsymbol{n}} : a_{\boldsymbol{i}}^{\mathbf{T}} x = b_{\boldsymbol{i}},\ (i = 1, \dots, m),\ x \geq 0\}$.

Definition (Valid inequalities and cuts)

A *valid inequality* for $\mathscr{X}$ is an inequality of the form

$$\alpha^{\mathbf{T}} x \leq \alpha_{\mathbf{0}}$$

that is satisfied for all $x \in \mathscr{X}$ (but not necessarily for all $x \in \mathscr{P}$).

A *cut* for $x^* \in \mathscr{P}$ is a valid inequality for $\mathscr{X}$ such that

$$\alpha^{\mathbf{T}} x^* > \alpha_{\mathbf{0}},$$

that is, $\mathscr{P} \cap \{x : \alpha^{\mathbf{T}} x \leq \alpha_{\mathbf{0}}\}$ is a tighter formulation of $\mathscr{X}$ that excludes x^*.

Algorithm (Cutting Plane Algorithm)

```
solve LP relaxation $x^* = \arg\max_{\mathbf{x}}\{\mathbf{c}^{\mathbf{T}}x : x \in \mathscr{P}\}$; // initialisation
while $x^*$ fractional do
    find a cut $\alpha^{\mathbf{T}}x \leq \alpha_{\mathbf{0}}$ for $x^*$;
    $\mathscr{P} \leftarrow \mathscr{P} \cap \{x : \alpha^{\mathbf{T}}x \leq \alpha_{\mathbf{0}}\}$;
    solve LP relaxation $x^* = \arg\max_{\mathbf{x}}\{\mathbf{c}^{\mathbf{T}}x : x \in \mathscr{P}\}$;
end
```

Notes:

- The algorithm relies on systematic methods to generate cuts, an issue we will discuss further.
- In contrast to the Branch & Bound Algorithm, the convergence of the Cutting Plance Algorithm is not guaranteed but depends on the nature of the cuts that are applied.
- Ideally, one would like to apply cuts that are easily computed and cut off a large part of $\mathscr{P}$, but the two goals are often contradictory.
- The Cutting Plane Algorithm can be combined with the Branch-and-Bound Method, which yields the most powerful black-box solvers for IPs (Branch-and-Cut Algorithm).

Chvàtal Cuts

Example (Chvàtal cut)

Consider the IP $\min_{\mathbf{x}}\{-x_1 - x_2 - x_3 : x \in \mathscr{X}\}$ with $\mathscr{X} = \mathscr{P} \cap \mathbb{Z}^3$ and

$$\mathscr{P} = \{x \in \mathbb{R}^3 : x_1 + x_2 \leq 1,\ x_2 + x_3 \leq 1,\ x_1 + x_3 \leq 1,\ x \geq 0\}.$$

Using slack variables, the LP relaxation of (IP) reads

$$(\text{LP}) \quad \min_{\mathbf{x} \geq \mathbf{0}} -x_1 - x_2 - x_3 \quad \text{s.t.} \begin{cases} x_1 + x_2 + x_4 = 1 \\ x_2 + x_3 + x_5 = 1 \\ x_1 + x_3 + x_6 = 1 \end{cases}$$

Multiplying the three equality constraints with 0.5 and adding them yields

$$x_1 + x_2 + x_3 + 0.5x_4 + 0.5x_5 + 0.5x_6 = 1.5.$$

Using the non-negativity of x_4, x_5, x_6, this implies $x_1 + x_2 + x_3 \leq 1.5$, which is a valid inequality not only for $\mathscr{X}$ but also for $\mathscr{P}$. Now using the integrality of x_1, x_2, x_3, we obtain the valid inequality

$$x_1 + x_2 + x_3 \leq 1,$$

which is a cut for x^* because $x_1^* + x_2^* + x_3^* = 1.5$.

We now generalise the approach described above: For any $r \in \mathbb{R}^n$, let $\lfloor r \rfloor = [\lfloor r_1 \rfloor, \dots, \lfloor r_n \rfloor]$.

Definition (Chvàtal cuts)

Let $\mathscr{P}^{(0)} = \{x \geq 0 : Ax = b\}$ be a polyhedron given by a system of m equations and n non-negativity constraints, and let $u \in \mathbb{R}^m$. The Chvàtal cut associated with u is given by

$$\alpha^T x \leq \alpha_0,$$

where $\alpha^T := \lfloor u^T A \rfloor$ and $\alpha_0 := \lfloor u^T b \rfloor$.

Lemma (Chvàtal cuts are valid inequalities)

All Chvàtal cuts are valid inequalities for the set $\mathscr{X} = \mathscr{P}^{(0)} \cap \mathbb{Z}^n$.

Proof. $x \in \mathscr{X}$ implies $Ax = b$, and hence, $u^T Ax = u^T b$. Using $x \geq 0$ this implies $\lfloor u^T A \rfloor x \leq u^T b$, and using integrality of x_i, $(i = 1, \dots, n)$, this implies $\lfloor u^T A \rfloor x \leq \lfloor u^T b \rfloor$.

We state the next result without proof.

Theorem (Separation of non-integral vertices)

Given a vertex x^* *of* $\mathscr{P}^{(0)}$, *there exists a vector* $u \in \mathbb{R}^m$ *such that* $\lfloor u^T A \rfloor x \leq \lfloor u^T b \rfloor$ *is a cut for* x^*.