1 Integer Linear Programming is NP-Complete

We show that the feasibility problem for 0-1 integer linear programs is NP-complete. The proof is by an easy reduction from the satisfiability problem for formulae in conjunctive normal form.

Let $\phi = C_1 \wedge C_2 \wedge \cdots \wedge C_m$ be a formula in conjunctive normal form. Each C_i is the disjunction of three literals $x_{i1} \vee x_{i2} \vee x_{i3}$ and each literal is either a variable v or a negated variable $\neg v$. Assume that our variables are $v_1, \ldots, v_n$. We turn ϕ into an arithmetic formula Φ as follows:

- We introduce an integer variable z_i for every boolean variable v_i. The integer variable is constrained to take values zero and one: zero represents false and one represents true.
- We then replace $\wedge$ by $\times$, $\vee$ by $+$, unnegated variables v_j by z_j and negated variables $\neg v_j$ by $1 - z_j$.
- Let Φ be the arithmetic formula obtained in this way.

The ILP is

$$\Phi \geq 1 \quad \text{and} \quad 0 \leq z_i \leq 1 \text{ and } z_i \in \mathbb{Z} \text{ for all } i\ .$$

Lemma 1 *ϕ is satisfiable iff the integer program above is feasible.*

Proof: Assume that ϕ is satisfiable and let v^* be a satisfying assignment. Construct z^* by setting $z_i^* = 1$ iff v_i^* is true. Then $\Phi \geq 1$.

Conversely, if z^* is a feasible solution for the ILP, each clause must have an integer value of at least one. Hence the corresponding truth assignment satisfies ϕ. ∎

The following theorem is an immediate consequence. We use Zero-One-ILP to refer to integer programs with zero-one-variables.

Theorem 1 *Zero-One-ILP is NP-complete*

Proof: The Lemma above implies that it is NP-hard. Also given a Zero-One-ILP we can guess and check a solution in polynomial time ∎

How about general integer linear programming? Being more general than Zero-One-ILP it is certainly NP-hard. In order to show NP-completeness, one also needs to show that it is in NP. The key is to show that if an ILP has a solution then it has a solution whose entries are bounded by 4^{nL}, where L is defined as in the ellipsoid method. Try to prove it.

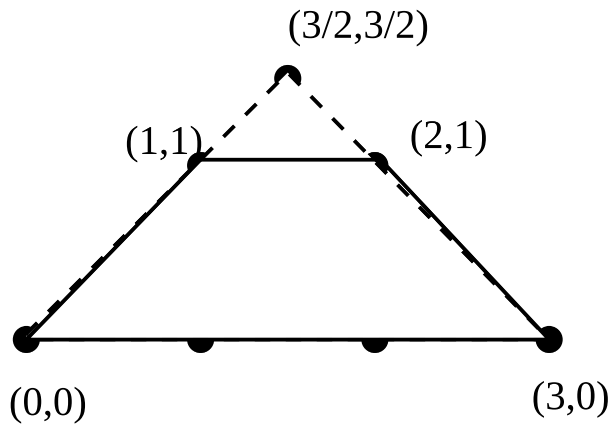

Figure 1: Consider the inequalities $y \leq x$, $y \leq 3-x$, $x \geq 0$, $y \geq 0$. The feasible region P is a triangle with vertices $(0,0)$, $(3,0)$, and $(3/2, 3/2)$. The boundary of P is shown dashed. P contains the integral points $\{(0,0),(1,0),(2,0),(3,0),(1,1),(2,1)\}$. They comprise the feasible set F_I of the integer program. The boundary of the convex hull P_I of F_I is shown in bold. It is defined by the inequalities $y \leq x$, $y \leq 3-x$, $x \geq 0$, $y \geq 0$, and $y \leq 1$.

2 General Cutting Plane Methods: Gomory Cuts

We show that ILPs can be solved by solving a series of linear programs.

Consider an integer linear program

$$\max\left\{c^T x \ ; \ Ax \leq b, x \geq 0, x \in I\!N^n\right\} \tag{ILP}$$

The *linear programming relaxation of ILP* is obtained by dropping the integrality constraint. We obtain the LP

$$\max\left\{c^T x \ ; \ Ax \leq b, x \geq 0, x \in I\!R^n\right\} \tag{LP}$$

Clearly, the maximal objective value of the LP is at least the objective value of the ILP; the linear programming relaxation provides an upper bound.

Let $F_I = \{x \ ; \ Ax \leq b, x \geq 0, x \in I\!N^n\}$ be the feasible set of ILP; observe that F_I is a discrete set. The feasible region of $P = \{x \ ; \ Ax \leq b, x \geq 0, x \in I\!R^n\}$ of LP is a superset of F_I; it is a connected subset of $I\!R^n$. In principle, the optimization problem over F_I can be formulated as a linear program. The convex hull P_I of F_I is a convex set with vertices in F_I. We could describe P_I by a system of inequalities and then use this system to optimize, see Figure 1. The problem with this approach is that

- the description could be big (= it might require a very large number of constraints to define P_I,
- the description could be hard to find,

- it would be overkill to find a complete description of P_I. We only need a linear description in the vicinity of the optimal solution.

The idea of cutting plane methods is to add linear inequalities to the linear program as needed, namely to cut off fractional solutions (hence the name). The idea is simple. We call an inequality $a^T x \leq \delta$ valid for ILP if $a^T x \leq \delta$ for all $x \in F_I$. We have the following simple, but powerful lemma.

Lemma 2 *If a is integral and*

$$a^T x \leq \delta$$

is valid for ILP then

$$a^T x \leq \lfloor \delta \rfloor$$

is valid for ILP.

Proof: If $a^T x \leq \delta$ and $x \in F_I$ then $a^T x \in \mathbb{Z}$ and hence $a^T x \leq \lfloor \delta \rfloor$. ∎

We next discuss Gomory's method for finding valid inequalities of the form $a^T x \leq \delta$ with a integral and δ fractional.

Let B be an optimal basis of the LP (we assume for simplicity that the LP is bounded) and x^* be the corresponding optimal solution to the LP. If x^* is integral, it is also an optimal solution of ILP and we are done.

So assume that x^* is not integral. Then one of the basic variables, say x_i, has a fractional value. Consider the corresponding row of the dictionary

$$x_i = \bar{b}_i + \sum_{j \in N} \bar{a}_{ij} x_j \ . \tag{1}$$

$\bar{b}_i$ is a rational number which is not integral. For a rational number β let $\{\beta\} = \beta - \lfloor \beta \rfloor$ be the fractional part of β. Then $\{\bar{b}_i\} \neq 0$. We split equation 1 into is integral part and its fractional part and obtain

$$x_i - \lfloor \bar{b}_i \rfloor - \sum_{j \in N} \bar{a}_{ij} x_j = \{\bar{b}_i\} + \sum_{j \in N} \{\bar{a}_{ij}\} x_j \ .$$

For every feasible solution of our ILP the left hand side is an integer (since the variables are constrained to be integral) and hence the right hand side is integral for every feasible solution of our ILP. Thus

$$E = \{\bar{b}_i\} + \sum_{j \in N} \{\bar{a}_{ij}\} x_j \in \mathbb{Z}$$

for every feasible solution of the ILP.

Let us take a closer look at this expression. We have $\{\bar{b}_i\} > 0$ since $\bar{b}_i$ is fractional, we have $\{\bar{a}_{ij}\} \geq 0$ by the definition of fractional value and we have $x_j \geq 0$ since our variables are constrained to be non-negative. Thus $E > 0$ for every feasible solution of our ILP. Since E is known to be integral, we even have $E \geq 1$. Thus

$$\{\bar{b}_i\} + \sum_{j \in N} \{\bar{a}_{ij}\} x_j \geq 1$$

for every feasible solution of our ILP. We may therefore add this inequality to the ILP and also to its linear programming relaxation.

The addition of the new constraint has no effect on the ILP; we just argued that $E \geq 1$ for every feasible solution to the ILP. It does however have an effect on the linear programming relaxation. Observe that in our current solution, we have $x_j^* = 0$ for all $j \in N$ and hence the current LP-solution is cut off by the new constraint. For this reason, the new constraint is called a cutting plane.

Let us look at an example. We have a ILP in two variables, say x and y, we are trying to maximize y and we have the constraints. $y \leq x$ and $y \leq 3 - x$. We add slack variables s and t and obtain the linear program $y + s - x = 0$ and $y + t + x = 3$. An optimal solution to the linear programming relaxation is $(x, y, s, t) = (3/2, 3/2, 0, 0)$. We use the fact that x has a fractional value to generate a cut. Solving for x in terms of the non-basic variables s and t yields

$$2x - s + t = 3 \quad \text{or} \quad x = \frac{3}{2} + \frac{1}{2}s - \frac{1}{2}t \ .$$

Splitting into integral and fractional part gives us

$$x - 1 - t = \frac{1}{2} + \frac{1}{2}s + \frac{1}{2}t \ .$$

The right hand side is integral and positive for every feasible solution of the ILP and hence we may generate the cut

$$\frac{1}{2} + \frac{1}{2}s + \frac{1}{2}t \geq 1 \quad \text{or} \quad s + t \geq 1$$

We may also interpret this constraint in terms of x and y. Adding our two original constraints gives $2y + s + t = 3$. Together with $s + t \geq 1$ this implies $y \leq 1$.

The technique for introducing new constraints just described was invented by Gomory in the late 50s and early 60s. It can be shown that if Gomory cuts are added in a careful way, the LP will have an integral optimal solution after a finite number of iterations, see [Sch86, Section 23.8]. The number may be large, however. It must be large in the worst case, since integer linear programming is NP-complete.

I called this section general cuts, because it works for *every* integer linear program. For specific integer linear programs, one can also use problem-specific cuts. We will see an example below in the section on the Traveling Salesman Problem. Problem-Specific Cuts are usually much more effective than general cuts.

How does one resolve an LP after adding a cutting plane. We know it already (see Section **??**), but it is worthwhile to review it. We introduced the constraint $E \geq 1$ which is not satisfied by our current basis solution. We add a non-negative slack variable s, the equation $s = E - 1$, and add s to the basis. In this way we obtain a dual feasible basis (since we have not changed the cost function) which we can take as the initial basis for phase II of the simplex method.

3 The Traveling Salesman Problem

Go through the chapter on the Traveling Salesman Problem in the book by Cook et al [CCPS98].

4 Column Generation and the Cutting Stock Problem

In the linear programming approach to the traveling salesman problem we used the cutting plane approach. The cutting plane approach is appropriate when the number of variables is reasonably small and the number of constraints is big. Column generation applies to the dual situation where the number of constraints is small and the number of variables is big.

4.1 A General View at Column Generation (also called Pricing)

Column generation is a method of dealing with linear programs with an exponential number of variables. It is the dual of cut generation which deals with linear programs with an exponential number of constraints.

Consider a linear program and assume that x is a non-basis variable. Let a be the column of the constraint matrix corresponding to x and let c be the coefficient of x in the cost function.

We use A for the part of the constraint matrix involving the basis and x_B for the basis variables. Then

$$x_B = A^{-1}b - A^{-1}ax$$

and

$$z = c_B^T x_B + cx = c_B^T A^{-1} b + (c - c_B^T A^{-1} a)x.$$

It pays off to add x to the basis if the reduced cost $c - c_B^T A^{-1} a$ of the variable x is negative.

Pricing treats the column vector a as an unknown and asks itself whether there is a variable x with corresponding column a such that $c - c_B^T A^{-1} a < 0$.

4.2 The Cutting Stock Problem

The following problem arises in a sawing mill. We have an unlimited number of raws of length L and need to cut d_i rods of length l_i for $1 \le i \le k$. The goal is to minimize the number of raws used. Let $d = (d_1, \ldots, d_k)$ be the demand vector.

The following ILP-formulation is natural. Let $D = \sum_i d_i$. Then we certainly do not need more than D raws. Introduce integral variables x_{ij}, $1 \le i \le k$, $1 \le j \le D$, and integral variables z_j, $1 \le j \le D$. All variables are non-negative and the z_j are binary (only values zero and one): x_{ij} is the number of rods of length l_i cut from the j-th raw and z_j is one if the j-th raw is used. Let $M = \max_i L/d_i$ be the maximal number of rods that can be cut from a raw. The our goal is to minimize $\sum_j z_j$ subject to the constraints $\sum_i l_i x_{ij} \le L$ for all j and $x_{ij} \le M z_j$ for all i and j. Observe that the second constraint forces z_j to be one if any x_{ij} is positive and the first constraint guarantees that no raw is over-used. The problem with this ILP-formulation is that its linear programming relaxation gives very weak bounds.

We next give an alternative formulation. The alternative formulation uses a large number of variables. A cutting pattern is any vector $p = (a_1, \ldots, a_k)$ of non-negative integers such that $\sum_i a_i l_i \le L$. There is in general an exponential number of cutting patterns. Trivial cutting patterns cut only rods of one kind, i.e., there is an i such that $a_i = \lfloor L/l_i \rfloor$ and $a_j = 0$ for $j \ne i$.

Let P be the set of all cutting patterns. For a pattern p let x_p the number of times the pattern p is used. The goal is to

$$\text{minimize } \sum_p x_p \text{ subject to } d = \sum_p p x_p,\ x_p \ge 0,\ x_p \text{ integral.}$$

As usual we obtain the LP relaxation by dropping the integrality constraint. The resulting LP has an exponential number of variables (but only k constraints). The advantage of the alternative LP over the natural LP is that it provides much better lower bounds for the underlying ILP. We need to discuss two issues

- How to solve the linear program?
- How to go from an optimal solution to the LP to an optimal solution of the ILP.

You can find a complete implementation on the web-page of the course.

4.3 Solving the LP

Let $P' \subset P$ be a subset of the cutting patterns. Initially, we might take P' as the set of all trivial cutting patterns. We solve the LP restricted to the variables in P'. The solution has a certain basis B. Let A be the matrix of cutting patterns in the basis. The cost vector is the all-one vector. Pricing leads to the following question:

Is there a cutting pattern, i.e., a vector $p = (a_1, \ldots, a_k)$ of non-negative integers with $\sum_i a_i l_i \le L$, such that $1 - u^T p < 0$, where $u^T = \mathbf{1}^T A^{-1}$ and $\mathbf{1}$ is the all-one vector? This is a knapsack problem. We have a knapsack of capacity L and k different objects. The i-th object has a weight a_i and a value u_i. The question is whether we can fill the knapsack (using a_i objects of kind i) such that the capacity of the knapsack is not exceeded, i.e., $\sum_i a_i l_i \leq L$ and such that the value of the knapsack exceeds one, i.e., $u^T p = \sum_i u_i a_i > 1$.

The knapsack problem is NP-complete, but not strongly NP-complete. Dynamic programming usually works well for the knapsack problem.

4.4 Dynamic Programming for the Knapsack Problem

We assume that the l_i are integer. We fill a table T with index set $[0 \mathinner{..} L]$ such that $T[l]$ is the maximal value of a knapsack whose weight is at most l. We do first using only objects of type one, then objects of type one and two, $\ldots$.

Suppose we have only one kind of object. There is not much choice we have: For $l < l_1$ the value is zero, i.e, $T[l] = 0$ for $l < l_1$ and for $l_1 \leq l \leq L$ we can add one item of value u_1 to a knapsack of weight at most $l - l_1$. Thus $T[l] = u_1 + T[l - l_1]$ for $l \geq l_1$.

Assume now that we considered the first $k-1$ items and now consider the k-th item. We can use it or not. Thus we should update the T-values according to the rule: For $l < l_k$ leave the T-value unchanged and for $l_k \leq l \leq L$ set $T[l] = \max(T[l], u_k + T[l - l_k])$.

It is important to consider the l's in increasing order (`for (int l = l[k], l <= L; l++)`) in the update step to account for the fact that we may take the k-th item any number of times. The initialization step is actually a special case of the update step. We could initialize T with the zero-vector and then start with $k = 1$. The running time of dynamic programming for the knapsack problem is $O(kL)$. This is pseudo-polynomial.

4.5 Solving the ILP

The solution of the LP gives us a lower bound for the optimal solution to the ILP. It also allows us to compute integer solutions.

The simplest way to obtain integer solutions is to round up the values of all variables. In this way all demands are certainly satisfied.

Alternatively, we could round down. Consider any basic variable x_p. We round it down and use the cutting pattern p, $\lfloor x_p \rfloor$ times. This will satisfy part of the demand (usually a large fraction of it). We satisfy the remaining demand using a heuristic.

If we are lucky the LP lower bound tells us that the solution obtained in this way is optimal.

If not, we use branching. We take one of the fractional variables, say x, in the LP (one whose value, say β is close to half-integral) and generate two subproblems. One with the additional constraint $x \leq \lfloor \beta \rfloor$ and one with $x \geq \lceil \beta \rceil$.

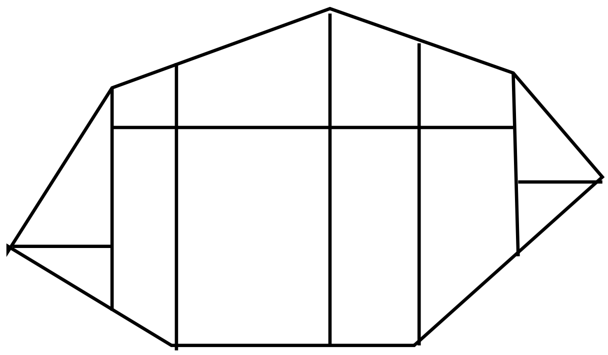

Figure 2: The feasible region P of the linear programming relaxation and its partition into trapezoids.

5 Integer Linear Programming in Finite Dimensions

Lenstra showed in 83 that integer linear programming in a fixed number of variables is in P. For any fixed number d of variables, the running time is polynomial in the number m of constraints and the size L of the problem (same L as in the Ellipsoid method). I cannot describe the method here (see Section 18.4 in [Sch86]), but I will outline a procedure for one and two dimensions. The solution for two dimensions is due to Ravi Kannan [Kan80].

If there is only a single variable x, the problem is trivial. The constraints are upper bounds $x \le u_i$ and lower bounds $x \ge l_j$. The problem is feasible if $\lceil \max_j l_j \rceil \le \lfloor \min_i u_i \rfloor$.

In two dimensions we have to work harder. In fact, we have to work surprisingly hard. Our goal is to solve integer linear programs in two variables.

$$\max \{c_1x_1 + c_2x_2 \; ; \; a_{i1}x_1 + a_{i2}x_2 \le b_i \text{ for } 1 \le i \le m, \; x_1 \text{ and } x_2 \text{ integral}\}$$

The coefficients c_1, c_2, a_{i1}, a_{i2}, b_i are assumed to be integral. Let $L = m + \log(1 + |c_1|) + \log(1 + |c_2|) + \sum_i (\log(1 + |a_{i1}|) + \log(1 + |a_{i2}|) + \log(1 + |b_i|))$ be the total bit length of the ILP.

Theorem 2 *ILPs in two variables can be solved in time polynomial in L.*

We first compute the feasible region P of the LP relaxation, i.e., the intersection of the half-spaces $a_{i1}x_1 + a_{i2}x_2 \le b_i$. Using methods from computational geometry this can be done with $O(m \log m)$ steps where each step is a multiplication/addition of L-bit numbers. Thus the time is $O(l^2 m \log m)$ assuming that we use the school method for integer multiplication.

We split P into trapezoids by drawing a vertical line through each vertex. We split each trapezoid into two trapezoid by a horizontal line, see Figure 2. Now our trapezoids have three sides aligned with the coordinate axes. We solve our optimization problem for each trapezoid separately. So we must solve the problem.

$$\max \{c_1x_1 + c_2x_2 \; ; \; a_1x_1 + a_2x_2 \le b, L \le x_1 \le U, \; x_1 \text{ and } x_2 \text{ integral}\}$$

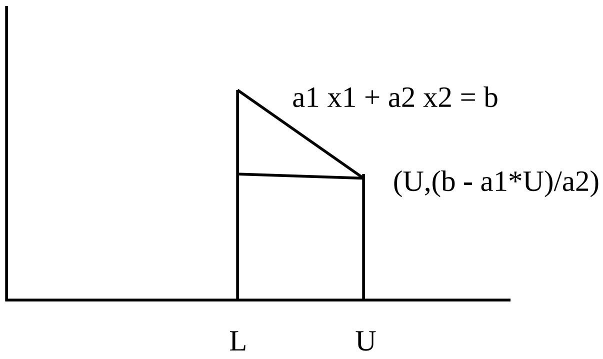

Figure 3:

where a_1, a_2, b_1, b_2 are integral and L and U are rational (they are the x-coordinates of intersections between two constraints). However we may replace $L \leq x_1 \leq U$ by $\lceil L \rceil \leq x_1 \leq \lfloor U \rfloor$ without changing the problem. Hence we may assume that L and U are integral. We may also assume that $a_1 \geq 0$ and $a_2 \geq 0$. Otherwise, we simple replace variables by their negative. Figure 3 illustrates the feasible region.

There are some trivial cases.

- If $c_1 \leq 0$, we set $x_1 = L$ and are left with a problem in a single variable. So we may assume $c_1 > 0$.
- If $a_1 = 0$, we set $x_1 = U$ (since we operate already under the assumption that $c_1 > 0$) and are left with a one-dimensional problem. So we may assume $a_1 > 0$.
- If $a_2 = 0$, the problem is infeasible, if $a_1 L > b$. If $a_2 = 0$ and $a_1 L \leq b$, the problem is feasible. If $c_2 \neq 0$, the problem is unbounded (set x_1 to L and x_2 to an arbitrarily large value of the same sign as c_2). If $c_2 = 0$, we have a problem in a single variable. So we may assume that $a_2 > 0$.
- If $c_2 < 0$, the problem is unbounded (we set x_1 to L and x_2 to an arbitrarily large negative value). If $c_2 = 0$, the optimal solution is (U, z), where z is such that $a_1 U + a_2 z \leq b$, i.e., $z = \lfloor (b - a_1 U)/a_2 \rfloor$. So we may assume $c_2 > 0$.
- If $L = U$, we have a problem in a single variable.
- We now have $a_1 > 0$, $a_2 > 0$, $c_1 > 0$, and $c_2 > 0$ and $L < U$.

We split our feasible region into a rectangle R and a triangle T. It is trivial to find the optimal point in the rectangle. Integral points in the triangle satisfy

$$L \leq x_1 \leq U, x_2 \geq \lceil (b - a_2 U)/a_1 \rceil, a_1 x_1 + a_2 x_2 \leq b \ .$$

We shift the triangle to the origin by the transformation

$$y_1 = x_1 - L \quad \text{and} \quad y_2 = x_2 - \lceil (b - a_1 U)/a_2 \rceil \ .$$

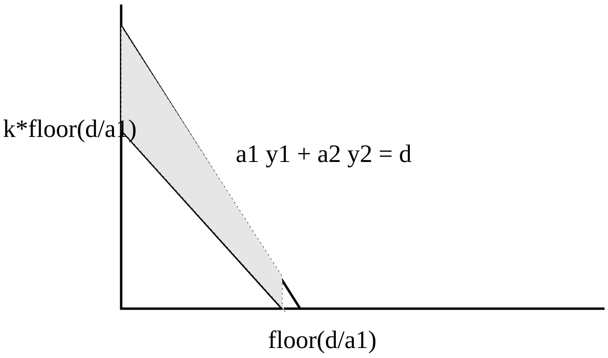

Figure 4: The optimal point lies in the emphasized region. We have $y_1 \geq 0$, $y_2 \geq 0$, $a_1y_1 + a_2y_2 \leq d$ (and hence $y_1 \leq \lfloor d/a_1 \rfloor$) and $y_2 \geq (\lfloor \frac{d}{a_1} \rfloor - y_1)k$.

Our problem changes into

$$\max \{c_1y_1 + c_2y_2 \; ; \; y_1 \geq 0, y_2 \geq 0, a_1y_1 + a_2y_2 \leq d = b - a_1L - a_2\lceil (b - a_1U)/a_2 \rceil, \; y_1 \text{ and } y_2 \text{ integral}\}$$

The constraint $y_1 \leq U - L$ is redundant and we dropped it. Recall $a_1 > 0$, $a_2 > 0$, $c_1 > 0$, and $c_2 > 0$. If $d < 0$, the problem is infeasible. If $d = 0$, $(0, 0)$ is the unique feasible and hence optimal solution.

So assume that $d > 0$. We may also assume $a_1 \geq a_2$; otherwise, we simply exchange the roles of y_1 and y_2. Let $k = \lfloor a_1/a_2 \rfloor$ and let $p = a_1 - ka_2$. Since $c_2 \geq 0$ and $y_2 \geq 0$, the optimal choice for y_2 given any choice for y_1 is $y_2 = \lfloor (d - a_1y_1)/a_2 \rfloor$. (We have $y_2 \leq (d - a_1y_1)/a_2$ by our constraint and hence $y_2 \leq \lfloor (d - a_1y_1)/a_2 \rfloor$ since y_2 is integral. Since c_2 is positive, we should choose y_2 maximal). Next observe that

$$y_2 = \lfloor \frac{(d - a_1y_1}{a_2} \rfloor = \lfloor (\frac{d}{a_1} - y_1)\frac{a_1}{a_2} \rfloor \geq (\lfloor \frac{d}{a_1} \rfloor - y_1)\lfloor \frac{a_1}{a_2} \rfloor = (\lfloor \frac{d}{a_1} \rfloor - y_1)k$$

where the inequality is argued as follows: if r and s are non-negative then $rs \geq \lfloor r \rfloor \lfloor s \rfloor$ and hence (since the right hand side is integral) $\lfloor rs \rfloor \geq \lfloor r \rfloor \lfloor s \rfloor$. We are in this situation since $y_1 \leq d/a_1$ and $a_1/a_2 \geq 1$. *We may therefore add the inequality*

$$y_2 \geq (\lfloor \frac{d}{a_1} \rfloor - y_1)k$$

without changing the optimal solution, see Figure 4. Our feasible region is now:

$$y_1 \geq 0, \; y_2 \geq 0, \; a_1y_1 + a_2y_2 \leq d, \; y_1 \leq \lfloor d/a_1 \rfloor \; y_2 \geq (\lfloor \frac{d}{a_1} \rfloor - y_1)k, \; y_1 \text{ and } y_2 \text{ integral} \; .$$

We next apply the transformation

$$z_1 = y_1 \quad z_2 = y_2 - k(\lfloor d/a_1 \rfloor - z_1)$$

and obtain

$$c_1y_1 + c_2y_2 = c_1z_1 + c_2(z_2 + k(\lfloor d/a_1 \rfloor - z_1)) = c_2k\lfloor d/a_1 \rfloor + (c_1 - kc_2)z_1 + c_2z_2$$

and

$$a_1y_1 + a_2y_2 = a_2k\lfloor d/a_1 \rfloor + (a_1 - ka_2)z_1 + a_2z_2 = a_2k\lfloor d/a_1 \rfloor + pz_1 + a_2z_2 \ .$$

Thus our problem is equivalent to

$$\text{maximize} \quad c_2k\lfloor d/a_1 \rfloor + (c_1 - kc_2)z_1 + c_2z_2$$

subject to

$$pz_1 + a_2z_2 \leq e = d - a_2k\lfloor d/a_1 \rfloor, \ z_1 \leq \lfloor d/a_1 \rfloor, \ z_1 \geq 0, \ z_2 \geq 0, \ z_1 \text{ and } z_2 \text{ integral}$$

and hence we are back to a problem of the same format as we started out with. If we are lucky the modified problem falls under one of the trivial cases discussed at the beginning of the proof.

If not, in what sense did we make progress? We started with a constraint $a_1x_1 + a_2x_2 \leq b$ with $a_1 \geq a_2$ and ended with a constraint of the form $pz_1 + a_2z_2 \leq e$. Here $p = a_1 - a_2\lfloor a_1/a_2 \rfloor < a_2$. In other words the product of the coefficient reduced from a_1a_2 to pa_2. If $a_1 \geq (3/2)a_2$, $pa_2 \leq a_2a_2 \leq (2/3)a_1a_2$. If $a_1 < (3/2)a_2$, $p = a_1 - a_2 \leq (1/2)a_2$, and hence $pa_2 \leq (1/2)a_2a_2 \leq (1/2)a_1a_2$. In either case the product of the coefficients of the constraint is reduced by at least[1] 2/3. Thus we are guaranteed to be in one of the trivial cases after at most $\log(1 + a_1a_2)$ iterations. Each iteration requires a constant number of arithmetic operations on integers of length L. The time is therefore bounded by $O(L^3)$.

References

[CCPS98] W.J. Cook, W.H. Cunningham, W.R. Pulleyblank, and A. Schrijver. *Combinatorial Optimization.* John Wiley & Sons, Inc, 1998.

[Kan80] R. Kannan. A polynomial algorithm for the two-variable integer linear programming problem. *JACM*, 27:133–145, 1980.

[Sch86] A. Schrijver. *Theory of Linear and Integer Programming.* Wiley, 1986.

[1]The reader may wonder why the two cases are unsymmetric. We could more generally, make the case distinction at $a_1 \geq \alpha a_2$ where α is a yet to be determined constant. If If $a_1 \geq \alpha a_2$, $pa_2 \leq a_2a_2 \leq (1/\alpha)a_1a_2$. If $a_1 < \alpha a_2$, $p = a_1 - a_2 \leq (\alpha - 1)a_2$, and hence $pa_2 \leq (alpha - 1)a_2a_2 \leq (\alpha - 1)a_1a_2$. Now choose α such that $\alpha - 1 = 1/\alpha$ or $\alpha^2 = 1 + \alpha$.