Course Notes
for
MS4315 Operations Research 2

J. Kinsella

October 29, 2015

Contents

About the Course

- Lectures: (J Kinsella: Weeks 7–12)
  - Monday 16:00 Lecture CG–058
  - Thursday 13:00 Lecture CG–055
- Tutorials:
  - Monday 17:00 C1–062
- Office hours: Tuesdays 14:00 & Wednesdays 17:00. B3-043.
- An attendance record will be kept.
- Assessment will be via end-of-semester written examination.

- Syllabus: (sequencing subject to change); **JK**, **MB**
  - **Game Theory - Concepts of equilibrium, matrix games, extensive form games and repeated games.**
  - **Applications of game theory - models of economic competition (Cornot, Bertrand), evolutionary game theory.**
  - **Integer programming - pure integer programming algorithms,**
  - **Branch & bound solutions to integer programs.**

- The Integer Programming Part of the Course consists of
  - Chapter 1 Integer Programs; Examples & Formulations
  - Chapter 2 Quality of IP Formulations
  - Chapter 3 Relaxations and Bounds
  - Chapter 4 IP Solution Techniques
- The material is based closely on "Integer Programming" by Laurence A. Wolsey and on "Introduction to Operations Research" by J. G. Ecker & M. Kupferschmid.

1 Integer Programs; Examples & Formulations

1.1 Linear Programs — A Reminder

Before specialising to Integer Linear Programs (IPs) in the next Section I need to remind you what Linear Programs (LPs) are and how they are solved. I'll briefly do the first in this very short Section and summarise the solution method (Simplex) in an Appendix.

Linear Programs A Linear Program (LP) is a minimisation (or maximisation) problem on $\mathbb{R}^n$ where the "objective function" to be minimised is linear (so $f(x) = c^T x + c_0$) and all the constraints are linear (either equality constraints $Ax = b$ or inequality constraints $Ax \leq b$ or a combination of both.)

I will usually drop the constant term c_0 as it has no effect on the solution, just adds to or subtracts from the value of the objective function.

(I can convert from max to min by simply replacing c by $-c$.)

There are many variations on the above — all can be transformed using simple matrix techniques into so-called "Standard Form":

Definition 1.1 (Linear Program in Standard Form) *Given* $c \in \mathbb{R}^n$, *an* $m \times n$ *real matrix* A *and* $b \in \mathbb{R}^m$ *I can write a Linear Program in Standard Form as:*

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \; & c^T x \\ Ax &= b \\ x &\geq 0 \end{aligned} \tag{1.1}$$

Variant Forms of Linear Program It is easy to transform any linear program to standard form. For instance, given the problem

$$\min c^T x, \text{subject to } Ax \leq b$$

(without any bounds on x), I can convert the inequality constraints to equalities by introducing a vector of **slack** variables z and writing

$$\min c^T x, \text{ subject to } Ax + z = b,\ z \geq 0 \tag{1.2}$$

or just

$$\min c^T x, \text{ subject to } \begin{bmatrix} A & I \end{bmatrix} \begin{bmatrix} x \\ z \end{bmatrix} = b,\ z \geq 0 \tag{1.3}$$

This form is still not quite standard, since not all the variables are constrained to be nonnegative. I deal with this by splitting x into its nonnegative and nonpositive parts, $x = x^+ - x^-$, where $x^+ = \max(x, 0) \geq 0$ and $x^- = \max(-x, 0) \geq 0$. The problem (1.2) can now be written as

$$\min \begin{bmatrix} c \\ -c \\ 0 \end{bmatrix}^T \begin{bmatrix} x^+ \\ x^- \\ z \end{bmatrix} \text{ s.t. } \begin{bmatrix} A & -A & I \end{bmatrix} \begin{bmatrix} x^+ \\ x^- \\ z \end{bmatrix} = b \, , \begin{bmatrix} x^+ \\ x^- \\ z \end{bmatrix} \geq 0. \quad (1.4)$$

which clearly has the same form as (1.1).

How Do I Solve LP's? You know the answer — the Simplex Method (though there are other interesting methods, Simplex is still the most widely used).

See Appendix A.1 for a review. (You won't need this initially but the solution method for Integer Programs uses Simplex as a component and you will need to have read the Appendix before I get to Chapter 4.)

See Appendix A.5 for an explanation of the Dual Simplex method, needed in Section 4.3.

1.2 What is an Integer Program?

The most general class of problem that I'll mention is the class of Mixed Integer Programs, defined on the next Slide.

Definition 1.2 (Linear) Mixed Integer Program (MIP))
A MIP is a linear programming problem (LP) with the added restriction that some or all of the variables must take integer values.

$$\max_{x_1,\ldots,x_n,\ y_1,\ldots,y_p} \sum_{j=1}^{n} c_j x_j + \sum_{j=1}^{p} h_j y_j$$

such that

$$\begin{cases} \sum_{j=1}^{n} a_{ij} x_j + \sum_{j=1}^{p} g_{ij} y_j \leq b_i, i = 1, \ldots, m \\ x_1, \ldots, x_n \geq 0 \\ y_1, \ldots, y_p \geq 0, y_j \in \mathbb{Z}, j = 1, \ldots, p. \end{cases}$$

Here the variables x_i are only required to be non-negative while the y_j are in addition required to be integer.

In matrix/vector notation I can re-write Definition 1.2 as

Definition 1.3 (MIP in Matrix/Vector notation) *If the vectors of variables are* $x \in \mathbb{R}^n$ *and* $y \in \mathbb{R}^p$ *and* A *is a real* $m \times n$ *matrix,* $b \in \mathbb{R}^m$, $c \in \mathbb{R}^n$, G *a real* $m \times p$ *matrix,* $h \in \mathbb{R}^p$ *then MIP can be written as:*

$$\max_{x,y} c^T x + h^T y$$

such that

$$Ax + Gy \leq b, \quad x \geq 0 \text{ and real}, y \geq 0 \text{ and integer}.$$

Exercise 1.1 *Does this definition cover the (?) special case*

$$Ax \leq b_x, \quad Gy \leq b_y, \quad x \geq 0 \text{ and real}, y \geq 0 \text{ and integer}?$$

Other Problem Formats The above is a **maximization** problem; the objective could also be to

$$\min c^T x + h^T y. \quad \text{(Just replace } c \text{ by } -c \text{ and } h \text{ by } -h\text{)}.$$

The above problem contains only "less-than-or-equal-to" constraints; I could also have

$$\sum_{j=1}^{n} a_{ij} x_j + \sum_{j=1}^{p} g_{ij} y_j \geq b_i, \text{ for some } i\text{'s and/or}$$

$$\sum_{j=1}^{n} a_{ij} x_j + \sum_{j=1}^{p} g_{ij} y_j = b_i, \text{ for some } i\text{'s.}$$

In the above problem all variables are constrained to be nonnegative; we could also have variables unrestricted in sign.

Special cases of MIP

Definition 1.4 *If all variables are integer: I have a (Linear) Integer Program (IP):*

$$\max_{x \in \mathbb{R}^n} \{c^T x : \ Ax \leq b, x \geq 0, x \in \mathbb{Z}^n\}.$$

I will concentrate on IP's in this course.

I will sometimes specialise further to "Binary Integer Programs" (BIP's) where some or all of the variables are restricted to binary $\{0, 1\}$ values.

Definition 1.5 (BIP)

$$\max_{x \in \mathbb{R}^n} \{c^T x : Ax \leq b, x \in \{0, 1\}^n\}$$

(Matlab solves these with `bintprog` — there is no built-in IP solver in Matlab but `bintprog` can easily be extended to solve general IP's.)

How to Convert an IP to a BIP (and back) The trick is to decide in advance how many binary digits (bits) it will take to code all the elements of the solution. Of course I don't know in advance how big the entries in x^* (the integer solution to the IP) will be.

Suppose that I decide to code the solution with $N+1 = 11$ binary digits ($N = 10$) based on knowledge of the problem so that any number between 0 and 2027 ($1 + 2 + 2^2 + \cdots + 2^N \equiv 2^{N+1} - 1$ equals $2^{11} - 1$) could be "coded" as an $N+1$–bit binary number.

If the IP to be solved is $\min c^T x$, subject to $Ax \leq b, x \geq 0, x \in \mathbb{Z}^n$

then **check** that the solution $x_b \in \{0,1\}^{n(N+1)}$ to BIP $\min c_b^T x$, subject to $A_b x_b \leq b, x_b \geq 0, x \in \{0,1\}^{n(N+1)}$ gives the solution x^* to the IP.

Take $c_b = [c', 2c', 4c', \ldots 2^N c']'$, a tall column of length $n(N+1)$. Also $A_b = [a_1, 2a_1, \ldots 2^N a_1, a_2, 2a_2, \ldots 2^N a_2, \ldots, a_n, 2a_n, \ldots 2^N a_n]$ where $a_1, \ldots, a_n$ are the columns of the matrix A. So A_b is an $m \times n(N+1)$ matrix.

The solution x_b is a **binary** column vector of length $n(N+1)$. To reconstruct the solution x^* just write
$x^* = \sum_{i=1}^{n} \sum_{j=0}^{N} x_{[(i-1)(N+1)+j+1]} 2^j$.

If you have access to `matlab`'s `bintprog` you can easily use this coding & decoding to solve any IP.

BIP's are easier to solve in general but the above coding procedure for IP's generates **big** IP's!

In `octave` the `glpk` package is included — it can solve both LP's and IP's (not just BIP's).

Combinatorial Optimisation Problems These are another problem type. "Combinatorial" ≡ "re-arrangements". See http://en.wikipedia.org/wiki/Combinatorial_optimization

Typically, I have a finite set $N = \{1, \ldots, n\}$, weights c_j for each $j \in N$ and a set $\mathcal{F}$ of feasible subsets of N. The problem (see Ex. 1.9 below) of finding a minimum (maximum) weight feasible subset is a Combinatorial Optimisation Problem.

Definition 1.6 (Combinatorial Optimisation Problem)

$$\min_{S \subseteq N} \left\{ \sum_{j \in S} c_j : S \in \mathcal{F} \right\}$$

In words; "pick the set S (in the list $\mathcal{F}$) that has the smallest weight/cost $\sum_{j \in S} c_j$".

Easy if only a small number of feasible sets — just check them all.

Very difficult if there are billions.

Formulating a problem as an optimisation problem

- Remember from OR1 that an optimisation problem is a mathematical model of a real world problem.
- To model a problem as an optimisation problem, the following steps are typically followed:
  1. Define what appear to be the necessary variables; usually quantitative expressions of decisions that need to be made.
  2. Use these variables to define a set of constraints; values of variables feasible for the constraints must correspond to feasible solutions to the problem and vice versa.
  3. Use these variables to define the objective function to be minimised or maximised.
- Remember that your original definition of variables may need to be revised and that there is often more than one way to model a problem.

1.2.1 Why Not Just Check all the Possible Solutions?

It is a reasonable enough idea — the solution has to be all-integer, there can't be that many integer points that satisfy all the constraints?

- Unfortunately there can.
- Some examples:
  - The Assignment problem (see Example 1.3 below) is a standard IP problem. A problem of size n has $n!$ solutions.
  - The Knapsack (Example 1.2) and Set Covering Problems (see below) — in both cases the number of possible solutions is $O(2^n)$.
  - The Travelling Salesman Problem (see Example 2.1 below). With n cities, there are $(n-1)!$ feasible tours.

- To interpret the above examples, consider the Table:

n	2^n	$n!$
10	1.02×10^3	3.6×10^6
100	1.27×10^{30}	9.33×10^{157}
1000	1.07×10^{301}	4.02×10^{2567}

- So solving even a small ($n = 100$) TSP by "Brute Force" is hopeless and solving a modest-sized ($n = 1000$) Knapsack problem is equally so.
- I need something better.

1.2.2 Are IP's Just a Special Case of LP's?

The obvious strategy for solving IP's/MIP's is to drop the requirement that the solution must be integer and use the Simplex Method to solve the revised problem, then round off the solution to the nearest whole numbers.

This is a good idea (and is a component of the standard method for solving IP's, the Branch & Bound Method, studied in Ch. 3)

Unfortunately it doesn't work on its own. ☹

An Example:

Example 1.1 *Consider the IP:*

$$\begin{aligned} \max 1.00x_1 + 0.64x_2 & \\ 50x_1 + 31x_2 &\leq 250 \\ 3x_1 - 2x_2 &\geq -4 \\ x_1, x_2 \geq 0, x_1, x_2 \in \mathbb{Z} & \end{aligned}$$

If I drop the integer requirement and solve Ex. 1.1 as an LP (using the Simplex Method) I find the solution $(376/193, 950/193) \approx (2, 5)$. But the optimal integer solution is $(5, 0)$.

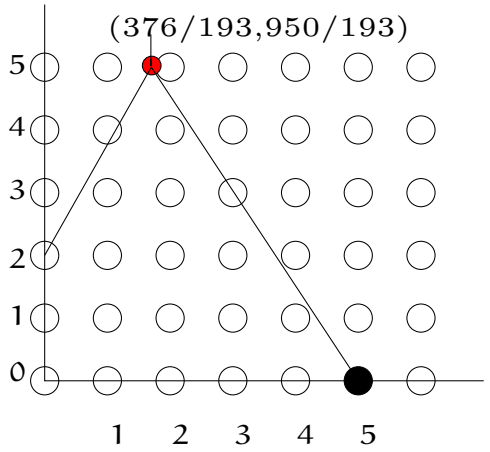

Figure 1: Rounding an LP

The IP optimal point is $(5, 0)$ while the LP optimal point is $(376/193, 950/193) \approx (2, 5)$.

So I need special techniques for IP's— to be discussed in Ch. 4. (But the Simplex Method will play a part.)

1.3 BIP Examples

Let's look at some classic examples of IP's, BIP's initially.

Example 1.2 (0–1 Knapsack Problem) *There is a budget* b *available for investment in projects during the coming year and* n *projects are under consideration, where* a_j *is the cost of investing in project* j *and* c_j *is the expected return. The goal is to choose a set of projects so that the budget is not exceeded and the expected return is maximized.*

Variables: $x_j = 1$ *if project* j *is selected,* 0 *otherwise.*

Problem Formulation: $\max \sum_{j=1}^{n} c_j x_j$, *(Expected return). s.t.*

- $\sum_{j=1}^{n} a_j x_j \leq b$ *(the budget cannot be exceeded)*
- $x_j \in \{0, 1\}, j = 1, \ldots, n$ *(all variables are binary).*

Example 1.3 (Assignment Problem) *There are* n *people available to carry out* n *jobs, all of which must be completed. Each person is assigned to carry out exactly one job. The estimated cost (pay) of assigning person* i *to job* j *is* $c_{ij}, i, j = 1, \ldots, n$. *The problem is to find a minimum cost assignment.*

Variables: $x_{ij} = 1$ *if person* i *does job* j *and* $x_{ij} = 0$ *otherwise.*
Problem Formulation: $\min \sum_{i=1}^{n} \sum_{j=1}^{m} c_{ij} x_{ij}$ *(Cost of the assignment) s.t.*

1. $\sum_{j=1}^{n} x_{ij} = 1, i = 1, \ldots, n$ *(Each person* i *does exactly one job),*
2. $\sum_{i=1}^{n} x_{ij} = 1, j = 1, \ldots, n$ *(Each job* j *is assigned to exactly one person)*
3. $x_{ij} \in \{0, 1\}, i, j = 1, \ldots, n$ *(All variables are binary).*

(Look carefully at how the first & second constraints work.)

The solution $\mathfrak{x}$, viewed as an $\mathfrak{n} \times \mathfrak{n}$ matrix with exactly one $\mathfrak{1}$ in each row & in each column, is just an identity matrix with the rows (or columns) shuffled.

Check how many such re-arrangements are possible? (I have answered the question on a previous Slide.)

Example 1.4 ("Set Covering" Example Problem) *A hospital A&E needs to keep doctors on call 24/7 so that a qualified person is available to perform every medical procedure that might be required (take it that there is an official list of such procedures).*

- *For each potential on-call doctor, I know the additional on-call bonus they need to be paid if selected and the "sub-set" of procedures that they are qualified to perform.*
- *The goal is to choose doctors so that each procedure is covered, at a minimum cost (different "grades" of doctor must be paid different levels of on-call bonus).*

Example 1.5 (Set Covering Problem Formulation) *I need notation for the problem data:*

Data representation: *(i.e. define the "incidence matrix"* A*)*

- *With* m *procedures and* n *available doctors, the data can be represented as an* $m \times n$ *binary matrix* A *where* $a_{ij} = 1$ *if doctor* j *can perform procedure* i *and* $a_{ij} = 0$ *otherwise.*
- *Also, let* $c_j, j = 1, \ldots, n$ *be the bonus that will need to be paid to doctor* j *for on-call duty.*

Variables: $x_j = 1$ *if doctor* j *is selected to be on call and* 0 *otherwise.*

Problem Formulation: $\min \sum_{j=1}^{n} c_j x_j$ *(***minimise total bonuses paid***) s.t.*

- $\sum_{j=1}^{n} a_{ij} x_j \geq 1, i = 1, \ldots, m; \quad x_j \in \{0, 1\}, j = 1, \ldots, n$ *(***at least one on-call doctor** j **must cover procedure** i*).*

Stopped here 17:00, Monday Week 7

Example 1.6 (General Set Covering Problem) *Suppose I am given a set* $M = \{1, \ldots, m\}$ *and* n *subsets* $M_j, j = 1, ..., n$ *of* M *together with weights (costs)* c_j *for each of the subsets* M_j , $j = 1, ..., n$.

Define a **Set Cover** *as a set* $T \subseteq \{1, ..., n\}$ *such that* $\cup_{j \in T} M_j = M$ *(i.e. a choice of some or all of the subsets* M_j *that covers* M*).*

In the above hospital example, $M = \{1, \ldots, m\}$ *is the entire set of procedures,* M_j *is the set of procedures that doctor* j *can perform and* c_j *is doctor* j*'s call-out bonus if selected. Then* T *is the set of doctors on call; note that it must be a cover of* M *(why?).*

In general the incidence matrix A *is defined by:* $a_{ij} = 1$ *if* $i \in M_j$ *and* 0 *otherwise.*

The hospital problem is a "minimum weight set cover" problem.

Example 1.7 (General Set Covering Problem Formulation) *Given a set* M *and* n *subsets* M_j *of* M *together with a weight* c_j *corresponding to each* M_j*. Then the minimum set covering problem can be stated as:*

$$\min_{T} \left\{ \sum_{j \in T} c_j : T \text{ is a cover for } M \right\}$$

$$\equiv \min \left\{ \sum_{j=1}^{n} c_j x_j : Ax \geq e, x \in \{0,1\}^n \right\}$$

where A *is the "incidence matrix" as above and* e *is a vector of* 1*'s. The variables* $x_j = 1$ *if* M_j *selected and* 0 *otherwise.*

Again, think carefully about the "coding" of the cover constraint on the right above — can you explain it?

Example 1.8 (Simple Set Covering Example) *Let* $M = \{1, 2, 3, 4, 5\}$. *Let* $M_1 = \{1, 2\}$, $M_2 = \{3, 4\}$, $M_3 = \{2, 5\}$, $M_4 = \{5\}$, $M_5 = \{1, 5\}$ *and* $M_6 = \{1, 4, 5\}$ *be* 6 *subsets of* M *(out of the 32 subsets that I could have chosen).*

There are many choices of cover T, *for example* $T = \{1, 2, 6\}$ *corresponding to choosing* M_1, M_2 *and* M_6. *Given a weight* c_j *for each* M_j *I could first list all possible covers (as the problem is small) then find the one with smallest total weight.*

What is the incidence matrix A *for this toy problem?* A *must be* $m \times n$. *In this case* $n = 6$ *and* $m = 5$.

(Remember that in general the incidence matrix A *is defined by:* $a_{ij} = 1$ *if* $i \in M_j$ *and* 0 *otherwise.) Can you write down the matrix* A*?*

Example 1.9 (Related problem: Maximum Weight Set Packing) *Again, I am given a set* M *and* n *subsets of* M, $M_j \subseteq M$ *for* $j = 1, \ldots, n$ *together with weights* $c_j, j = 1, \ldots, n$ *for each of the subsets.*

A **Set Packing** *is a collection* $T \subseteq \{1, \ldots, n\}$ *such that* $M_j \cap M_k = \varnothing$ *for all* $j, k \in T, j \neq k$.. *(A selection of disjoint subsets of* M*.)*

Maximum weight set packing problem*:*

$$\max_T \left\{ \sum_{j \in T} c_j : T \text{ is a packing} \right\}$$

$$\equiv \max \left\{ \sum_{j=1}^{n} c_j x_j : Ax \leq e, x \in \{0,1\}^n \right\}.$$

Example 1.10 (Maximum Weight Set Packing in words)
"Find the selection of disjoint subsets of $\mathbb{M}$ *that has the largest total weight (payoff)".*

Can you explain the packing constraint on the right in the previous Slide?

Reanalyse the toy problem above as a Set Packing problem.

Example 1.11 (Related problem: Set Partition) *Again, I am given a set* M *and* n *subsets of* M, $M_j \subseteq M$ *for* $j = 1, \ldots, n$ *together with weights* $c_j, j = 1, \ldots, n$ *for each of the subsets.*

A **Set partition** *is a selection* $T \subseteq \{1, \ldots, n\}$ *which is both a cover and a packing. (A selection of disjoint subsets of* M *that provides a* **disjoint covering** *of* M*.)*

Maximum/minimum weight set partition problem*:*

$$\max_T \left\{ \sum_{j \in T} c_j : T \text{ is a partition} \right\}$$

$$\equiv \max \left\{ \sum_{j=1}^{n} c_j x_j : Ax = e, x \in \{0,1\}^n \right\}.$$

Again, can you explain the constraint on the right?

1.4 Exercises

1. I'll just give you one problem to work on — no mathematics needed but some thought. You are not asked to solve the problem, just to pose (formulate) it as an Integer Program.
   - A student is attending a summer school in Operations Research.
   - There are four courses (**A**lgorithms, **B**ranch & Bound, **C**utting Plane Methods & **D**ynamic Programming) running — he/she must attend a class in each every day.
   - Each class lasts an hour but because of the large number of students each course is taught several times each day by different lecturers.

- I'll refer to version ν of class c as (c, ν) and say that it meets at the hour $t_{c\nu}$, where classes begin on the hour each day at times $1, \ldots, N$.
- The student's choice for what time slot (version) to choose for each course on a given day is influenced by the lecturer's reputation and the time of day.
- Let $p_{c\nu}$ be the student's preference for (c, ν).
- Due to clashes (two preferred choices scheduled at the same time), the student cannot always make the choice that he/she prefers.

(a) Formulate an IP to choose a feasible course schedule that maximises the student's preferences.

(b) Modify the formulation so that the student never has more than two classes in a row.

(c) Modify the formulation so that the student chooses a schedule in which his/her first class of the day is as late as possible. (Tricky.)

(d) What if (unlikely) the student wanted his/her first class as early as possible?

(e) Or if the student wanted his/her last class as early as possible?

Hints:

- As the problem stands it is difficult to formulate as an IP.
- Try reformulating to work with course c and time t rather than course c and version v so that your decision variable is x_{ct} which is one if the student attends class c at time t and zero otherwise.
- – For (c), consider how to minimise the maximum element of a vector y, say.
  – If $Z \geq y_i$ for $i = 1, \ldots, N$ then $Z \geq \max y$ so minimising Z subject to these constraints will minimise the max element of y.
  – So come up with a formula for y so that making its min element large makes the time of the first scheduled lecture as late (large) as possible.

Solution:

(a)
- Use the info in t_{cv} to pre-calculate $s_{ct} = 1$ if course c has a version scheduled at time t, 0 otherwise.
- Also pre-calculate p_{ct} from the info in p_{cv}.
- Define the binary variable x_{ct}.
- Obj fun is $z = \max \sum_{c,t} p_{ct} x_{ct} s_{ct}$.

Constraints
- $\sum_{c} x_{ct} \leq 1, \forall t$ — no more than one class to be chosen at any given time.
- Also $\sum_{t} x_{ct} s_{ct} = 1, \forall c$ — each course attended exactly once each day.

(b) Additional constraint:

$$\sum_{c} (x_{ct} + x_{c,t+1} + x_{c,t+2}) \leq 2, \forall t = 1, \ldots, T-2.$$

(c) Define $Y = \sum_{c} x_{ct}.\frac{1}{t}$, $t = 1, \ldots, N$.

- Y will look like $(0, 0, \ldots, 0, 1/t_1, 0, 0, \ldots, 1/t_4, 0, \ldots, 0)$.
- I can find the max element of Y $(1/t_1)$ by adding the variable Z and adding the constraints $Z \geq Y_t, t = 1, \ldots, N$.
- Any feasible Z will be $\geq 1/t_1$.
- I want $1/t_1$ as small as possible so minimise Z.
- So replace the objective function by $z = \min Z$ subject to the extra constraints $Z \geq Y_t, t = 1, \ldots, N$.

(d) So I want to make $1/t_1$ as **large** as possible. So with the same definition for Y, solve the problem
$\max Z$, s.t.$Z \geq Y_t, t = 1, \ldots, N$, with the extra condition $Z \leq 1$.
(I added the upper bound on Z as otherwise the problem might be unbounded.)

(e) Simpler. Re-define $Y = \sum_c x_{ct}.t$, $t = 1, \ldots, N$. I want to maximise t_4 which is the maximum of the re-defined Y. So maximise Z s.t.$Z \geq Y_t, t = 1, \ldots, N$, with the extra condition $Z \leq N$ again introduced to avoid unboundedness.

2 Quality of IP Formulations

I will start with some more examples of IP's, then see how different "formulations" can be compared.

2.1 BIP Examples

Example 2.1 (Traveling Salesman Problem (TSP)) *(The most famous IP of them all!)*

- *A salesman must visit each city in a set* $N = \{1, ..., n\}$ *of cities exactly once and return to the starting point.*
- *The time it takes to travel from city* i *to city* j *is* c_{ij}.
- *Find the order in which he should make his tour so as to finish in minimal time.*

Variables: $x_{ij} = 1$ *if the salesman goes from city* i *directly to city* j *and* 0 *otherwise (diagonal elements* x_{ii} *of the "time/distance matrix" need not be and are not defined).*

Objective function: $\min \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij}$.

Constraints:

1. $x_{ij} \in \{0, 1\}, \forall i \neq j$,
2. $\sum_{j:j \neq i} x_{ij} = 1, i \in N$ *(Leave each city* i *exactly once)*
3. $\sum_{i:i \neq j} x_{ij} = 1, j \in N$ *(Enter each city* j *exactly once)*

This is not three constraints but three **sets** *of constraints; how many?*

N.B. So far, these are just the constraints for an Assignment problem — see Example 1.3.

A typical solution for an assignment problem is a matrix x *with a single* 1 *in each row and in each column. (For a general assignment problem, the diagonal elements may also be used.)*

For example: $x =$

$$\begin{bmatrix} - & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & - & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & - & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & - & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & - & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & - & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & - & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & - \end{bmatrix}$$

But this matrix $\mathbf{x}$ *generates a "tour" of the form*

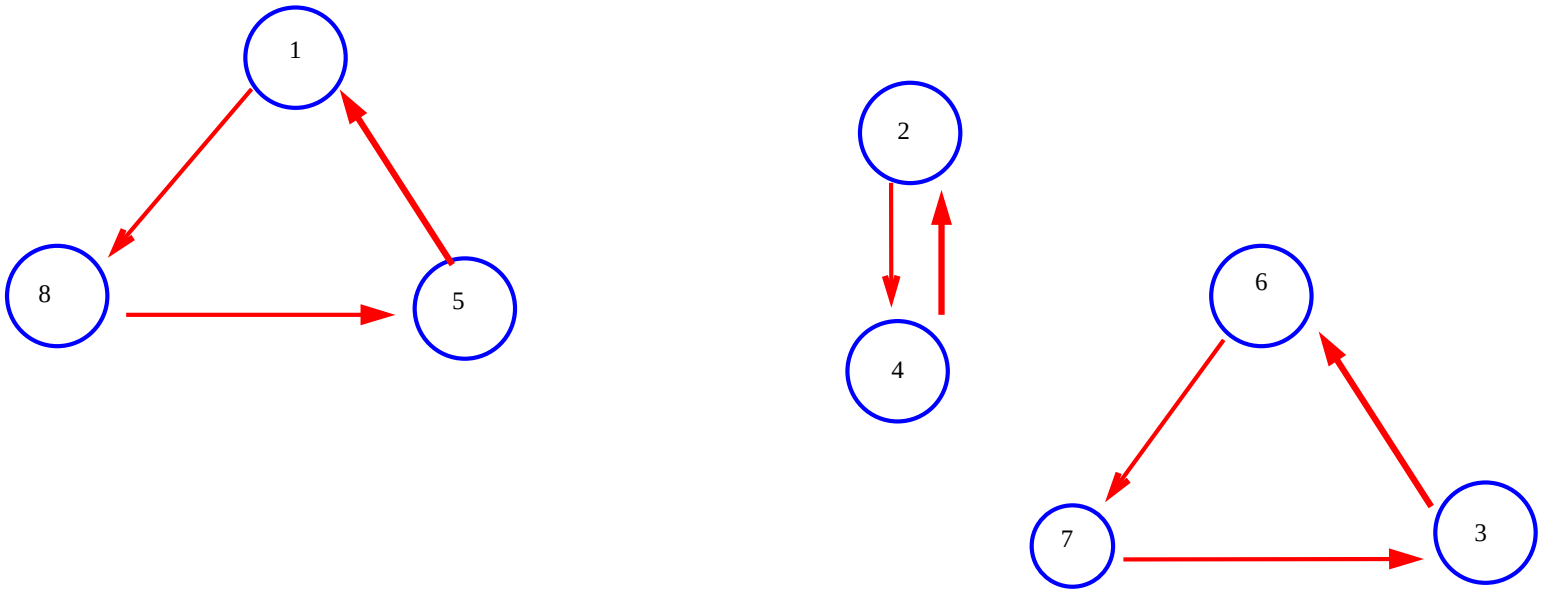

Figure 2: TSP with Subtours

There are three "subtours" $(1,8,5,1)$, $(4,2,4)$ *and* $(6,7,3,6)$ — *I need three salesmen to visit all 8 "cities"!*

I need extra constraints for a TSP to avoid **subtours** *— clumps of separate tours as in the Figure.*

Example 2.2 (Eliminating subtours in TSP) **Either** *of the following additional constraints will prevent subtour solutions being accepted:*

1. *For* **all** $2^n - 2$ *(why?) non-empty strict subsets* S *of* N*, any tour must leave* S *at least once (the so-called "cut-set constraints"):*

$$\sum_{i \in S} \sum_{j \notin S} x_{ij} \geq 1, \forall S \subset N, S \neq \varnothing \tag{2.1}$$

2. *Alternatively; for* **all** $2^n - n - 2$ *(why?) non-empty subsets* S *of* N *with cardinality* $2 \leq |S| \leq n-1$ *(i.e. other than the full set itself and single points/cities) require that there are at most* $|S|-1$ *"links" from elements of the set* S *to other elements of* S *("sub-tour elimination"):*

$$\sum_{i \in S} \sum_{j \in S \setminus \{i\}} x_{ij} \leq |S| - 1 \text{for } S \subset N, 2 \leq |S| \leq n-1. \qquad (2.2)$$

The three sets of constraints on Slide 45 together with **either** (2.1) *or* (2.2) *are valid Problem Formulations of the TSP.*

(How many extra constraints have I added by including (2.1) *or* (2.2)*?)*

2.2 MIP Examples

Some MIP's:

Example 2.3 (Uncapacitated Lot Sizing (ULS))
("Uncapacitated" means that I do not take into account limits on production capacity.)

I need to decide on a production plan for the next $\mathtt{n}$*-day (or hour or week or month) horizon for a single product.*

Data for the problem is as follows:

- $\mathtt{f_t}$ — *fixed cost of producing in period* t,
- $\mathtt{p_t}$ — *unit production cost in period* t,
- $\mathtt{h_t}$ — *unit storage cost in period* t,
- $\mathtt{d_t}$ — *demand in period* t.

Variables for the problem:

- x_t — *production volume in period* t.
- s_t — *stock level in period* t.
- *Initial stock level* $s_0 = 0$.
- $y_t = 1$ *if production occurs in period* t, 0 *otherwise.*

Problem formulation:

$$\begin{aligned} \min \quad & \sum_{t=1}^{n} p_t x_t + \sum_{t=1}^{n} h_t s_t + \sum_{t=1}^{n} f_t y_t \\ \text{s.t.} \quad & s_{t-1} + x_t = d_t + s_t, t = 1, \ldots, n. \textit{ (Inventory balance)} \\ x_t \quad & \leq \quad M y_t, t = 1, \ldots, n. \textit{ (Forcing constraints)} \\ & s_t, x_t \geq 0, y_t \in \{0, 1\}, t = 1, \ldots, n. \, s_0 = 0 \end{aligned}$$

Here, $M > 0$ *is a given upper bound on* $x_t, t = 1, \ldots, n$.
Constraints $s_t \geq 0$ *ensure the demand is satisfied in each period.*
The variables $y_t = 1$ *if production occurs in period* t, 0 *otherwise.*

Can you see how x_t *and* y_t *interact?*

Example 2.4 (ULS — Alternative Description) *I can "eliminate" the variable* s *with* $s_t = \sum_{i=1}^{t} x_i - \sum_{i=1}^{t} d_i$ *(Exercise.)*

- *So the Objective becomes*
  $\sum_{t=1}^{n} p_t x_t + \sum_{t=1}^{n} h_t \left(\sum_{i=1}^{t} x_i - \sum_{i=1}^{t} d_i \right) + \sum_{t=1}^{n} f_t y_t.$
- *Re-ordering the sums I can write the problem description as:*

$$\begin{aligned} \min \quad & \sum_{t=1}^{n} c_t x_t + \sum_{t=1}^{n} f_t y_t - K \\ \text{s.t.} \quad & \sum_{i=1}^{t} (x_i - d_i) \geq 0, t = 1, \ldots, n. \textit{ (Equiv to } s_t \geq 0\textit{)} \\ x_t \quad \leq \quad & M y_t, t = 1, \ldots, n. \textit{ (Forcing constraints)} \\ & x_t \geq 0, y_t \in \{0, 1\}, t = 1, \ldots, n. \, s_0 = 0 \end{aligned}$$

Exercise 2.1 *Find expressions for* c_t *and* K.

Example 2.5 (Uncapacitated Facility Location (UFL))
(Again, "Uncapacitated" means that I do not take into account limits on production capacity.)

- *Given a set of potential depots* $N = \{1, \ldots, n\}$ *and a set* $M = \{1, \ldots, m\}$ *of clients, I need to decide which depots to open and how to utilize them to serve clients.*
- *The cost* c_{ij} *of supplying the entire demand of client* i *from depot* j *and the cost* f_j *of building depot* j *are given.*

Variables:

- $y_j = 1$ *if depot* j *is built,* $y_j = 0$ *otherwise;*
- x_{ij} *is the fraction of the demand of client* i *supplied by depot* j

Problem Formulation:

$$
\begin{aligned}
\min \quad & \sum_{i \in M} \sum_{j \in N} c_{ij} x_{ij} + \sum_{j \in N} f_j y_j \\
\text{s.t.} \quad & \sum_{j \in N} x_{ij} = 1, \ \forall i \in M \ \textit{(Demand satisfied)} \\
x_{ij} \quad & \leq \quad y_j, \ \forall i \in M, j \in N \ \textit{(Forcing constraints)} \\
x_{ij} \quad & \geq \quad 0, \ \forall i \in M, j \in N. \\
y_j \quad & \in \quad \{0, 1\}, \ \forall j \in M.
\end{aligned}
$$

(Can you explain how x_{ij} and y_j interact?)

Example 2.6 (An Alternative Problem Formulation for UFL)

I will shortly consider the topic of different ways to **formulate** *IP's & MIP's and the question of which Problem Formulation is best. Here is an alternative Problem Formulation for UFL:*

Same variables: $y_j = 1$ *if depot* j *is built,* $y_j = 0$ *otherwise;* x_{ij} *is the fraction of demand of client* i *supplied by depot* j.

Alternative Problem Formulation:

$$\min \quad \sum_{i \in M} \sum_{j \in N} c_{ij} x_{ij} + \sum_{j \in N} f_j y_j$$

$$\text{s.t.} \quad \sum_{j \in N} x_{ij} = 1, \forall i \in M \ (\textit{As before})$$

$$\sum_{i \in M} x_{ij} \leq m y_j, \forall j \in N. \ (\textit{Weaker constraint} \equiv \textit{more solutions.})$$

$$x_{ij} \geq 0, \forall i \in M, j \in N. \ (\textit{As before})$$

$$y_j \in \{0, 1\}, \forall j \in M. \ (\textit{As before})$$

2.3 Formulations — a Geometric Perspective

For most problems that can be modeled as IP's/MIP's, there are several ways of doing so. To enable us to assess which formulation is better, let's first make the concept of a **formulation** more specific.

Up to now I have used the term **problem formulation** to mean the statement of an IP/MIP consisting of an Objective to be maximised/minimised and a set of constraints.

I will now define the term **set formulation** or just **formulation** to mean a collection of linear inequality constraints satisfied by all feasible points ("a polyhedron containing the feasible region").

First I define the term **polyhedron**.

Definition 2.1 (Polyhedron) *A subset* P *of* $\mathbb{R}^n$ *that can be described by a finite set of linear constraints* $P = \{x \in \mathbb{R}^n : Ax \leq b\}$ *is a polyhedron.*

Definition 2.2 (Formulation for a Set) *A polyhedron* $P \subseteq \mathbb{R}^{n+p}$ *is a* **formulation** *for a set* $X \in \mathbb{Z}^n \times \mathbb{R}^p$ *if and only if* $X = P \cap (\mathbb{Z}^n \times \mathbb{R}^p)$.

For a pure IP, the above reduces to

Definition 2.3 (Formulation for a Set (IP)) *A polyhedron* $P \subseteq \mathbb{R}^n$ *is a* **formulation** *for a set* $X \in \mathbb{Z}^n$ *if and only if* $X = P \cap \mathbb{Z}^n$ *(the integer elements of* P*).*

Stopped here 14:00, Thursday Week 7

Example 2.7 *Consider the set* $X = \{(1,1),(2,1),(3,1),(1,2),(2,2),(3,2),(2,3)\}$. *What are possible formulations for* X *(in the sense of the above definition)? What if I remove point (2, 2) from the set?*

Example 2.8 (Formulations for a "Knapsack set") *Consider the feasible region of the following* 0/1 *knapsack problem:*

$$\begin{aligned} X &= \left\{x \in \{0,1\}^4 : 83x_1 + 61x_2 + 49x_3 + 20x_4 \leq 100\right\} \\ &= \{(0,0,0,0),(1,0,0,0),(0,1,0,0),(0,0,1,0), \\ &\qquad (0,0,0,1),(0,1,0,1),(0,0,1,1)\} \end{aligned}$$

The following are some formulations for X *:*

$$\begin{aligned} P_1 &= \left\{x \in \mathbb{R}^4 : 0 \le x \le e,\ 83x_1 + 61x_2 + 49x_3 + 20x_4 \le 100\right\} \\ P_2 &= \left\{x \in \mathbb{R}^4 : 0 \le x \le e,\ 4x_1 + 3x_2 + 2x_3 + x_4 \le 4\right\} \\ P_3 &= \left\{x \in \mathbb{R}^4 : 0 \le x \le e, 4x_1 + 3x_2 + 2x_3 + x_4 \le 4,\right. \\ &\qquad \left. x_1 + x_2 + x_3 \le 1, x_1 + x_4 \le 1\right\}. \end{aligned}$$

Which is "best"?

Answer: show that $P_3 \subseteq P_2 \subseteq P_1$.

2.4 Is there an ideal formulation? Yes.

Let X be the set of feasible solutions of some IP (or MIP). There are infinitely many alternative formulations for X!

Definition 2.4 (Convex hull) *Given a finite set* $X = \{x^{(1)}, \ldots, x^{(t)}\} \subset \mathbb{R}^n$, *the* **convex hull** *of* X , *written* $\text{conv}(X)$, *is defined as*

$$\text{conv}(X) = \left\{ x : x = \sum_{i=1}^{t} \lambda_i x^{(i)}, \, s.t. \sum_{i=1}^{t} \lambda_i = 1, \lambda_i \geq 0 \right\}.$$

The geometric interpretation of the **convex hull** $\text{conv}(X)$ of a set of vectors X is that if $\mathbf{z_1}$ and $\mathbf{z_2}$ are points in $\text{conv}(X)$ then every point on the line section joining $\mathbf{z_1}$ and $\mathbf{z_2}$ is also.

Exercise 2.2 **Check** *this claim (all points* $\mathbf{z}$ *on the line connecting* $\mathbf{z_1}$ *and* $\mathbf{z_2}$ *can be written* $\mathbf{z} = \alpha \mathbf{z_1} + (1 - \alpha)\mathbf{z_2}$, $0 \leq \alpha \leq 1$*).*

So the convex hull of a set of points contains all intermediate points — draw a sketch.

Theorem 2.1 conv(X) *is a polyhedron if the set* X *is finite, or if it is the set of feasible solutions of some MIP (not true for an arbitrary set X !).*

If X is the set of feasible solutions of some IP or MIP, then the polyhedron conv(X) is the ideal formulation for X.

Why is $\text{conv}(X)$ **the ideal formulation?**

- Suppose that $P \subset \mathbb{R}^n$ is a formulation for $X \subset \mathbb{Z}^n$.
- To solve the IP $\max\{c^T x : x \in X\} = \max\{c^T x : x \in P \cap \mathbb{Z}^n\}$, I will see later that I often begin by solving the **linear relaxation** of this problem, the LP $\max\{c^T x : x \in P\}$, (LPR) say.
- The solution x^{LP} (say) will be found at an extreme point (vertex) of P and if $x^{LP} \in X$ then I have found the optimal solution of (IP) just by solving (LPR).

Definition 2.5 (Extreme Point) *A point* v *in a set* P *is an extreme point of* P *if it does not lie on any line section joining two points in* P. *(Draw a sketch!)*

Theorem 2.2 *All extreme points of* $\text{conv}(X)$ *lie in* X .

Proof: Exercise! Or see App. A.6.

2.5 When is one formulation better than another?

- What makes the polyhedron conv(X) the "ideal" formulation is the fact that for any other formulation P for X ,
  $X \subseteq \text{conv}(X) \subseteq P$ — i.e. conv(X) is the **smallest** formulation for X.
- So just calculate conv(X) and I am done?
- Unfortunately, calculating conv(X) (finding the huge set of linear inequality constraints that define the convex hull) is just as hard a task as solving the IP/MIP directly by one of the methods to be discussed below.
- But the result has considerable technical importance.

Define what I mean by one formulation being "better" than another.

Definition 2.6 (Better formulation) *Given a set* $X \in \mathbb{R}^n$ *and two formulations,* P_1 *and* P_2 *for* X*; I say that* P_1 *is a* **better formulation** *than* P_2 *if* $P_1 \subset P_2$ *(*P_1 *is a strict subset of* P_2*).*

(Note: formulations cannot always be compared.)

For the formulations of the knapsack above, $P_3 \subset P_2 \subset P_1$ and in fact, $P_3 = \text{conv}(X)$.

Exercise 2.3 *Check.*

2.6 Exercises

1. Show that the three sets
   $X_1 = \{x \in \mathbb{B}^4 : 97x_1 + 32x_2 + 25x_3 + 20x_4 \leq 139\}$,
   $X_2 = \{x \in \mathbb{B}^4 : 2x_1 + x_2 + x_3 + x_4 \leq 3\}$ and
   $X_3 = \{x \in \mathbb{B}^4 : x_1+x_2+x_3 \leq 2;\ x_1+x_2+x_4 \leq 2;\ x_1+x_3+x_4 \leq 2\}$,
   are equal to a set X of elements $\mathbb{B}^4$.

2. List the elements of X.

3. Now show that when X_1, X_2, X_3 are rewritten as formulations with $\mathbb{B}^4$ replaced by $x \in \mathbb{R}$ with $0 \leq x_i \leq 1$ that $X_3 \subseteq X_2 \subseteq X_1$.

4. Which formulation is best?

5. Is the best formulation equal to the convex hull of X?

3 Relaxations and Bounds

Given an IP $z^* = \max\{c^T x : x \in X\}$ one strategy is to "zoom in" on the solution by continually updating upper & lower bounds on z^*.

How do I find z^* or prove that a feasible solution x is optimal or close to optimal? (I'll use z^* to signify the optimal — max/min — solution of an IP.)

- Search for an lower and upper bound on the optimal objective value: $\underline{z} \leq z^* \leq \bar{z}$.
- If $\underline{z} = \bar{z}$, then the optimal objective value z^* has been found.
- If $x \in X$ satisfies $\underline{z} \leq c^T x \leq \bar{z}$ and $\bar{z} - \underline{z} \leq \varepsilon$, where $\varepsilon > 0$ is an "optimality tolerance" parameter, then obviously x satisfies $c^T x \geq z^* - \varepsilon$.

Primal Bounds

- Primal bounds are **lower**/**upper** bounds on the optimal objective function value of a **max**/**min** problem.
- Any feasible solution gives a primal bound — can you explain why?
- For example, for the TSP any permutation of the n cities gives a tour and the length of the tour gives a primal (upper) bound on the minimum tour length z^*.

Dual bounds

- Finding upper bounds for a maximisation problem (lower bounds for a minimisation problem) is the "dual" problem.
- The word "dual" is related to the concept of duality from Linear Programming — more later.
- **Upper**/**lower** bounds bounds on the optimal objective function value of a **max**/**min** problem are often obtained via **relaxations** of the problem (IP):

$$z^* = \max\{c^T x : x \in X \subseteq \mathbb{R}^n\}. \quad \text{(IP)}$$

- The idea of relaxation is to replace a "hard" **max**/**min** IP by an easier "relaxed" problem whose optimal value is **greater than or equal to**/**less than or equal to** the optimal value z^*.

For the relaxed problem to have this property, there are two obvious approaches:

1. Widen the set of feasible solutions so that I am optimising over a bigger set, or
2. Replace the objective function to be **maximised**/**minimised** by a function that has the same or a **larger**/**smaller** value everywhere.

Definition 3.1 (Relaxation) *A problem (RP)* $z^R = \max\{f(x) : x \in \tilde{X} \subseteq \mathbb{R}^n\}$ *is a* **relaxation** *of (IP) if:*

- $X \subseteq \tilde{X}$ *and*
- $f(x) \geq c^T x$ *for all* $x \in X$.

Theorem 3.1 *If (RP) is a relaxation of (IP), then* $z^{\mathsf{R}} \geq z^*$.

Proof: Exercise.

- The obvious question: what are the "interesting" relaxations?
- One of the most useful (and natural) is the Linear Programming (LP) relaxation.

3.1 Linear Programming Relaxations

Definition 3.2 (LP relaxation) *For the integer program* $\max\{c^T x : x \in P \cap \mathbb{Z}^n\}$ *with formulation* $P = \{x \in \mathbb{R}^{n+} : Ax \leq b\}$, *the linear programming relaxation is the linear program* $z^{LP} = \max\{c^T x : x \in P\}$.

This is clearly a relaxation as $P \cap \mathbb{Z}^n \subseteq P$ and the objective function is unchanged.

Example 3.1 *Consider the IP*

$$\begin{aligned} z^* = \max\, & 4x_1 - x_2 & & \\ & 7x_1 - 2x_2 & \leq & \ 14 \\ & x_2 & \leq & \ 3 \\ & 2x_1 - 2x_2 & \leq & \ 3 \\ x_1, x_2 \geq 0, & x_1, x_2 \in \mathbb{Z} & & \end{aligned}$$

- *To get a primal (lower) bound, notice that* $(2,1)$ *is a feasible point so I have the lower bound* $z^* \geq 7$.
- *To get a dual (upper) bound I look at the LP relaxation (LPR).*
  - *The optimal soln to LPR is* $x^* = (20/7, 3)$ *with* $z^{LP} = 59/7$.
  - *So I have the upper (dual) bound* $z^* \leq 59/7$.
  - *As the optimal value must be integral I can round down to the nearest integer and find that* $z^* \leq 8$.
- *Even though I haven't solved the original IP I have placed very tight bounds on the optimal* z^*.
- *In fact* $(2,1)$ *is the optimal solution.*
- *I could have taken* $(1,0)$ *as my randomly chosen integer feasible point giving the weaker lower bound* $z^* \geq 4$.

- The definition of better formulations from Ch. 1 is closely related to the question of how to find better relaxations.
- In particular, better formulations give sharper dual bounds (as the feasible region is smaller so the value of z^{LP} will be less).
- I formalise this as a Theorem.

Theorem 3.2 *Suppose that* P_1, P_2 *are two formulations for the IP* $\max\{c^T x : x \in X \subseteq \mathbb{Z}^n\}$ *with* P_1 *a better formulation than* P_2*, i.e.* $P_1 \subset P_2$ *. If* $z_i^{LP} = \max\{c^T x : x \in P_i\}$ *for* $i = 1, 2$ *are the values of the associated LP relaxations, then* $z_1^{LP} \leq z_2^{LP}$*.*

Proof: Exercise.

Relaxations don't just give dual bounds, they sometimes allow us to prove optimality.

Theorem 3.3 *Given an IP;*

1. *If a relaxation (RP) is infeasible, the original problem (IP) is infeasible.*
2. *Let* x^R *be an optimal solution of (RP). If* $x^R \in X$ *and* $f(x^R) = c^T x^R$, *then* x^R *is an optimal solution of (IP).*

Proof:

1. As RP is infeasible, the relaxed feasible region $\tilde{X}$ is empty and so $X = \varnothing$ as $X \subset \tilde{X}$.
2. As $x^R \in X$, the optimal solution x^* of the IP satisfies $z^* \geq c^T x^R = z^R$. But as $z^* \leq z^R$ I have $c^T x^R = z^* = z^R$. ■

Exercise 3.1 *Can you explain this in your own words?*

- In practice, the Theorem applies when I find an optimal solution x^* to an LP relaxation of an IP.
- If x^* is itself integer then I can conclude that x^* is an optimal solution to the IP.

Example 3.2 *The LP relaxation of the IP*

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

has optimal solution $x^R = (1,1,0,0)$. *As* x^R *is integral, it satisfies the IP.*

Conclude that x^R *is an optimal solution of the IP.*

3.2 Combinatorial Relaxations–Skip

- Whenever the relaxed problem is a **combinatorial optimisation** problem (Definition 1.6) I can use the term **combinatorial relaxation**.
- In many cases (such the first Example below) the relaxation is an easy problem that can be solved quickly.

Some examples:

1. **The TSP**. Remember the discussion of the TSP starting on Slide 44. One way to describe a TSP with distance matrix c_{ij} is that a TSP tour is an assignment containing no subtours; so (the sum is over the elements of the $n \times n$ 0/1 matrix x)

$$z^{TSP} = \min_{x \in \mathbb{B}^{n \times n}} \left\{ \sum_{i,j,i \neq j} c_{ij} x_{ij} : x \text{ forms a tour} \right\} \geq$$

$$\min_{x \in \mathbb{B}^{n \times n}} \left\{ \sum_{i,j,i \neq j} c_{ij} x_{ij} : x \text{ forms an assignment} \right\} = z^{ASS}.$$

This is a min problem so the relaxation gives a lower bound on z. Assignment problems are easier to solve than TSP's.

This is sometimes called an "assignment relaxation". (See Exercise 6 on Slide 201.)

2. **The Knapsack Problem**. A combinatorial relaxation of the set $X = \{x \in \mathbb{Z}^n_+ : \sum_{j=1}^n a_j x_j \leq b\}$ is the set

$$\tilde{X} = \{x \in \mathbb{Z}^n_+ : \sum_{j=1}^n \lfloor a_j \rfloor x_j \leq \lfloor b \rfloor\}$$

where $\lfloor a \rfloor$ is the largest integer less than or equal to a.

Can you see why?

Hint: show that $X \subseteq \tilde{X}$ by taking $x \in X$ and argue that as

- $\lfloor \sum_{j=1}^n a_j x_j \rfloor \leq \sum_{j=1}^n a_j x_j \leq b$ and that
- $\sum_{j=1}^n \lfloor a_j \rfloor x_j \leq \lfloor \sum_{j=1}^n a_j x_j \rfloor$ (remember that $a_j > 0$ for $j = 1, \ldots, n$)

I can conclude that as $\sum_{j=1}^n \lfloor a_j \rfloor x_j$ is an integer $\leq b$ I must have $\sum_{j=1}^n \lfloor a_j \rfloor x_j \leq \lfloor b \rfloor$.

Exercise 3.2 *Fill in the gaps in the argument.*

3.3 Lagrangian Relaxation–Skip

- Suppose that I am given an integer program (IP) $z^* = \max\{c^T x : Ax \leq b, x \in X \subseteq \mathbb{Z}^n\}$.
- If the IP is too hard (takes too much CPU) to solve directly, one possibility is to just drop the constraints $Ax \leq b$.
- The resulting problem $z' = \max\{c^T x : x \in X \subseteq \mathbb{Z}^n\}$ is certainly a relaxation of the IP.
- In the TSP mentioned previously, when I drop the subtour constraints I get an assignment problem.

- A clever extension of this idea (more later) is not to simply drop the so-called "complicating constraints" to get a relaxation but instead to move them from the list of constraints into the objective function with "Lagrange multipliers".
- The Lagrangian relaxation of (IP) is written:
  $z(u) = \max\{c^T x + u^T(b - Ax) : x \in X\}$, for any vector $u \geq 0$.
- The Lagrangian relaxation of (IP) is a relaxation as the feasible region is larger and for any x feasible for (IP),
  $z(u) \geq c^T x + u^T(b - Ax) \geq c^T x$.

I can state a useful Theorem that gives us an upper (dual) bound for z^*: (z^* is the optimal objective value for the IP.)

Theorem 3.4 *Let* $z(u) = \max\{c^T x + u^T(b - Ax) : x \in X\}$. *Then* $z^* \leq z(u)$ *for all* $u \geq 0$.

Proof: Let x^* be the optimal solution of the IP. As x^* is feasible for IP, $x^* \in X$. Also I have that $Ax^* \leq b$ and so as $u \geq 0$, $c^T x^* \leq c^T x^* + u^T(b - Ax^*) \leq z(u)$. ■

So **any** Lagrangian relaxation (any choice of $u \geq 0$) of an IP gives an upper (dual) bound for x^* (for a max problem).

3.4 Duality

I have used the word dual/duality up to now in a general sense. For LP's, LP duality gives a natural way to find upper bounds. I will show that these ideas can be extended to IP's.

The standard Linear Programming definition;

Definition 3.3 (LP Duality) *Given an LP (the* **Primal** *problem) I can write a closely related LP, its* **Dual***:*

$$z = \max\{c^T x : Ax \leq b, x \in \mathbb{R}^n, x \geq 0\} \quad \textbf{Primal} \qquad (3.1)$$

$$w = \min\{b^T y : A^T y \geq c, y \in \mathbb{R}^m, y \geq 0\}. \quad \textbf{Dual} \qquad (3.2)$$

I can view this as a "recipe" for computing the Dual version of an LP.

Variant Forms for the Dual Pair Different authors often choose different forms for the Dual Pair — these are of course equivalent.

One example:

- Start with

$$\begin{aligned} z &= \max\{c^T x : Ax \le b, x \in \mathbb{R}^n, x \ge 0\} \quad \textbf{Primal} \\ w &= \min\{b^T y : A^T y \ge c, y \in \mathbb{R}^m, y \ge 0\}. \quad \textbf{Dual} \end{aligned}$$

- Make the changes $c \to -c$, $A \to -A$, $b \to -b$;

$$\begin{aligned} z &= -\min\{c^T x : Ax \ge b, x \in \mathbb{R}^n, x \ge 0\} \quad \textbf{Primal} \quad (3.3) \\ w &= -\max\{b^T y : A^T y \le c, y \in \mathbb{R}^m, y \ge 0\}. \quad \textbf{Dual} (3.4) \end{aligned}$$

(The minus signs on the min & max can be dropped — but not forgotten!)

One nice property of the "Primal to Dual" operation is that the Dual of the Dual problem is the (original) Primal problem.

Another and more important result is that:

Theorem 3.5 ((Strong) Duality Theorem) *The optimal solution to the Dual equals the optimal solution to the Primal (when both problems are feasible and bounded).*

In other words, if I solve the Primal LP (in one of the forms above) and solve the corresponding Dual LP the values of z and w are the same.

Any dual feasible point for an LP (max problem) provides an upper bound on the primal objective thanks to the Weak Duality Theorem:

Theorem 3.6 (Weak Duality) *Let a Primal LP and its Dual be stated as (the original form):*

$$\begin{aligned} z &= \max\{c^T x : Ax \le b, x \in \mathbb{R}^n, x \ge 0\} \quad \textbf{Primal} \\ w &= \min\{b^T y : A^T y \ge c, y \in \mathbb{R}^m, y \ge 0\}. \quad \textbf{Dual} \end{aligned}$$

then for **any** *primal feasible point* x *and* **any** *dual feasible point* y, $b^T y \ge c^T x$ *(that is, the dual objective is an upper bound on the primal objective) — a result known as weak duality.*

Proof: Let x satisfy $Ax \le b$ and let y satisfy $A^T y \ge c$, together with $x, y \ge 0$ then $b^T y \ge (Ax)^T y = x^T A^T y \ge x^T c$. ■

Exercise 3.3 *State and prove the Weak Duality Theorem when the Primal–Dual pair is given by Eqs.* (3.3) *and* (3.4).

Stopped here 14:00,Thursday Week 8

This suggests the following general definition:

Definition 3.4 *The two problems*

$$\begin{aligned} z^* &= \max\{c^T x : x \in X \subseteq \mathbb{Z}^n\} \quad \textit{(IP)} \quad \textit{and} \\ w^* &= \min\{\omega(u) : u \in U\} \quad \textit{(D)} \end{aligned}$$

form a (weak)-dual pair if $c^T x \leq \omega(u)$ *for all* $x \in X$ *and all* $u \in U$. *When* $z^* = w^*$ *, they form a strong dual pair.*

- The advantage of a dual problem (as opposed to a relaxation) is that **any** feasible solution of the dual provides a dual bound — while a relaxation needs to be solved to optimality.
- Can I find dual problems?
- An LP relaxation provides a weak dual:

Theorem 3.7 *The IP* $z = \max\{c^T x : Ax \leq b, x \in \mathbb{Z}^{n+}\}$ *and the dual LP* $w^{LP} = \min\{b^T u : A^T u \geq c, u \in \mathbb{R}^{m+}\}$ *form a weak dual pair.*

Proof: For **any** $u \in \mathbb{R}^{m+}$; $b^T u \geq w^{LP} \geq z^{LP} \geq z^*$. The first $\geq$ is by definition of w^{LP}, the second follows from the Weak Duality Theorem and the last holds as z^{LP} is a relaxation of the IP. ■

So **any** feasible point u for the Dual LP
$\min\{b^T u : A^T u \geq c, u \in \mathbb{R}^{m+}\}$ gives an upper bound for the optimal solution z^* of the original IP, namely $z^* \leq b^T u$.

Let's see an Example.

Example 3.3 (Weak Dual Upper Bound) *Consider the IP Ex 3.1*

$$\begin{aligned} z^* = \max\, & 4x_1 - x_2 & & \\ & 7x_1 - 2x_2 & \leq & \ 14 \\ & x_2 & \leq & \ 3 \\ & 2x_1 - 2x_2 & \leq & \ 3 \\ & x_1, x_2 \geq 0, x_1, x_2 \in \mathbb{Z}. & & \end{aligned}$$

The Dual of the LP relaxation is

$$\begin{aligned} w^* = \min\, & 14u_1 + 3u_2 + 3u_3 & & \\ & 7u_1 + 0u_2 + 2u_3 & \geq & \ 4 \\ & -2u_1 + u_2 - 2u_3 & \geq & \ -1 \\ & u_1, u_2, u_3 \geq 0, u_1, u_2, u_3 \in \mathbb{R}. & & \end{aligned}$$

- *I need to find* **any** *vector* u *that satisfies the constraints of the Dual LP above.*
- *The point* $u = (5/8, 1/4, 0)^T$ *is dual feasible (check) and gives an upper bound* $b^T u = 14 \cdot 5/8 + 3 \cdot 1/4 = 19/2$.
- *Not as good as the upper bound (8) found by solving the LP relaxation but here all we needed was a dual feasible point.*
- *Of course, finding a feasible point for a large LP is often nearly as difficult as actually solving the problem!*
- *And a bad choice (e.g.* $u = (1, 2, 0)^T$*) gives a very large upper bound (20).*

By analogy with Thm. 3.3, dual problems sometimes allow us to prove optimality.

Theorem 3.8 *Suppose that IP & D are a weak dual pair.*

1. *If D is unbounded then IP is infeasible.*
2. *If* $x^* \in X$ *and* $u^* \in U$ *satisfy* $c^T x^* = w(u^*)$ *then* x^* *is optimal for IP and* u^* *is optimal for D.*

Proof:

1. By definition of a weak dual pair, $w(u) \geq c^T x$, for all primal feasible x & dual feasible u. Assume that there is some fixed primal feasible x. Then as D is unbounded, I can find a sequence of dual feasible u_k s.t. $w(u_k) \downarrow -\infty$. So $w(u_k) \geq c^T x$ for all k is impossible. Contradiction. So the IP is infeasible. ■

2.
- From $w(u) \geq c^T x$ for all primal feasible x & dual feasible u, I have $\min_{u \in U} w(u) \geq c^T x$ for all primal feasible x and $w(u) \geq \max_{x \in X} c^T x$ for all dual feasible u.
- So $\min_{u \in U} w(u) \geq c^T x^*$ and $w(u^*) \geq \max_{x \in X} c^T x$.
- Combining, I have $\min_{u \in U} w(u) \geq c^T x^* = w(u^*) \geq \max_{x \in X} c^T x$.
- So x^* is feasible and $c^T x^* \geq \max_{x \in X} c^T x$ and therefore "attains the maximum'.
- Similarly u^* is feasible and $w(u^*) \leq \min_{u \in U} w(u)$ and therefore "attains the minimum".

■

3.5 Primal Bounds: Greedy & Local Search–Skip

How to obtain primal bounds? (Lower bounds on z^* for a max IP).

In other words how can I find **some** integer feasible point?

One approach is to use a **greedy heuristic** — i.e. to construct a solution starting from the empty set, choosing at each step the element that gives the largest possible improvement.

Two examples:

Example 3.4 (0–1 Knapsack problem)

$$\max 12x_1 + 8x_2 + 17x_3 + 11x_4 + 6x_5 + 2x_6 + 2x_7$$
$$4x_1 + 3x_2 + 7x_3 + 5x_4 + 3x_5 + 2x_6 + 3x_7 \leq 9$$
$$x \in \mathcal{B}^7$$

The variables are deliberately ordered (re-ordered if necessary) so that the "payoff/cost" ratio is decreasing: $\frac{c_j}{a_j} \geq \frac{c_{j-1}}{a_{j-1}}$ *for* $j = 1, \ldots, n-1$.

I can easily generate a greedy solution as follows:

(i) *As* c_1/a_1 *is maximal and there is "room in the knapsack" (9 units); fix* $x_1 = 1$.

(ii) *Now, considering the remaining variables, as* c_2/a_2 *is largest and there is room left* $(9 - 4 = 5)$*; fix* $x_2 = 1$.

(iii) *As each item* $3, 4, 5$ *in that order requires more space that the* $5 - 3 = 2$ *units available, set* $x_3 = x_4 = x_5 = 0$.

(iv) *As* c_6/a_6 *is maximal (over the remaining variables* x_6–x_7*), set* $x_6 = 1$.

(v) *Finally, set* $x_7 = 0$ *as no further space is available.*

So the greedy solution is $x^G = (1, 1, 0, 0, 0, 1, 0)$ *with objective value* $z^G = c^T x^G = 22$.

Example 3.5 (Symmetric Travelling Salesman Problem)

Consider an "instance" of this problem with distance matrix:

$$C_e = \begin{bmatrix} * & 9 & 2 & 8 & 12 & 11 \\ & * & 7 & 19 & 10 & 32 \\ & & * & 29 & 18 & 6 \\ & & & * & 24 & 3 \\ & & & & * & 19 \\ & & & & & * \end{bmatrix}$$

The Greedy heuristic examines the links in order of non-decreasing length.

- *The "cheapest" link is* $(1,3)$ *with* $c_{1,3} = 2$. *Select the link by setting* $x_{1,3} = 1$.
- *The next cheapest remaining link is* $(4,6)$ *with* $c_{4,6} = 3$. *As links* $(1,3)$ *and* $(4,6)$ *can appear together in a tour, set* $x_{4,6} = 1$.
- *Set* $x_{3,6} = 1$ *where* $c_{3,6} = 6$.
- *The next cheapest link is* $(2,3)$. *Set* $x_{2,3} = 0$ *as "city"* 3 *already has degree* 2 *(two arcs at the node) so all three links* $(1,3), (3,6)$ *and* $(2,3)$ *cannot be in the tour.*

- *Set* $x_{1,4} = 0$ *as link* $(1,4)$ *forms a subtour with the links already chosen.*
- *Set* $x_{1,2} = 1$ *where* $c_{1,2} = 9$.
- *Continue, choosing links* $(2,5)$ *with cost* 10 *and* $(4,5)$ *with cost* 24 *to complete the tour.*
- *All other* x_{ij} *are set to zero.*

The Greedy tour is $(1,3,6,4,5,2,1)$ *with length* $\sum_{i,j} c_{ij}x_{ij} = 54$.

Local Search

- Once an initial feasible solution (sometimes called the **incumbent**) has been found it is natural to try to improve it.
- One idea, a **local search** heuristic, tries to find the best solution in a "neighbourhood" of the incumbent.
- If it is better than the incumbent, it replaces it and the procedure is repeated.
- Otherwise the incumbent is "locally optimal" (wrt the chosen definition of neighbourhood) and the heuristic stops.

Again, an example.

Example 3.6 (Uncapacitated Facility Location) *Consider an instance with* $m = 6$ *clients and* $n = 4$ *depots and costs given by:*

$$C_{ij} = \begin{bmatrix} 6 & 2 & 3 & 4 \\ 1 & 9 & 4 & 11 \\ 15 & 2 & 6 & 3 \\ 9 & 11 & 4 & 8 \\ 7 & 23 & 2 & 9 \\ 4 & 3 & 1 & 5 \end{bmatrix} \quad \textit{and } f_j = (21, 16, 11, 24).$$

*It can be shown (***check** *) if* $N = \{1, 2, 3, 4\}$ *denotes the set of depots and* $S \subseteq N$ *denotes the set of open (in use) depots then the cost is:*

$$c(S) = \sum_{i=1}^{6} \min_{j \in S} c_{ij} + \sum_{j \in S} f_j.$$

For example, if $S^0 = \{1, 2\}$ *is the initial incumbent,*
$c(S^0) = (2 + 1 + 2 + 9 + 7 + 3) + 21 + 16 = 61.$

- *I need to define a neighbourhood of an incumbent* S*.*
- *One possibility is to consider as neighbours all sets obtained from* S *by the addition or removal of a single element:*

$$Q(S) = \{T \subseteq N \text{ such that } T = S \cup \{j\} \text{ for } j \notin S \text{ or } T = S \backslash \{i\} \text{ for } i \in S\}.$$

- *So* $Q(S_0) = \{\{1\},\{2\},\{1,2,3\},\{1,2,4\}\}$ *with costs* $c(\{1\}) = 63$, $c(\{2\}) = 66$, $c(\{1,2,3\}) = 60$, $c(\{1,2,4\}) = 84$.
- *The new incumbent is* $S^1 = \{1,2,3\}$ *with* $c(S^1) = 60$ *and* $Q(S^1) = \{\{1,2\},\{1,3\},\{2,3\},\{1,2,3,4\}\}$.
- *The new incumbent is* $S^2 = \{2,3\}$ *with* $c(S^2) = 42$ *and* $Q(S^2) = \{\{2\},\{3\},\{1,2,3\},\{2,3,4\}\}$.
- *The new incumbent is* $S^3 = \{3\}$ *with* $c(S^3) = 31$ *and* $Q(S^3) = \{\varnothing,\{1,3\},\{2,3\},\{3,4\}\}$.
- *There is no improvement so* $S^3 = \{3\}$ *is a locally optimal solution.*

3.6 Exercises

1. Find primal and dual bounds for the integer knapsack problem:

$$\begin{aligned} z = \max\, & 42x_1 + 26x_2 + 35x_3 + 71x_4 + 53x_5 \\ & 14x_1 + 10x_2 + 12x_3 + 25x_4 + 20x_5 \leq 69 \\ & x \in \mathbb{Z}^{5+}. \end{aligned}$$

2. Consider the 0–1 IP

$$(P_1) \quad \max \left\{ c^T x : \sum_{j=1}^{n} a_{ij} x_j = b_i, \text{ for } i = 1, \ldots, m, x \in \mathcal{B}^m \right\}.$$

and the 0–1 Equality Knapsack problem

$$(P_2) \quad \max \left\{ c^T x : \sum_{j=1}^{n} \left(\sum_{i=1}^{m} u_i a_{ij} \right) x_j = \sum_{i=1}^{m} u_i b_i, \text{ for } i = 1, \ldots, m, x \in \mathcal{B}^m \right\}.$$

where $u \in \mathbb{R}^m$.

Show that P_2 is a relaxation of P_1.

3. Consider the Equality Integer Knapsack problem:

$$(P_1) \quad \min \left\{ \sum_{j=1}^{5} c_j x_j : \frac{7}{4}x_1 - \frac{2}{3}x_2 + \frac{5}{2}x_3 - \frac{5}{12}x_4 + \frac{19}{6}x_5 = \frac{8}{3}, x \in \mathbb{Z}^{5+} \right\}$$

(a) Show that the problem

$$(P_2) \quad \min \left\{ \sum_{j=1}^{5} c_j x_j : \frac{3}{4}x_1 + \frac{1}{3}x_2 + \frac{1}{2}x_3 + \frac{7}{12}x_4 + \frac{1}{6}x_5 = \frac{2}{3} + w, x \in \mathbb{Z}^{5+}, w \in \mathbb{Z}^{+} \right\}$$

is a relaxation of P_1.

(b) Show that the problem

$$(P_3) \quad \min \left\{ \sum_{j=1}^{5} c_j x_j : \frac{3}{4}x_1 + \frac{1}{3}x_2 + \frac{1}{2}x_3 + \frac{7}{12}x_4 + \frac{1}{6}x_5 \geq \frac{2}{3}, x \in \mathbb{R}^{5+} \right\}$$

is a relaxation of P_2.

4. Apply a greedy heuristic to the instance of the UFL problem in Ex. 3.6 above.

4 IP Solution Techniques

4.1 Divide and conquer approach

Consider the generic problem: $z = \max\{c^T x : x \in S\}$.

How can I

- Break the problem into a series of smaller subproblems that are easier to solve;
- Solve the sub-problems;
- Re-assemble the sub-problems to solve the original problem?

A simple observation: (Obvious?)

Theorem 4.1 *Let* $S = S_1 \cup \cdots \cup S_N$ *be a decomposition of the set* S *and let* $z^k = \max\{c^T x : x \in S_k\}$ *for* $k = 1, ..., N$. *Then* $z = \max_k z^k$.

"The oldest/tallest person in the room is the oldest/tallest of the oldest/tallest people in each row".

- I'll refer to the problems $\max\{c^T x : x \in S_k\}$ as subproblems.
- The solution to the original problem can be found by solving subproblems $1, ..., N$ and taking the best solution found.
- If any of the subproblems is still too hard to be solved directly, the set S_k can be further decomposed, etc., thus forming a "tree" of subproblems.
- Taken to the extreme, this leads to complete enumeration which is a Bad Thing ©.

Enumeration Trees A typical way to represent a divide & conquer approach is via an **enumeration tree**. For example if $S \subseteq \{0,1\}^3$ I can construct the enumeration tree in the Figure.

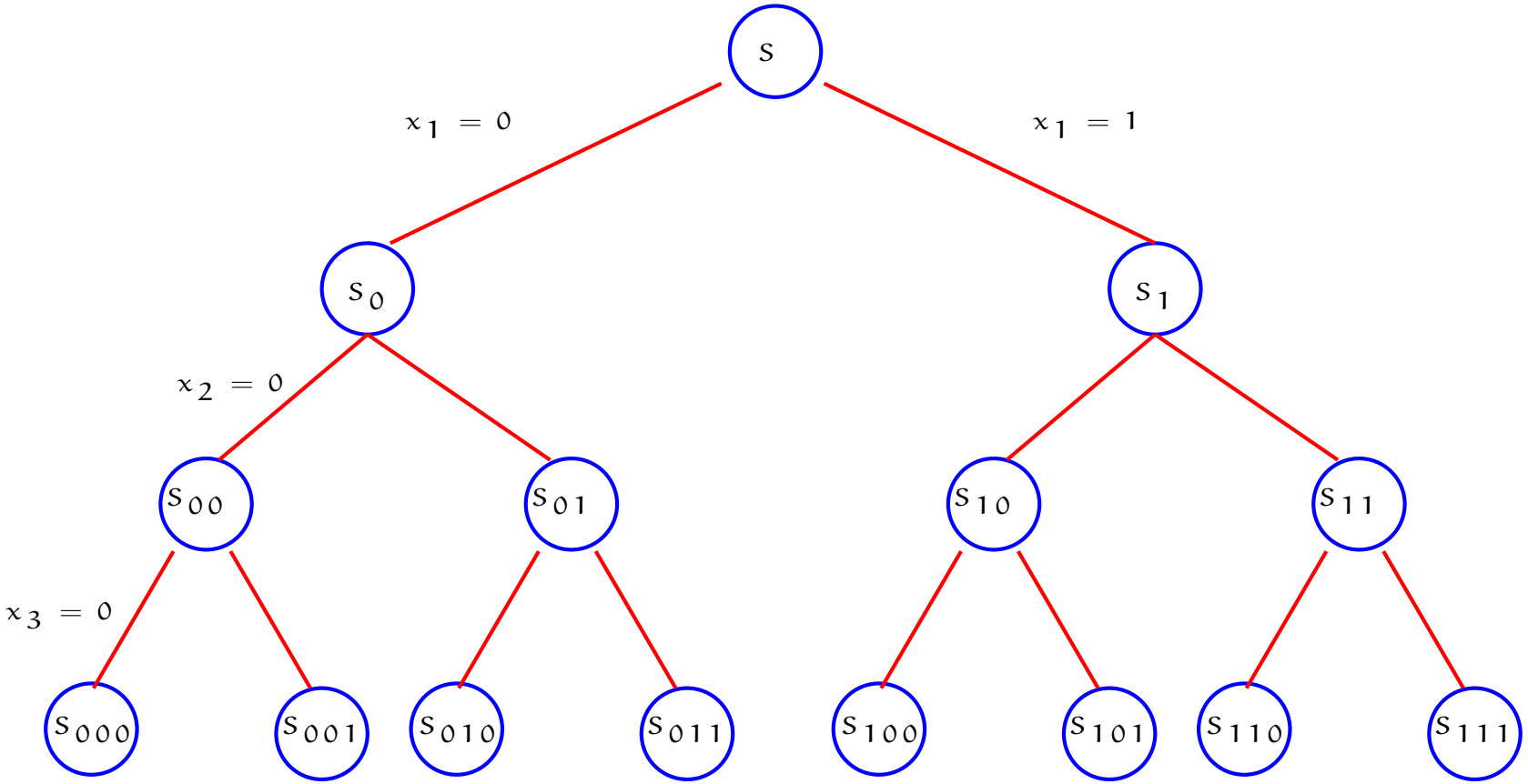

Figure 3: Binary Enumeration Tree

Here I first divide (decompose) S into

$S_0 = \{x \in S : x_1 = 0\}$ and $S_1 = \{x \in S : x_1 = 1\}$, then
$S_{00} = \{x \in S_0 : x_2 = 0\} \equiv \{x \in S : x_1 = x_2 = 0\}$ and
$S_{01} = \{x \in S_0 : x_2 = 1\}$, ... and so on ....

Note that a "leaf" of the tree $S_{i_1 i_2 i_3}$ is non-empty if and only if $x = (i_1, i_2, i_3)$ is in S (is feasible).

The leaves of the tree in the Figure correspond precisely to the points of $\mathbb{B}^3$ to be examined if complete enumeration was carried out.

(The tree is drawn upside down with the root at the top.)

For another example take the enumeration of all the tours of the TSP on 4 "cities" (i.e. all permutations of $1, 2, 3, 4$).

- Define S to be the set of all tours on the 4 cities.
- First divide S into $S_{(12)}$, $S_{(13)}$, $S_{(14)}$ where $S_{(ij)}$ is the set of all tours containing the arc (link) (ij).
- Then $S_{(12)}$ is divided again into $S_{(12)(23)}$ and $S_{(12)(24)}$ and so on.
- Note that at the top level of the tree I have chosen to branch on the arcs leaving node 1 and at the second level on the arcs leaving node 2 that do not immediately create a sub-tour with the previous branching arc.
- The resulting tree is shown below.
- (This is an example of multiple rather than binary branching as each set can be divided into more than two parts.)

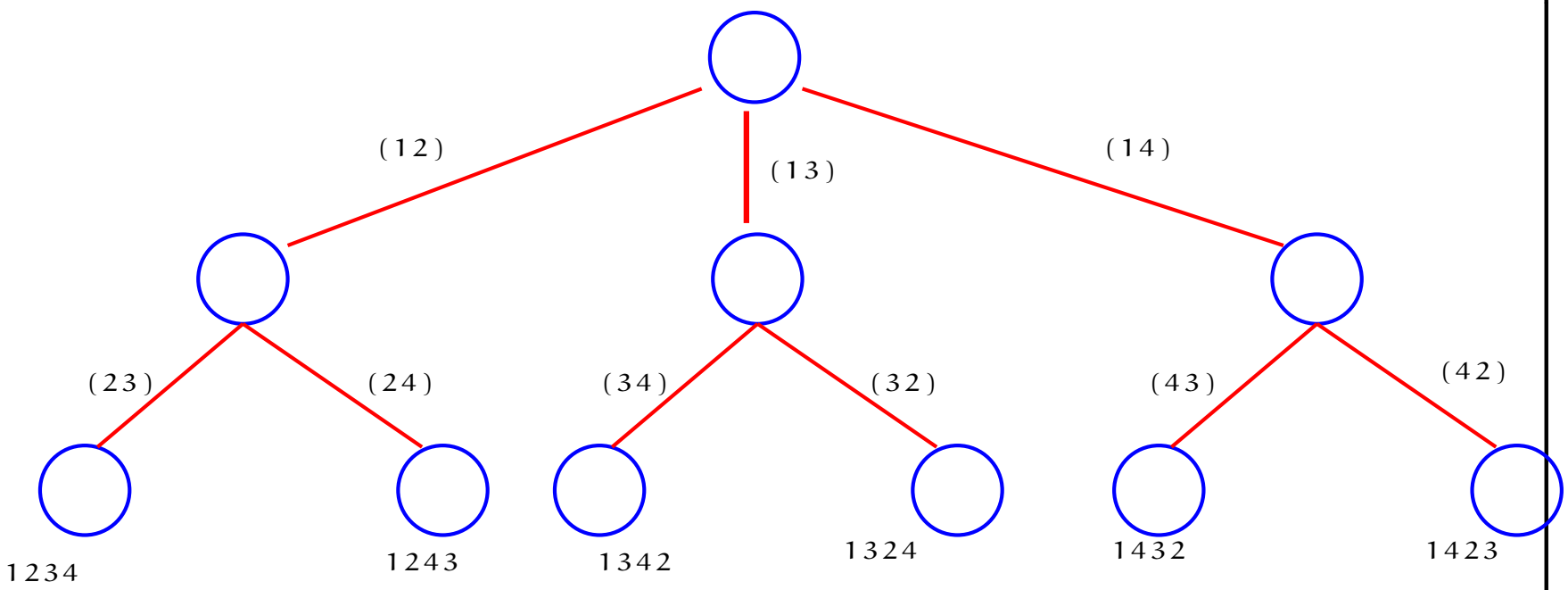

Figure 4: TSP Enumeration Tree

The six leaves of the tree correspond to the $(n-1)!$ tours shown, where $i_1 i_2 i_3 i_4$ means that the cities are visited in the sequence i_1, i_2, i_3, i_4.

Example 4.1 $S = \{x \in \{0,1\}^3 \; s.t. : 61x_1 + 49x_2 + 20x_3 \leq 100\}$. *I can use the same tree as in Fig. 4 to search for leaves that satisfy the constraint.*

- $S = S_0 \cup S_1$, *where* $S_0 = \{x \in S : x_1 = 0\}$ *and* $S_1 = \{x \in S : x_1 = 1\}$.
- $S_0 = S_{00} \cup S_{01}$, *where* $S_{00} = \{x \in S : x_1 = 0, x_2 = 0\}$, *etc.*
  - $S_{00} = S_{000} \cup S_{001}$
  - $S_{01} = S_{010} \cup S_{011}$
- $S_1 = S_{10} \cup S_{11}$
  - $S_{10} = S_{100} \cup S_{101}$
  - $S_{11} = S_{110} \cup S_{111}$

So, can you find the feasible leaves?

4.2 Implicit Enumeration

- I know that complete enumeration (of all possible solutions) is completely impractical for all but the smallest IP's.
- So simply "dividing and conquering" indefinitely is not a viable strategy.
- How can I use bounds on the values of $z^k = \max\{c^T x : x \in S_k\}$ for $k = 1, ..., N$ to "prune" the tree? (The word "fathom" is often used instead of "prune".)
- First; how can I assemble bound information?

Theorem 4.2 *Let* $S = S_1 ... S_N$ *be a decomposition of* S *into smaller sets and let* $z_k = \max\{c^T x : x \in S_k\}$ *for* $k = 1, ..., N$. *Let* $z = \max c^T x$ *on* S.

For each k, *let* $\bar{z}_k$ *be an upper bound on* z_k *and* $\underline{z}_k$ *be a lower bound on* z_k . *Then* $\bar{z} = \max_k \bar{z}_k$ *is an upper bound on* z *and* $\underline{z} = \max_k \underline{z}_k$ *is a lower bound on* z.

Proof: Obvious? ■

I now look at three Examples that will motivate the Branch & Bound Method.

Example 4.2 (Pruned/Fathomed By Optimality) *In the Figure I show a decomposition of a problem set* S *into two sets* S_1 *and* S_2 *with upper and lower bounds on each sub-problem.*

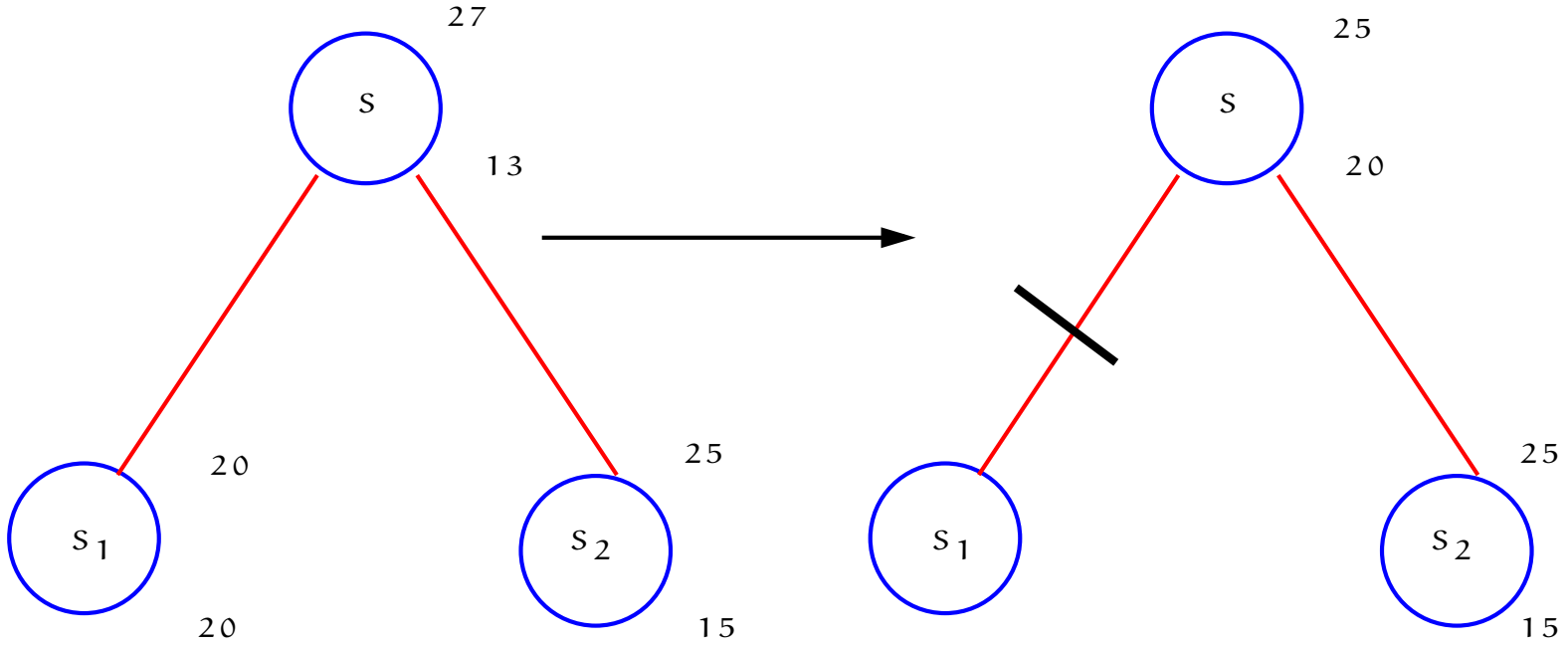

Figure 5: Pruned/Fathomed by Optimality

- *First notice that* $\bar{z} = \max_k \bar{z}^k = \max\{20, 25\} = 25$ *and* $\underline{z} = \max_k \underline{z}^k = \max\{20, 15\} = 20$.
- *Now observe that as the upper & lower bounds on* z_1 *are equal, I have* $z_1 = 20$.
- *So no reason to examine the set* S_1 *further as I have found the best (optimal) solution in* S_1.
- *So the branch* S_1 *of the enumeration tree can be* **pruned/fathomed by optimality**.

Example 4.3 (Pruned/Fathomed by Bound) *In the Figure I again decompose a problem set* S *into two sets* S_1 *and* S_2 *with upper and lower bounds on each sub-problem.*

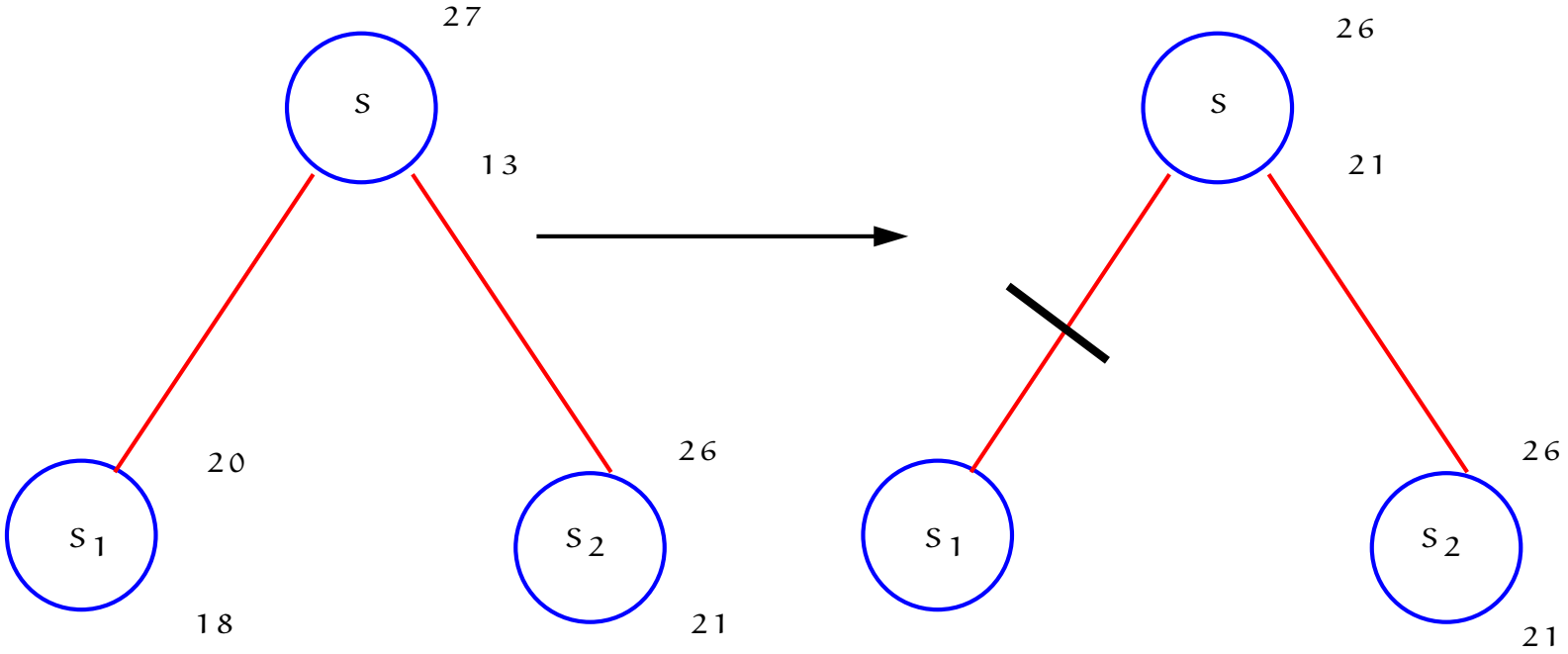

Figure 6: Pruned/Fathomed by Bound

- I note that $\bar{z} = \max_k \bar{z}^k = \max\{20, 26\} = 26$ and $\underline{z} = \max_k \underline{z}^k = \max\{18, 21\} = 21$.
- Now observe that as the optimal value is at least 21and $\bar{z}^1 = 20$, no optimal solution can lie in the set S_1.
- Therefore the branch S_1 of the enumeration tree can be **pruned/fathomed by bound**.

Example 4.4 (No Pruning Possible) *In the Figure I again decompose a problem set* S *into two sets* S_1 *and* S_2 *with upper and lower bounds on each sub-problem.*

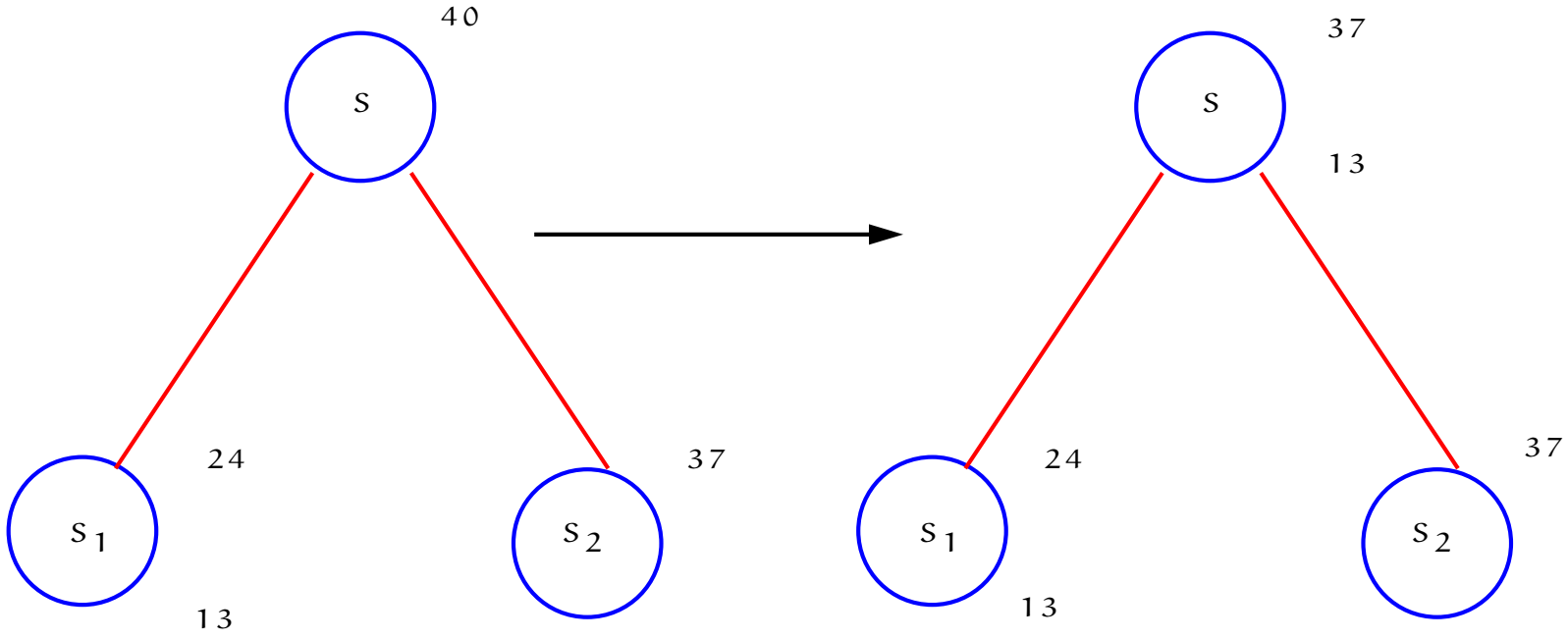

Figure 7: No Pruning Possible

- First notice that $\bar{z} = \max_k \bar{z}^k = \max\{24, 37\} = 37$ and $\underline{z} = \max_k \underline{z}^k = \max\{13, -\infty\} = 13$. (As there is no lower bound on $\underline{z}_2$ I set it to $-\infty$.)
- For this example I cannot draw any more conclusions and I need to explore the two sets S_1 and S_2 further.

Based on these three Examples I can list (at least) three situations that allow us to prune the tree.

1. Pruning by Optimality: $z^k = \{\max \mathbf{c}^T \mathbf{x} : \mathbf{x} \in S_k\}$ has been solved.
2. Pruning by Bound: $\bar{z}^k < \underline{z}$: no optimal solution can lie in the set S_k.
3. Pruning by Infeasibility: $S_k = \varnothing$.

How To Obtain Bounds On Sub-Problems? The answer (as previously) is the primal/dual dichotomy.

In other words (for a max IP)

- the primal (lower) bounds are obtained by finding particular feasible points
- the dual (upper) bounds are found by relaxation or duality.

It is clear how to build an implicit enumeration algorithm based on these ideas, at least in outline.

There are practical issues that need to be addressed before an "industrial strength" algorithm can be designed:

- What relaxation or dual problem should I use to generate upper bounds?
  - How should I choose between a weak bound that can be calculated quickly and a strict (tight) bound that takes almost as long to compute as the full problem?
- How should the feasible region S be partitioned into smaller regions $S_1 \cup \cdots \cup S_N = S$?
  - Should I divide into two/more parts?
  - Should I use an a priori rule for partitioning the set S.
  - Or should the divisions emerge as a consequence of earlier decisions?

- In what order should the sub-problems be examined?
  - Usually there is a list of "active" sub-problems not yet examined.
  - Should the next be chosen the basis of:
    - last-in-first-out?
    - best/largest upper bound first?
    - randomly?

These questions are important but I won't discuss them further here.

Now for an Example of a (reasonably) practical solution method.

4.3 Branch and Bound: A Worked Example

Here is a problem that I showed you previously as Example 3.1:

Example 4.5 (Branch and Bound)

$$\begin{aligned} z = \max\, & 4x_1 - x_2 \\ & 7x_1 - 2x_2 \leq 14 \\ & x_2 \leq 3 \\ & 2x_1 - 2x_2 \leq 3 \\ & x \in \mathbb{Z}^{2+} \end{aligned}$$

The solution method is given in complete detail in the following.

Once the method is familiar the process is faster!

Bounding

- To get a first dual (upper) bound, we solve the LP relaxation (integrality requirement dropped).
- Writing the problem as a min problem ($c \to -c$) I have the

initial tableau:

	x_1	x_2	x_3	x_4	x_5
0	−4	1	0	0	0
14	7	−2	1	0	0
3	0	1	0	1	0
3	2	−2	0	0	1

(**Note**: check carefully whether you are solving a max or a min problem. The tableau version Alg. A.1 of the Simplex Method for LP's is designed for min problems.)

- After 3 iterations of Simplex, the optimal tableau is:

	x_1	x_2	x_3	x_4	x_5
59/7	0	0	4/7	1/7	0
20/7	1	0	1/7	2/7	0
3	0	1	0	1	0
23/7	0	0	−2/7	10/7	1

LP(S)

- **N.B.** Reading from the tableau I have $z = -59/7$ but the original problem was a max one so I must flip the sign of the optimal z!!!
- I find a non-integer solution $(\bar{x}_1, \bar{x}_2) = (20/7, 3)$ and so $\bar{z} = 59/7$.
- **Any** integer solution will give us a primal (lower) bound $\underline{z}$.
  - Unfortunately I have no obvious way to find one.
  - The convention in this situation is to set $\underline{z} = -\infty$.

Branching

- Now because $\underline{z} < \bar{z}$ I need to divide (branch) the feasible region.
- How?
- One simple idea:
  - Choose an integer variable that is basic and fractional in the LP solution.
  - Split the problem in two about this fractional value.
  - If $x_j = \bar{x}_j \notin \mathbb{Z}$ I can take:

$$\begin{aligned} S_1 &= S \cap \{x : x_j \leq \lfloor \bar{x}_j \rfloor\} \\ S_2 &= S \cap \{x : x_j \geq \lceil \bar{x}_j \rceil\} \end{aligned}$$

- Obviously $S = S_1 \cup S_2$ and $S_1 \cap S_2 = \varnothing$.
- Another reason for this choice is that the solution $\bar{x}$ of LP(S) is not feasible for either LP(S_1) or LP(S_2).
- So provided the original problem is not degenerate, $\max\{\bar{z}_1, \bar{z}_2\} < \bar{z}$ so the upper bound will strictly decrease. (Remember that the original problem is a max problem.)
- Pursuing this idea, as $\bar{x}_1 = 20/7 \notin \mathbb{Z}$, I take $S_1 = S \cap \{x : x_1 \leq 2\}$ and $S_2 = S \cap \{x : x_1 \geq 3\}$.
- I now have the tree shown in Fig. 8 on the next slide.

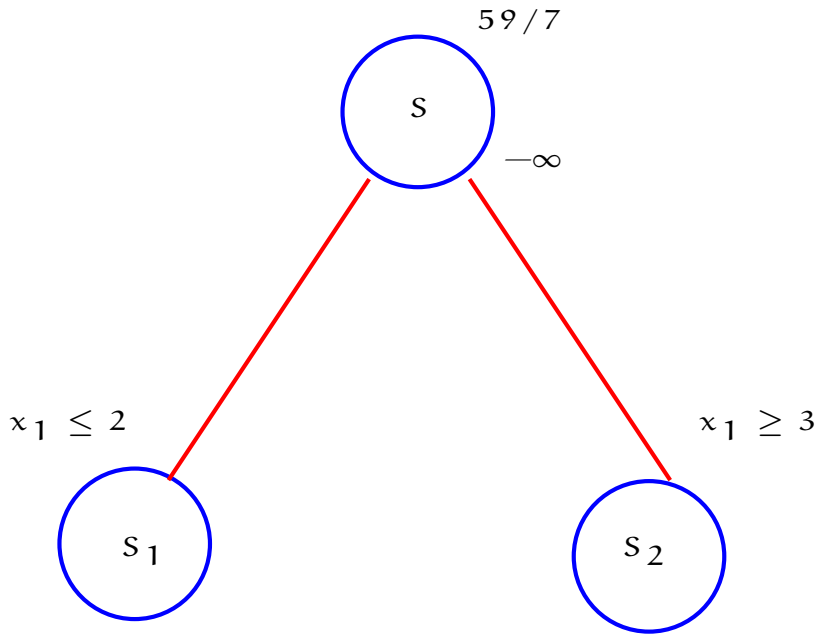

Figure 8: Partial B&B Tree # 1

Choosing a node

- The subproblems (nodes) that must still be examined are called **active**.
- The list of active sub-problems (nodes) to be examined now consists of S_1 and S_2.
- I arbitrarily choose S_1.

Re-optimising

- How should I solve the new modified LP's, $\mathsf{LP}(S_1)$ and $\mathsf{LP}(S_2)$?
- Re-solving from scratch seems inefficient — and is, for large problems at least.
- I need the **Dual Simplex Method** (see Appendix A.5 for details):
- Remember that

$$\begin{aligned} z^* &= \max\{c^T x : Ax \leq b, x \in \mathbb{R}^n, x \geq 0\} \quad \textbf{Primal} \\ w^* &= \min\{b^T y : A^T y \geq c, y \in \mathbb{R}^m, y \geq 0\}. \quad \textbf{Dual} \end{aligned}$$

- And $z^* = w^*$ (the Duality Theorem).

- Adding an extra constraint means adding an extra row to both A & b changing them to $\tilde{A}$ and $\tilde{b}$ say.
- The primal optimal solution x^* is no longer feasible.
- BUT if I augment y^* by adding an extra zero element, the **dual** constraints $\tilde{A}^T\tilde{y} \geq c, \tilde{y} \geq 0$ still hold!
- So the optimal soln to the primal provides a dual feasible soln to the perturbed primal problem generated by adding an extra constraint!
- More importantly the dual feasible point y^* is very close to dual optimality and usually it only takes a small number of Simplex iterations to converge to the Dual optimal point for the perturbed problem.

- Once the Dual Problem is solved the corresponding Primal optimal solution may be recovered (the decision variables for the primal are the multipliers/shadow prices for the Dual and vice versa).

- For the present Example, $A = \begin{bmatrix} 7 & -2 \\ 0 & 1 \\ 2 & -2 \end{bmatrix}$, $b = \begin{bmatrix} 14 \\ 3 \\ 3 \end{bmatrix}$, $c = \begin{bmatrix} -4 \\ 1 \end{bmatrix}$.

- Adding the extra constraint $x_1 \leq 2$ changes A to $\tilde{A} = \begin{bmatrix} 7 & -2 \\ 0 & 1 \\ 2 & -2 \\ 1 & 0 \end{bmatrix}$

  and b to $\tilde{b} = \begin{bmatrix} 14 \\ 3 \\ 3 \\ 2 \end{bmatrix}$

- How to solve the new LP?
- Simply use http://jkcray.maths.ul.ie/ms4315/Pivot.m or http://jkcray.maths.ul.ie/ms4315/SM.m (if you are too lazy to work out what row & column to pivot on).
- Or just hand calculation (because you need the practice) to solve the LP:

$$\begin{aligned} z = \max 4x_1 - x_2 \\ 7x_1 - 2x_2 \leq 14 \\ x_2 \leq 3 \\ 2x_1 - 2x_2 \leq 3 \\ x_1 + 0x_2 \leq 2 \end{aligned}$$

- I find (starting from scratch) that after 2 simplex steps $(\bar{x}_1^1, \bar{x}_2^1) = (2, 1/2)$ and so $\bar{z}_1 = 15/2$.
- (The superscript 1 on $\bar{x}_1, \bar{x}_2$) and the subscript 1 on $\bar{z}_1$ corresponds to the fact that I am solving subproblem S_1.

- However, I can instead solve the perturbed Dual problem:

$$\min\{b^T y : -A^T y \leq -c, y \in \mathbb{R}^m, y \geq 0\}.$$

- **What I really do is apply the Dual Simplex Method to the tableau for the "perturbed" problem.**
- **The only tricky bit**: I use Row 1 of the tableau to solve for the basic variable x_1 in terms of the non-basic variables x_3 & x_4 — i.e. $x_1 = 20/7 - 1/7x_3 - 2/7x_4$.
- So the added constraint $x_1 + x_6 = 2$ becomes $-1/7x_3 - 2/7x_4 + x_6 = -6/7$.
- **Alternatively, pivot on Row 1 & Col 1 of the tableau, the result is of course the same as that on the next Slide.**

- N.B. I'll use the standard convention that the top row is Row 0 & the LH column is Column 0.
- The starting non-Canonical (non-Canonical, optimal but infeasible) Tableau is:

	x_1	x_2	x_3	x_4	x_5	x_6
59/7	0	0	4/7	1/7	0	0
20/7	1	0	1/7	2/7	0	0
3	0	1	0	1	0	0
23/7	0	0	−2/7	10/7	1	0
−6/7	0	0	−1/7	**−2/7**	0	1

- I now apply the Dual Simplex Method Alg. A.2 to this tableau.
- I choose Row 4 ($-6/7$ in LH column) and the x_4 column (as $(1/7)/(-2/7) = -1/2 > (4/7)/(-1/7) = -4$).

- After pivoting I have:

	x_1	x_2	x_3	x_4	x_5	x_6
8	0	0	1/2	0	0	1/2
2	1	0	0	0	0	1
0	0	1	−1/2	0	0	7/2
−1	0	0	**−1**	0	1	5
3	0	0	1/2	1	0	−7/2

- I now choose Row 3 (−1 in LH column) and the x_3 column.

The LP(S_1) Sub-Problem

- After pivoting I have:

	x_1	x_2	x_3	x_4	x_5	x_6
15/2	0	0	0	0	1/2	3
2	1	0	0	0	0	1
1/2	0	1	0	0	−1/2	1
1	0	0	1	0	−1	−5
5/2	0	0	0	1	1/2	−1

LP(S_1)

which is LP–optimal.

- I have $\bar{x}_1^1 = 2$, $\bar{x}_2^1 = 1/2$ and $\bar{z}_1 = 15/2$.
- **N.B.** Reading from the tableau I have $z = -15/2$ but (as before) the original problem was a max one so I must flip the sign of the optimal z.

Branching

- S_1 cannot be pruned/fathomed.
- So using the same branching rule as before I create two new nodes $S_{11} = S_1 \cap \{x : x_2 \leq 0\}$ and $S_{12} = S_1 \cap \{x : x_2 \geq 1\}$ and add them to the node list.
- The tree is now as in Figure 9 on the next Slide.

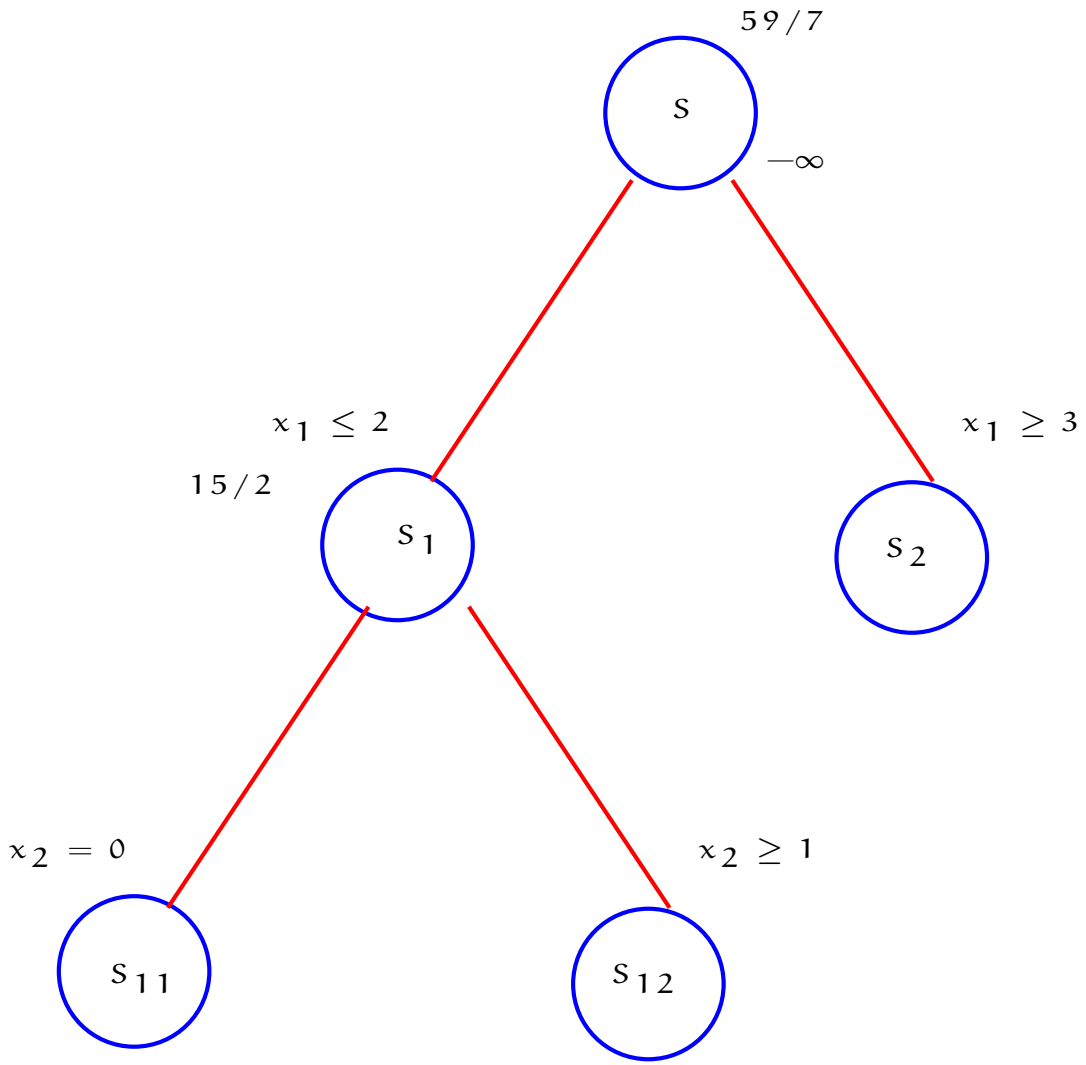

Figure 9: Partial B&B Tree # 2

Choosing a Node

- The active node list now contains S_2, S_{11} and S_{12}.
- I (arbitrarily) choose S_2, remove it from the node list and examine it in detail.

Re-optimising

- To solve LP(S_2) I use the Dual Simplex method as previously.
- The constraint $x_1 \geq 3$ is first written as $x_1 - x_6 = 3$, $x_6 \geq 0$.
- Expressing this new constraint in terms of the non-basic variables (from the Optimal tableau LP(S) for the initial LP relaxation on Slide 131): $1/7x_3 + 2/7x_4 + x_6 = -1/7$.
- Or just add the row `[-3 -1 0 0 0 0 1]` (and a column of zeros on the right!) to the tableau and pivot on Row 1 and Column 1 — the last row becomes `[ -1/7 0 0 1/7 2/7 0 1]`, equivalent to $1/7x_3 + 2/7x_4 + x_6 = -1/7$.
- But on examination of this new constraint I see that LP(S_2) **is infeasible** and so $\bar{z}_2 = -\infty$ and so node S_2 is **pruned/fathomed by infeasibility**.

Choosing a Node

- The node list now contains S_{11} and S_{12}.
- Arbitrarily choosing S_{12} I remove it from the list.

Re-optimising

- I have $S_{12} = S_1 \cap \{x : x_2 \geq 1\} \equiv S \cap \{x : x_1 \leq 2, x_2 \geq 1\}$.
- The extra constraint $x_2 \geq 1$ is first written as $x_2 - x_7 = 1$.
- I'll "flip" the sign of the constraint to $-x_2 + x_7 = -1$ so that I get "a column from the identity matrix" in the x_7 column.
- Expressing this new constraint in terms of the non-basic variables from the Optimal tableau for LP(S_1) on Slide 145, $-1/2x_5 + x_6 + x_7 = -1/2$.
- **Again, in practice I would add the row `[-1 0 -1 0 0 0 0 1]` (and a column of zeros) and pivot on Row 2 and Column 2, giving the Tableau on the next Slide.**

- I now apply Dual Simplex to the non-Canonical tableau:

	x_1	x_2	x_3	x_4	x_5	x_6	x_7
15/2	0	0	0	0	1/2	3	0
2	1	0	0	0	0	1	0
1/2	0	1	0	0	−1/2	1	0
1	0	0	1	0	−1	−5	0
5/2	0	0	0	1	1/2	−1	0
−1/2	0	0	0	0	**−1/2**	1	1

- I select Row 5 and the x_5 column.

- Pivoting I find:

	x_1	x_2	x_3	x_4	x_5	x_6	x_7
7	0	0	0	0	0	4	1
2	1	0	0	0	0	1	0
1	0	1	0	0	0	0	−1
2	0	0	1	0	0	−7	−2
2	0	0	0	1	0	−1	1
1	0	0	0	0	1	−2	−2

  which is optimal so that $\bar{x}^{12} = (2, 1)$ with z-value 7.

- As $\bar{x}^{12}$ is integer, $\underline{z}^{12} = \bar{z}^{12} = 7$.

Updating the Incumbent

- As the solution of LP(S_{12}) is integer, I update the value of the best feasible solution yet found: $\underline{z} \leftarrow \max\{\underline{z}, 7\}$, a new tighter (bigger) lower bound on the optimal z^*.
- I store the corresponding solution $(2, 1)^\mathsf{T}$ — the "incumbent".
- S_{12} is now **pruned/fathomed by optimality**.

Choosing a Node The node list now consists of S_{11} only.

Re-optimising

- $S_{11} = S \cap \{x : x_1 \leq 2, x_2 \leq 0\}$.
- **Check** using Dual Simplex (starting with optimal tableau for $LP(S_1)$ and adding the extra constraint $x_2 = 0$) that the resulting $LP(S_{11})$ has optimal solution $\bar{x}^{11} = (3/2, 0)^T$ with z–value 6.
- (Note that as the extra constraint is an equality constraint I don't add a column of zeros on the right of the tableau.)
- As $\bar{z}_{11} = 6 < \underline{z} = 7$, the node is **pruned/fathomed by bound**.

Choosing a Node

- As the node list is empty, the algorithm terminates.
- The incumbent solution $x^T = (2, 1)$ with value $z = 7$ is optimal.

- The complete branch & bound tree is shown on the next Slide.

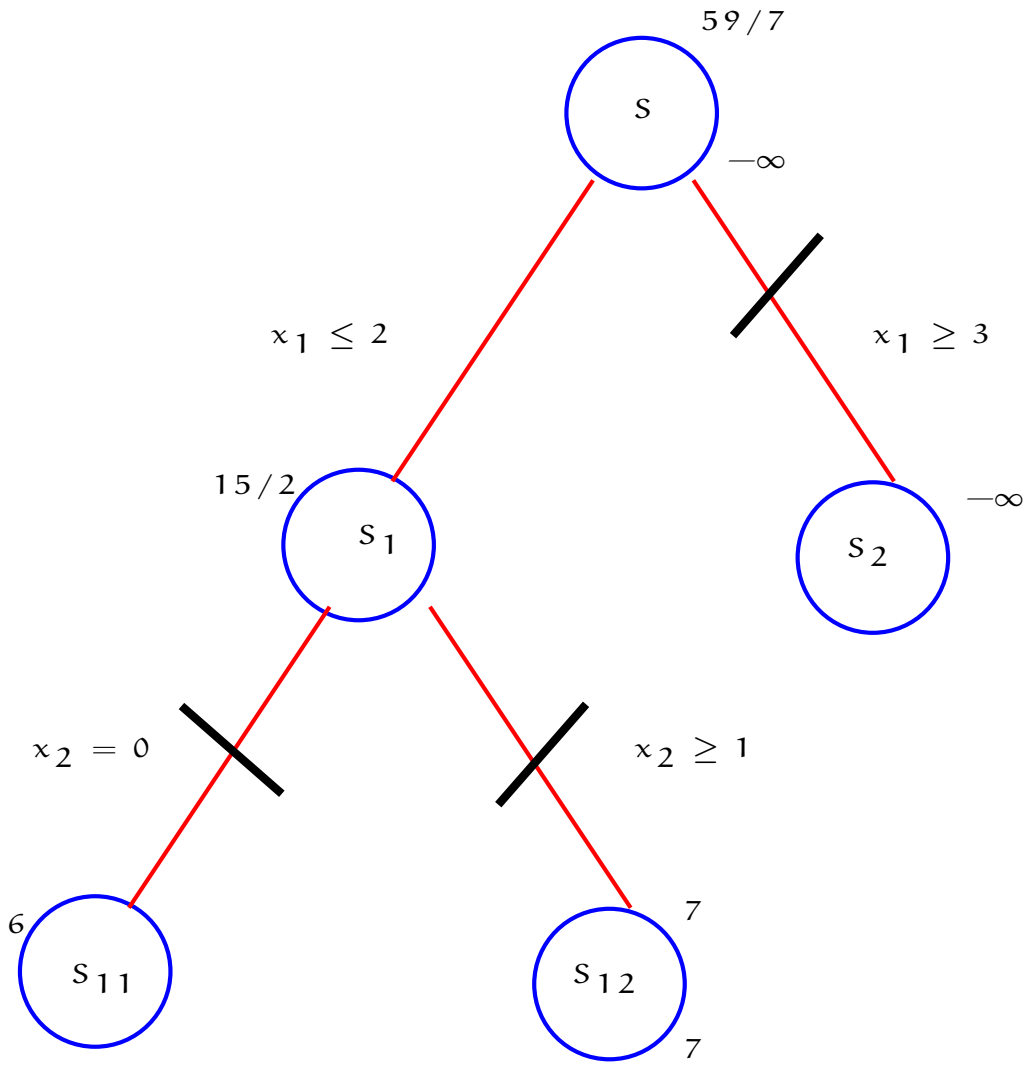

Figure 10: Complete B&B Tree

4.4 LP-Based B&B

Here is a pseudo-code description for the Branch & Bound method.

Algorithm 4.1

```
(1)  Branch & Bound for MAX problem
(2)  begin
(3)    Initialise : z = −∞, incumbent x* empty, NodeListEmpty := FALSE
(4)    Initialise node list as Problem S with formulation P.
(5)    while ¬ NodeListEmpty do
(6)          begin
(7)            Choose Problem S^i with formulation P^i.
(8)            Solve LP relax P^i_LP of P^i.
(9)            Dual bound z̄^i = LP value, x^i_LP = LP sol.
(10)           if P^i_LP is infeas.
(11)             then prune by infeasibility — remove S^i from node list.
(12)           fi
(13)           if z̄^i ≤ z
(14)             then prune by bound — remove S^i from node list.
(15)           fi
(16)           if x^i_LP integer
(17)             then update primal bound z = z̄^i and incumbent x* = x^i_LP
(18)                  prune by optimality — remove S^i from node list.
(19)           fi
(20)           if node S^i not removed
(21)             then Branch into two sub-nodes S^i_1 and S^i_2. Add S^i_1 and S^i_2 to node list.
(22)           fi
(24)         end
(25)   end
```

4.5 A Bigger B & B Example

Example 4.6 (Branch and Bound)

$$\begin{aligned} z = \min\, & 3x_1 - 7x_2 - 12x_3 \\ -3x_1 + 6x_2 + 8x_3 &\leq 12 \\ 6x_1 - 3x_2 + 7x_3 &\leq 8 \\ -6x_1 + 3x_2 + 3x_3 &\leq 5 \\ x &\in \mathbb{Z}^{3+} \end{aligned}$$

See the enumeration tree on the next Slide.. off you go! (I'll go through this in tutorial.)

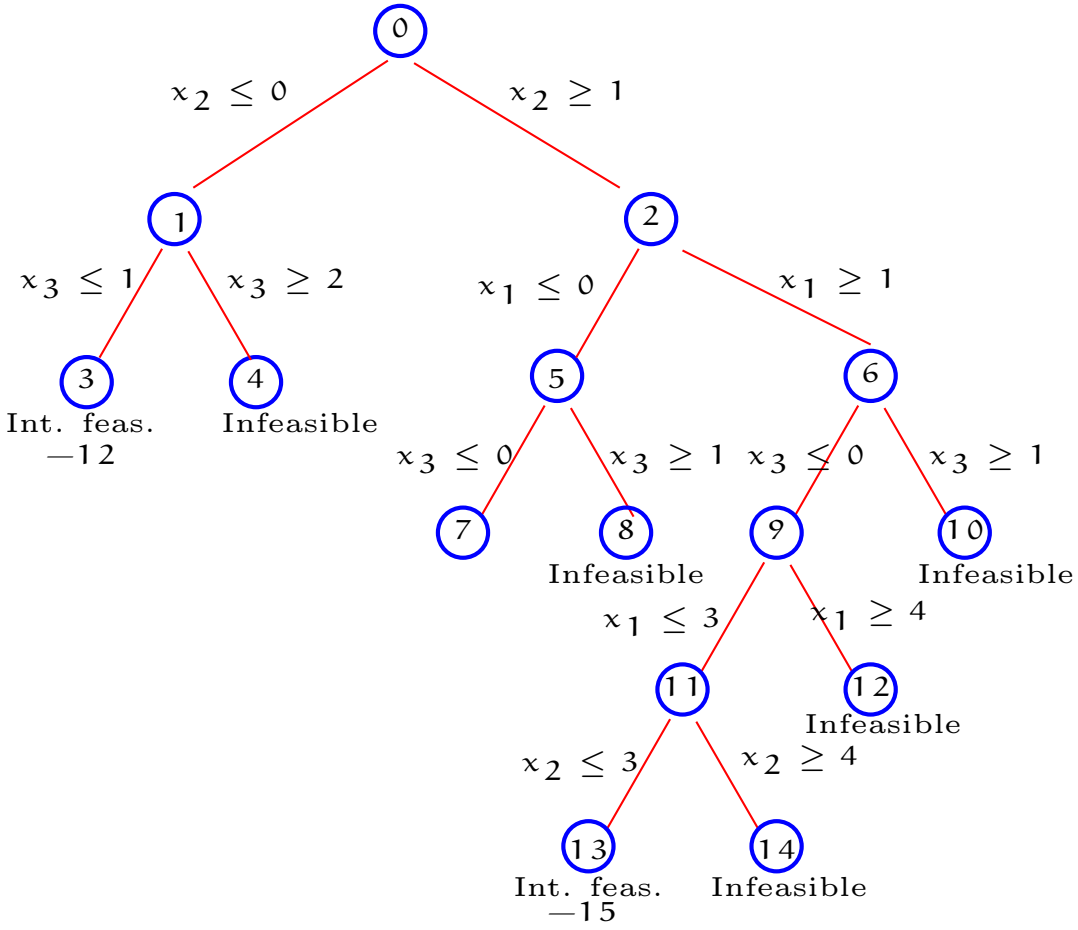

Figure 11: Large B&B Example

0. $z_L = -17.4$. $z_U = 0$ ($x_U = (0,0,0)$ — incumbent.)
1. $z_L = -13.7$
2. $z_L = -17$
3. $z_L = z_U = -12$. Incumbent is $(0,0,1)$. Prune by Opt.
4. Infeasible.
5. $z_L = -15$
6. $z_L = -16.8$
7. $z_L = -11.67$. Prune by bound as larger than z_U.
8. Infeasible.

9. $z_L = -15.56$
10. Infeasible.
11. $z_L = -15.5$
12. Infeasible.
13. $z_l = z_U = -15$. New incumbent is $(2, 3, 0)$. Prune by Opt.
14. Infeasible.

So $x^* = (2, 3, 0)$ and $z^* = -15$.

4.6 B&B for Zero-One IP's

When the IP is constrained to have only 0 or 1–valued solutions, B&B is particularly straightforward.

- It is convenient to make the "standard" problem a **min** one when solving Zero-One IP's.
- It makes keeping track of minus signs easier below.
- Remember how easily I can change a max problem into a min one.
- And I won't need to solve LP relaxations so I don't need to worry about Canonical form etc.

I'll use this example to illustrate the ideas.

Example 4.7

$$\begin{aligned} z = \min\, & 2x_1 + 3x_2 + 7x_3 + 7x_4 \\ & x_1 + x_2 - 2x_3 - 5x_4 \geq 2 \\ & -x_1 + 2x_2 + x_3 + 4x_4 \geq -3 \\ & x \in \mathbb{B}^4 \equiv \{0,1\}^4 \end{aligned}$$

- Note that all cost coefficients are non-negative — this can always be arranged by a simple change of variables, I'll explain how in Sec. 4.6.3.
- Also note I can always arrange that the (now non-negative) cost coefficients can be made non-decreasing (as in the Example) by simply re-ordering and re-labelling the variables.

The first step in the standard B&B algorithm is to solve the LP relaxation of the starting problem. I then branch on one of the variables with a non-integer LP optimal value. I could do the same here but a different relaxation is much quicker.

- For 0/1 IP's I'll use a relaxation that simply ignores the inequality constraints while keeping the 0/1 requirement. This satisfies the definition of a relaxation Def. 3.1 above. I'll call it a **binary** relaxation (not standard terminology).
- This relaxation gives a vector x that is 0/1 but may fail to satisfy some or all of the inequality constraints.
- So the relaxation gives a solution (x_B say) to $\min\{c^T x | x \in \mathbb{B}^n\}$.
- The advantage is that this **binary** relaxation is very easy to calculate when the cost coefficients are arranged as above, namely $x_B = \mathbf{0}$ — giving a lower bound of $z_L = 0$ on z^*, the optimal value of $z = c^T x$.
- If x_B is feasible for the inequality constraints then the problem is solved and $z^* = 0$.

How to get an upper bound z^u on z^* the optimal value of $z = c^T x$?

- In the Example, the largest value z can have (as the cost coefficients are all non-negative) is $z^u = 2 + 3 + 7 + 7 = 19$, corresponding to $x_1 = x_2 = x_3 = x_4 = 1$.
- So I can always find an upper bound z^u and lower bound z^L for z^* once the cost coefficients are non-negative — for the current Example I have $z^u = 19$ and $z^L = 0$.

- The strategy for the Binary B&B algorithm will be to systematically eliminate subsets of $\mathbb{B}^n$ that can be shown not to contain x^*, the optimal solution to the original 0/1 IP.
- In "standard" B&B I generated LP solutions to subproblems and checked whether they were integer feasible.
- Now I will generate solutions in $\mathbb{B}^n$ (strings of 1's and 0's) to subproblems and check whether they satisfy the inequality constraints.

Applying this approach to the Example, I'll branch on x_1: $x_1 = 0$ or $x_1 = 1$ — giving two subproblems (call them the left and right subproblems).

- I need lower bounds on the objective value over the two sub-problems.
- For the right sub-problem just set $x_1 = 1$ so $z = 2 + 3x_2 + 7x_3 + 7x_4$ whose minimum value is $z = 2$ corresponding to $x = (1, 0, 0, 0)$ — called a "**zero completion**".
- This point is infeasible for one of the inequality constraints but still gives a lower bound on z for the subproblem.
- Using the same reasoning I find a lower bound for z on the left subproblem (with $x_1 = 0$) of $z = 0$ corresponding of course to $x = (0, 0, 0, 0)$.

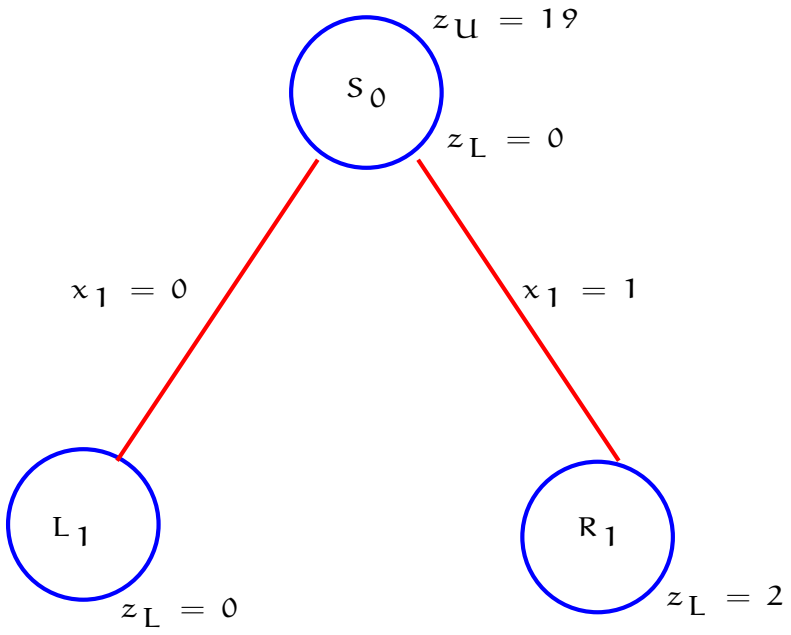

Figure 12: Binary B&B Example

Now let's see if I can "prune" either of the two nodes.

- As both lower bounds are less than the current upper bound of 19 I cannot prune by bound. (Remember that this is a **min** problem.)
- Next I check for infeasibility. For the left subproblem I ask the question: are there any "feasible completions" — binary strings starting with 0 that satisfy the inequality constraints? For a large problem it is not practical to check all such strings. Instead I will try to eliminate subproblems that can easily be found to be infeasible.
- Consider the inequality constraints for the Example for the left subproblem ($x_1 = 0$). They take the form

$$x_2 - 2x_3 - 5x_4 \geq 2$$

$$2x_2 + x_3 + 4x_4 \geq -3$$

- The largest value for the LHS in either inequality is found by setting $x_i = 1$ if the coefficient is positive and $x_i = 0$ if it is negative. So the largest values that the two LHS's can take are 1 and 7 respectively. The first inequality constraint cannot be satisfied for the left subproblem.
- So the left subproblem is **pruned/fathomed by infeasibility**.
- What about the right subproblem ($x_1 = 1$)? Using the same reasoning the inequality constraints take the form

$$1 + x_2 - 2x_3 - 5x_4 \geq 2$$
$$-1 + 2x_2 + x_3 + 4x_4 \geq -3$$

The largest possible values for the two LHS's are 2 and 6 respectively — consistent with the two constraints — so the

right subproblem cannot be pruned/fathomed by infeasibility.

- Note that I can't guarantee that the right subproblem is feasible, I just haven't shown that it is infeasible!
- Finally I need to check whether the "**zero completion**" $x_L = (1, 0, 0, 0)$ used to get the lower bound $z_L = 2$ is feasible for the right subproblem. It is easy to check that it is not.

  So the right sub-problem cannot be pruned/fathomed for optimality.

I need to branch the right sub-problem – choosing x_2 arbitrarily, branch: $x_2 = 0$ or $x_2 = 1$ — giving two new left and right subproblems.

I need to find a lower bound for z on each of the two new subproblems. For the moment I ignore feasibility and just look for the lowest possible z value.

For the right subproblem with the partial solution $x = (1, 1, *, *)$, the lowest possible value is with $x = (1, 1, 0, 0)$, namely $z_L = 5$.

For the left subproblem with the partial solution $x = (1, 0, *, *)$, the lowest possible value is with $x = (1, 0, 0, 0)$, namely $z_L = 2$.

- Neither node can be pruned/fathomed by bound as both the lower bounds are less than the current upper bound ($z_U = 19$).
- Now check for feasibility.

  With the partial solution $x = (1, 0, *, *)$, the first inequality cannot be satisfied so the left subproblem is infeasible and I can prune that node.

  With the partial solution $x = (1, 1, *, *)$, both inequalities can be satisfied so the right subproblem is not pruned/fathomed for infeasibility.
- Finally, check the right subproblem for optimality. The point $x = (1, 1, 0, 0)$ used to get a lower bound for the right subproblem is feasible for both inequality constraints — so this point is the new incumbent and $z_U = z_L = 5$.

  I can prune the node for optimality.

No unpruned nodes remain, the solution is $x^* = (1, 1, 0, 0)$.

Exercise 4.1 *Finish drawing the enumeration tree.*

Note: In a single step of the process, I never try to find the **best feasible completion**:

- when checking for infeasibility I ignore the objective value
- when finding a point that minimises the objective value I don't check whether the inequality constraints are satisfied.

4.6.1 Looking Ahead

Remember, a node can be **pruned/fathomed by bound** when $z_{\mathrm{L}}^{(\mathrm{i})} \geq z_{\mathrm{U}}$, i.e. the lower bound on z for the current node/subproblem (subproblem i, say) is greater than or equal to the current upper bound on z^*.

The bigger the $z_{\mathrm{L}}^{(\mathrm{i})}$ value for a subproblem, the sooner the node is likely to be pruned/fathomed by bound — saving work.

For this reason it is sometimes worth doing a little extra work to get better (bigger) lower bounds $z_{\mathrm{L}}^{(\mathrm{i})}$ than those found by using **zero completions**.

For example, suppose I change the Example problem slightly ($-5 \to 5$ in the first inequality) to

Example 4.8

$$\begin{aligned} z = \min\, & 2x_1 + 3x_2 + 7x_3 + 7x_4 \\ & x_1 + x_2 - 2x_3 + 5x_4 \geq 2 \\ & -x_1 + 2x_2 + x_3 + 4x_4 \geq -3 \\ & x \in \mathbb{B}^4 \equiv \{0,1\}^4 \end{aligned}$$

Solving this problem using the procedure above will result in 19 subproblems being solved — worse than simply checking all the 16 possibilities! (I might solve this in a tutorial...)

- When I started the solution of the original Example, I took $x = (0, 0, 0, 0)$.
- Although infeasible it gave a lower bound of $z_L = 0$ for z^*.
- For the altered Example, a better (bigger) lower bound than $z_L = 0$ can be found by noticing that:

  as the point $x = (0, 0, 0, 0)$ is infeasible, any feasible point must have at least one element $x_j = 1$.
- As all the cost coefficients c_j are positive, any optimal z must be at least as big as the smallest c_j, namely $c_1 = 1$.
- So I can use a lower bound of $z_L = 2$ corresponding to $x_L = (1, 0, 0, 0)$.
- If x_L was feasible for the inequality constraints, I could conclude that it was optimal.

- Unfortunately x_L isn't feasible so I'll partition the feasible set as before by branching on $x_1 = 0/1$.
- Consider the left subproblem ($x_1 = 0$) — I need a lower bound on z as usual.
  - I can use the same reasoning that I used for the starting node to get a better (bigger) lower bound on z for the left subproblem.
  - The zero completion is of course $x_L^{(left)} = (0, 0, 0, 0)$ — giving $z_L^{(left)} = 0$.
  - $x_L^{(left)}$ is infeasible (check) for the inequality constraints, so any feasible completion to $(0, *, *, *)$ must have at least one of the other components (x_2, x_3 or x_4) equal to 1.
  - Examining c_2, c_3 and c_4, I conclude that $z_L^{(left)} \geq 3$ corresponding to $x_L^{(left)} = (0, 1, 0, 0)$. (Why?)

- The same reasoning allows me to get a better (bigger) lower bound on the right subproblem.
  - I already found (at the start node) that the zero completion $x = (1, 0, 0, 0)$ is infeasible.
  - So at least one of the other components (x_2, x_3 or x_4) must be equal to 1.
  - So conclude that $z_L^{(\mathtt{right})} \geq 2 + 3 = 5$ corresponding to $x_L^{(\mathtt{right})} = (1, 1, 0, 0)$. (Why?)

Generalise the above reasoning to:

Definition 4.1 (Look Ahead Rule) *Because the zero completion is infeasible:*

- *Find the completion having exactly one previously unassigned variable* $x_j = 1$*, where* j *is chosen corresponging to the smallest* c_j*.*
- *Compute the lower bound* z_L *using that completion.*

Exercise 4.2 *Finish the solution of the problem using the Look Ahead Rule.*

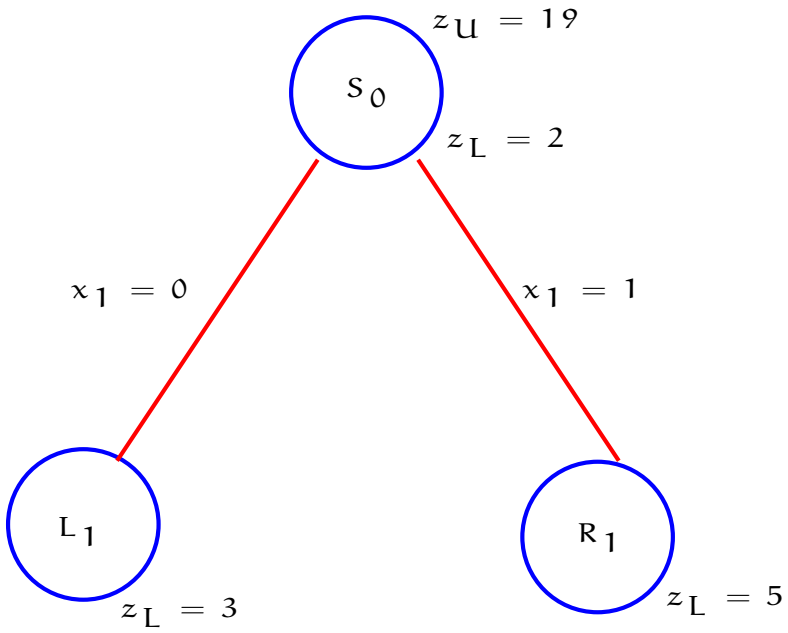

Figure 13: Altered Binary B&B Example

- Now branch L_1 on $x_2 = 0/1$.
  - Left subproblem L2: $x_2 = 0$.
  - Zero completion is $(0, 0, 0, 0)$ — $z_L = 0$ infeasible again.
  - LAR: $x_3 = 1$ (or $x_4 = 1$), $x = (0, 0, 1, 0)$, $z_L = 7$.
  - Right subproblem R2: $x_2 = 1$.
  - Zero completion is $(0, 1, 0, 0)$ — $z_L = 3$ and infeasible again.
  - LAR: $x_3 = 1$ (or $x_4 = 1$), $x = (0, 1, 1, 0)$, $z_L = 3 + 7 = 10$.
- Cannot prune either node (yet).

- Now branch R_1 on $x_2 = 0/1$.
  - Left subproblem L3: $x_2 = 0$.
  - Zero completion is $(1, 0, 0, 0)$ — $z_L = 2$ infeasible again.
  - LAR: $x_3 = 1$ (or $x_4 = 1$), $x = (1, 0, 1, 0)$, $z_L = 2 + 7 = 9$.
  - Right subproblem R3: $x_2 = 1$.
  - Zero completion is $(1, 1, 0, 0)$ — $z_L = 5$ and feasible for the inequality constraints so I also have $z_U = 5$ — new incumbent, pruned/fathomed by optimality.

But the lower bounds for L_2, R_2 and L_3 (7, 10 & 9 respectively) are all greater than the upper bound just found ($z_U = 5$) so L_2, R_2 and L_3 are **now** all **pruned/fathomed by bound** so, as R_3 is pruned/fathomed by optimality, no unpruned nodes remain and the solution is $x^* = (1, 1, 0, 0)$.

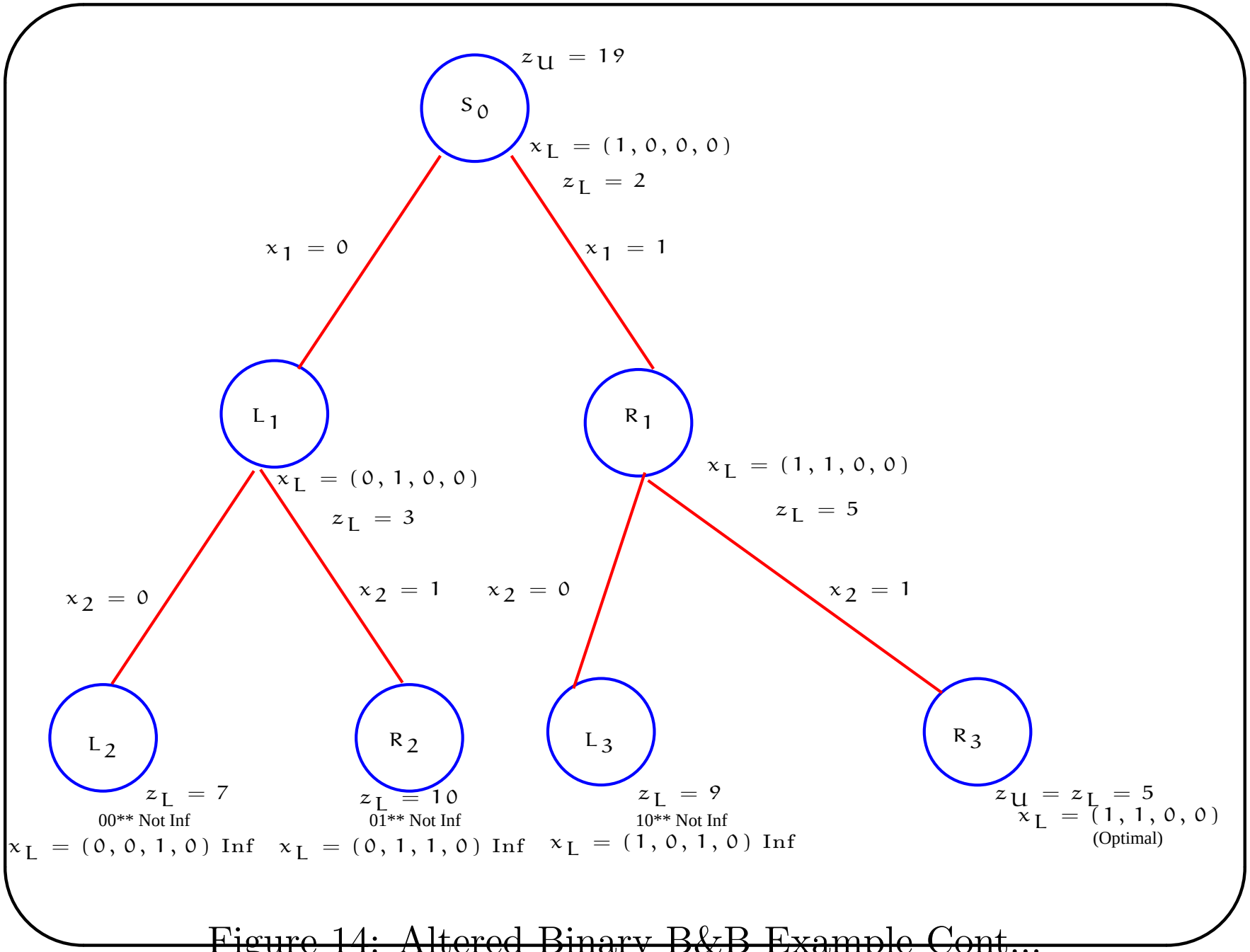

Figure 14: Altered Binary B&B Example Cont...

4.6.2 Look-Ahead Binary B&B Algorithm

Here is a pseudo-code description for the (0/1) B&B method.

Algorithm 4.2

(1) Branch & Bound
(2) begin
(3) **Initialise** : if $x = 0$is feasible then $x = 0$ is opt, STOP fi
(4) Set bounds on z^*: $z_U = \sum_j c_j$, $z_L = z_1$, $x_L = (1, 0, \ldots, 0)$
(5) Initialise NodeList to x_L.
(6) if x_L is feasible then x_L is opt, STOP fi
(7) $k \leftarrow 1$.NodeListEmpty := FALSE
(8) while $\neg$ NodeListEmpty do
(9) begin
(10) **Branch**: Select a remaining subset of feasible solutions (node)
(11) (the whole feasible set at the first iteration) and partition the subset
(12) into two smaller subsets by adding constraints $x_k = 0$ and $x_k = 1$.
(13) **Bound**: For each new subset, set x_L to the completion having its
(14) $(k+1)$st component $= 1$ and all subsequent components $= 0$.
(15) Use x_L to determine a lower bound z_L on z over that subset.
(16) **Prune/fathom**: Examine each unpruned node and prune it if
(17) $z_L \geq z_U$ **OR** one or more constraints cannot be satisfied
(18) by any completion in the subset **OR** x_L is feasible.
(19) if x_L feasible then x_L is new **incumbent**, $z_U \leftarrow z_L$.
(20) GOTO 16 and Check if other nodes can be pruned.
(21) fi
(22) **Test**:if no unpruned nodes left then STOP, **incumbent** is opt. else $k \rightarrow k + 1$. fi
(23) end
(24) end

Exercise 4.3 *Apply this algorithm to the final version on Slide 195 of Example 4.8 in the next Section:*

$$z = \min x_1 + 2x_2 + 3x_3 + 7x_4 + 8x_5 + 8x_6 - 5$$
$$5x_1 - 3x_2 + 2x_3 - 3x_4 - x_5 + 2x_6 \geq -5$$
$$-x_1 + 0x_2 + 2x_3 + x_4 + 3x_5 - 3x_6 \geq -1$$
$$x_1 + 2x_2 - x_3 + 0x_4 + 5x_5 - x_6 \geq 3$$
$$x \in \mathbb{B}^6 \equiv \{0,1\}^6$$

[Initialise] $x = 0$ *is infeasible for third inequality. Set* $z_U = 1 + 2 + 3 + 7 + 8 + 8 = 29$. *Set* $z_L = 1$ *with* $x_L = (1,0,0,0,0,0)$. *As* x_L *infeasible, set* $k = 1$.

[Branch] *Branch on* x_1, *getting the two subproblems* L_1 *and* R_1 *as shown on the next Slide.*

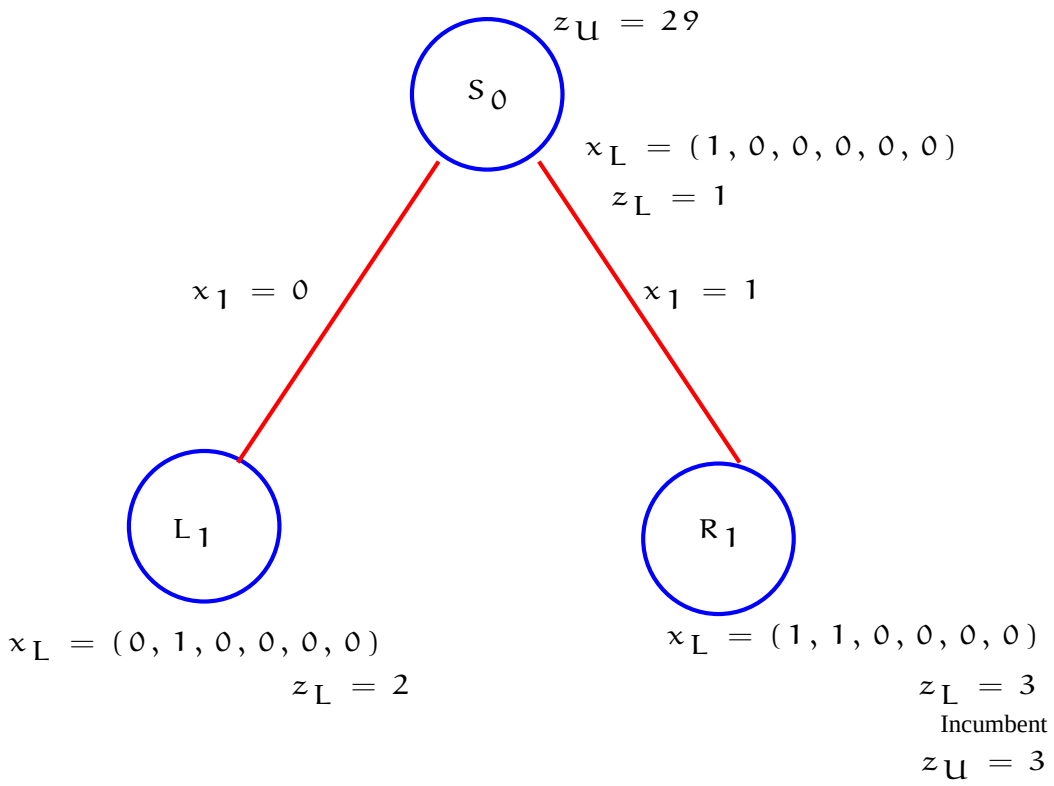

Figure 15: Third Binary B&B Example

[Prune] *For* L_1, $z_L = 2$ *using the completion* $x = (0,1,0,0,0,0)$.

The LA Rule works here as I already know from the starting node that the zero completion for L_1 *is infeasible.*

For R_1, *I use the completion* $x = (1,1,0,0,0,0)$ *to get the lower bound* $z_L = 3$ *(again the LA Rule works here as I already know from the starting node that the zero completion* $x = (1,0,0,0,0,0)$. *is infeasible.*

The point $x = (1,1,0,0,0,0)$ *is feasible for* R_1 *so I prune by optimality and update* z_U *to* $z_U = 3$.

The left node L_1 *cannot be pruned.*

[Test] *Set* $k = k + 1 = 2$ *and go to Line 10 (***Branch***) step in the algorithm.*

Finish it!

4.6.3 How to Ensure all Cost Coefficients are Non-Negative and Increasing

I'll use an example to illustrate:

Example 4.9

$$z = \min x_1 + 8x_2 - 3x_3 + 7x_4 - 2x_5 + 8x_6$$
$$5x_1 - x_2 - 2x_3 - 3x_4 + 3x_5 + 2x_6 \geq -4$$
$$-x_1 + 3x_2 - 2x_3 + x_4 + 0x_5 - 3x_6 \geq -3$$
$$x_1 + 5x_2 + x_3 + 0x_4 - 2x_5 - x_6 \geq 2$$
$$x \in \mathbb{B}^6 \equiv \{0,1\}^6$$

- First rearrange the variables so that the cost coefficients are in increasing order of magnitude, ignoring their signs.

$$z = \min x_1 - 2x_5 - 3x_3 + 7x_4 + 8x_2 + 8x_6$$

$$5x_1 + 3x_5 - 2x_3 - 3x_4 - x_2 + 2x_6 \geq -4$$

$$-x_1 + 0x_5 - 2x_3 + x_4 + 3x_2 - 3x_6 \geq -3$$

$$x_1 - 2x_5 + x_3 + 0x_4 + 5x_2 - x_6 \geq 2$$

$$x \in \mathbb{B}^6 \equiv \{0,1\}^6$$

- Now make the substitutions:

$$\begin{aligned} x_1 &\rightarrow x_1 \\ x_5 &\rightarrow 1 - x_2 \\ x_3 &\rightarrow 1 - x_3 \\ x_4 &\rightarrow x_4 \\ x_2 &\rightarrow x_5 \\ x_6 &\rightarrow x_6 \end{aligned}$$

  The rationale:

  - the variables are now ordered so that the cost coefficients are in increasing order of size.
  - By replacing the variables x_j with negative cost coefficients with $1 - x_j$ I have "flipped" the sign of the corresponding cost coefficients.
  - And as the x_j's are in $\{0, 1\}$, so are the $1 - x_j$'s.

- The result is

$$
\begin{aligned}
z = \min\, & x_1 - 2(1-x_2) - 3(1-x_3) + 7x_4 + 8x_5 + 8x_6 \\
& 5x_1 + 3(1-x_2) - 2(1-x_3) - 3x_4 - x_5 + 2x_6 \geq -4 \\
& -x_1 + 0(1-x_2) - 2(1-x_3) + x_4 + 3x_5 - 3x_6 \geq -3 \\
& x_1 - 2(1-x_2) + (1-x_3) + 0x_4 + 5x_5 - x_6 \geq 2 \\
& x \in \mathbb{B}^6 \equiv \{0,1\}^6
\end{aligned}
$$

which simplifies to:

$$
\begin{aligned}
z = \min\, & x_1 + 2x_2 + 3x_3 + 7x_4 + 8x_5 + 8x_6 - 5 \\
& 5x_1 - 3x_2 + 2x_3 - 3x_4 - x_5 + 2x_6 \geq -5 \\
& -x_1 + 0x_2 + 2x_3 + x_4 + 3x_5 - 3x_6 \geq -1 \\
& x_1 + 2x_2 - x_3 + 0x_4 + 5x_5 - x_6 \geq 3 \\
& x \in \mathbb{B}^6 \equiv \{0,1\}^6
\end{aligned}
$$

- The constant term -5 doesn't affect the minimisation — I can ignore it during the minimisation and just adjust the optimal z^*–value afterwards.

4.7 Exercises

1. Consider the following enumeration tree (for a min problem).
   (a) Give the tightest possible lower & upper bounds on the optimal z.
   (b) Which nodes may be pruned and which must be explored further?

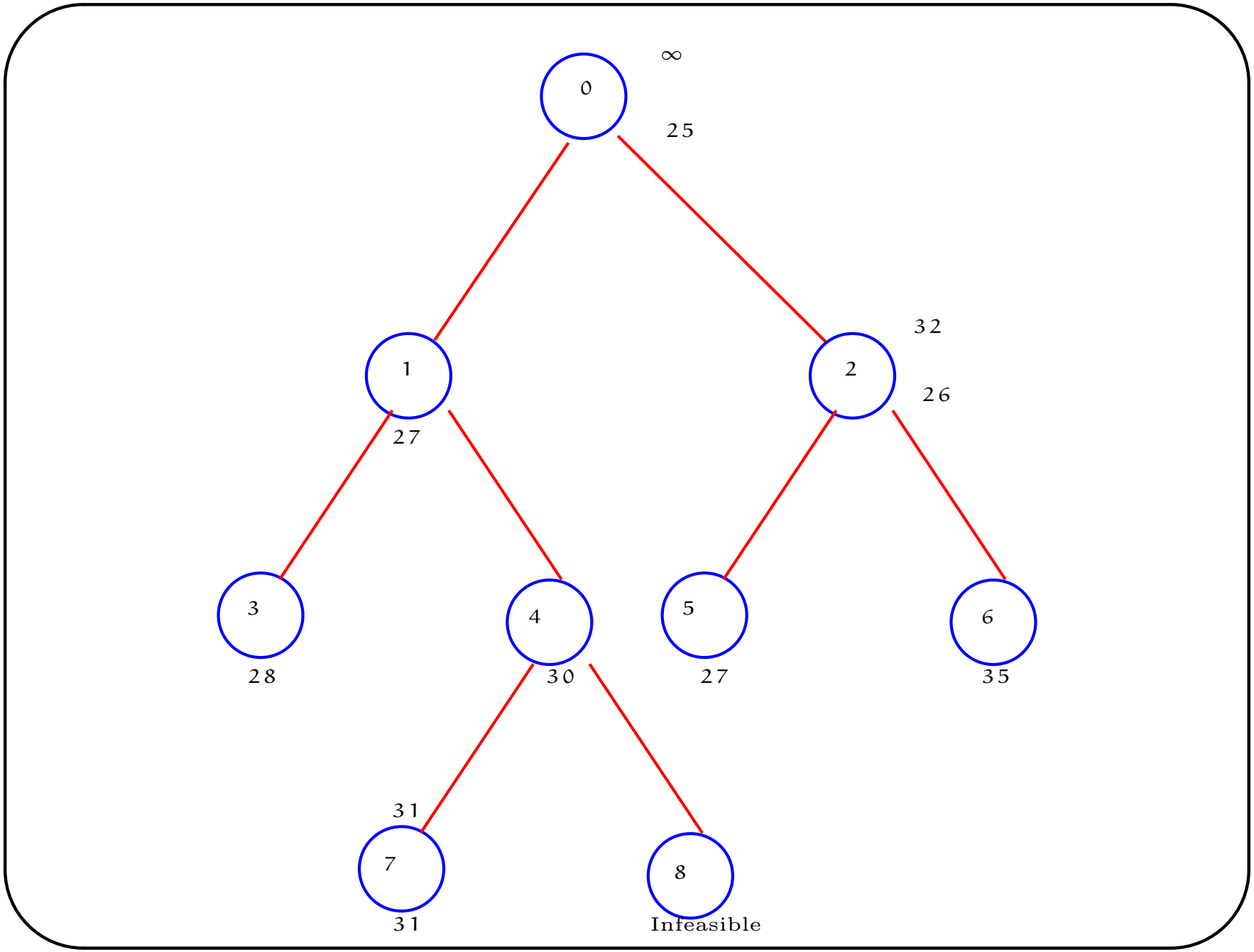

Figure 16: Enumeration Tree (min)

2. Consider the two-variable IP:

$$\begin{array}{lllll} \max & 9x_1 & & +5x_2 & \\ & 4x_1 & & +9x_2 & \leq 35 \\ & x_1 & & & \leq 6 \\ & x_1 & & -3x_2 & \geq 1 \\ & 3x_1 & & +2x_2 & \leq 19 \\ & x_1, x_2 \in \mathbb{Z}^+ & & & \end{array}$$

Solve by branch & bound and graphically (squared paper).

3. (a) Consider the $0-1$ knapsack problem

$$\max\{c^T x | a^T x \leq b, x \in \mathbb{B}^n\}, a_j, c_j > 0, j = 1, \ldots, n.$$

Show that if the ratio $\frac{c_i}{a_i}$ is non-increasing with $i = 1, \ldots, n$, $\sum_{j=1}^{r-1} a_j \leq b$ and $\sum_{j=1}^{r} a_j > b$ then the solution of the LP relaxation is $x_j = 1$ for $j = 1, \ldots, r-1$,
$x_r = \left(b - \sum_{j=1}^{r-1} a_j\right)/a_r$ and $x_r = 0$ for $i > r$.

(b) Solve the following 0–1 knapsack problem by branch & bound (use (a) to generate LP relaxations).

$$\begin{aligned} \max \quad & 17x_1 + 10x_2 + 25x_3 + 17x_4 \\ & 5x_1 + 3x_2 + 8x_3 + 7x_4 \leq 12 \\ & x \in \mathbb{B}^4 \end{aligned}$$

4. Solve the following integer knapsack problem by branch & bound.

$$\begin{aligned} \max \quad & 10x_1 + 12x_2 + 7x_3 + 3/2x_4 \\ & 4x_1 + 5x_2 + 3x_3 + 1x_4 \leq 10 \\ & x_1, x_2 \in \mathbb{Z}^+, x_3, x_4 \in \mathbb{B} \end{aligned}$$

5. Use the branch & bound method to solve the following IP:

$$\begin{aligned} \max \; & -3x_1 + 7x_2 && +12x_3 \\ & -3x_1 + 6x_2 && +8x_3 \leq 12 \\ & 6x_1 - 3x_2 && +7x_3 \leq 8 \\ & -6x_1 + 3x_2 && +3x_3 \leq 5 \\ & x_1, x_2, x_3 \in \mathbb{Z}^+ \end{aligned}$$

6. Solve the TSP with $n = 4$ and distance matrix

$$D = \begin{bmatrix} * & 7 & 6 & 3 \\ 3 & * & 6 & 9 \\ 2 & 3 & * & 1 \\ 7 & 9 & 4 & * \end{bmatrix}$$

by branch & bound using an "assignment relaxation" (as in Example 1 on Slide 78) to generate bounds.

7. Solve the min IP:

```
A=[-3 6 8
    6 -3 7
    -6 3 3];
A=[A eye(3)];% augment with slacks
b=[12 8 5]';
c=[3 -7 -12]';
c=[c; zeros(3,1)];% zero coeffs for slacks
T=[0 c';[b A]];
```

(You will need to use **SM.m** and **DSM.m** unless you have an exceptional love of matrix arithmetic.)

8. Use the B&B algorithm to solve the following IP's:

(a)

$$\begin{aligned} z = \min & -x_1 - 2x_2 \\ & -x_1 + x_2 \leq 10 \\ & 15x_1 + 16x_2 \leq 240 \\ & x \in \mathbb{Z}^{+2} \end{aligned}$$

(b)

$$\begin{aligned} z = \min & -3x_1 - 3x_2 + 13x_3 \\ & -3x_1 + 6x_2 + 7x_3 \leq 8 \\ & 6x_1 - 3x_2 + 7x_3 \leq 8 \\ & x \in \mathbb{Z}^{+3} \end{aligned}$$

9. Use the L–A Binary B&B algorithm to solve the following Binary IP's:

(a)

$$\begin{aligned} \min -2x_1 - 10x_2 - x_3 \\ 5x_1 + 2x_2 + x_3 \leq 7 \\ 2x_1 + x_2 + 7x_3 \leq 9 \\ x_1 + 3x_2 + 2x_3 \leq 5 \\ x \in \mathbb{B}^3 \end{aligned}$$

(b)

$$\begin{aligned} \min 10x_1 + 2x_2 - 11x_3 \\ 2x_1 - 7x_2 + x_3 \leq 2 \\ 5x_1 - 8x_2 + 2x_3 \leq 1 \\ x \in \mathbb{B}^3 \end{aligned}$$

(c)

$$\begin{aligned} \min\, & 2x_1 + 4x_2 - 5x_3 + 7x_4 \\ & -x_1 - 2x_2 - 3x_3 - 3x_4 \geq -8 \\ & -2x_1 + 3x_2 + x_3 + 2x_4 \geq 2 \\ & x_{\in} \mathbb{B}^4 \end{aligned}$$

(d)

$$\begin{aligned} \min\, & 0x_1 + 4x_2 + 5x_3 + 7x_4 \\ & x_1 - 2x_2 - 3x_3 + 3x_4 \leq 5 \\ & 2x_1 - 3x_2 + x_3 - 2x_4 \leq -1 \\ & \in \mathbb{B}^4 \end{aligned}$$

10. Solve the IP version of the Tables & Chairs problem from OR1:

$$\max z = 20x_1 + 15x_2$$

subject to

$$x_2 \leq 8$$

$$2x_1 - x_2 \leq 0$$

$$2x_1 + x_2 \leq 12.5$$

$$x_1, x_2 \geq 0 \quad \text{and integer.}$$

Compare the solution with the LP solution.

A Supplementary Material

A.1 Review of the Simplex Method

Remember that a Standard Form LP 1.1 can be written:

Definition A.1 (Linear Program in Standard Form) *Given* $c \in \mathbb{R}^n$, *an* $m \times n$ *real matrix* A *and* $b \in \mathbb{R}^m$ *we can write a Linear Program in Standard Form as:*

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \; & c^T x \\ Ax &= b \\ x &\geq 0 \end{aligned} \tag{A.1}$$

The Simplex Method is based on a few simple ideas:

- The $m \times n$ matrix A ($m < n$) is "wide and short" so we can't simply "solve for x" with $x = A^{-1}b$.
- Instead we pick m columns of A (elements of x) to be "basic".
- Equivalently pick m column indices $\mathcal{B} \in \{1, \ldots, n\}$ — a choice of "basis".
- Let the $n \times n$ matrix B consist of the columns of A whose indices are in $\mathcal{B}$.
- Let $\mathcal{N} = \{1, \ldots, n\} \backslash \mathcal{B}$ and N consist of the columns of A whose indices are in $\mathcal{N}$.
- Set x_N (the elements of x corresponding to $\mathcal{N}$) equal to zero.
- Solve for x_B (the elements of x corresponding to $\mathcal{B}$) by solving $x_B = B^{-1}b$.
- If $x_B \geq 0$ then we have found a feasible point!

The Bad News

- Unfortunately a random choice of $\mathtt{m}$ columns from $\mathtt{A}$ (equivalent to choosing a random corner/vertex) almost certainly will not satisfy the non-negativity cndition.
- In fact, a random choice of $\mathtt{m}$ columns from $\mathtt{A}$ will give a matrix $\mathtt{B}$ that is not even invertible!
- This is where the Simplex Method comes in.

The Good News

- The Simplex Method starts with an "initial feasible point" or equivalently with a "basis" $\mathcal{B}$ that generates $x_B \geq 0$ s.t. $Bx_B = b$ (so that x with $x_{\mathcal{B}} = x_B$ and $x_{\mathcal{N}} = x_N = 0$ satisfies $Ax = b, x \geq 0$.
- The SM then systematically generates a new basis (new solution) at each iteration that maintains feasibility $Ax = b, x \geq 0$ and reduces $c^T x$ (for a minimise problem).

Simplex Tableaux

- The Simplex tableau (plural tableaux) is a tabular representation of an LP in standard form.
- It also allows the current feasible point to be read off and the next iteration performed.

If an LP is in Standard Form (A.1) where the current objective value z is given by $z - d = \mathbf{c}^T\mathbf{x}$, we can write its tableau (using vector notation) as:

$-d$	$\mathbf{c^T}(1 \times n)$
$\mathbf{b}(m \times 1)$	$\mathbf{A}(m \times n)$

More explicitly:

$-d$	c_1	c_2	$\ldots$	c_n
b_1	A_{11}	A_{12}	$\ldots$	A_{1n}
b_2	A_{21}	A_{22}	$\ldots$	A_{2n}
$\vdots$	$\vdots$	$\vdots$	$\ldots$	$\vdots$
b_m	A_{m1}	A_{m2}	$\ldots$	A_{mn}

Canonical Form A Standard Form tableau is of limited use as it doesn't allow a solution to be "read off"'. However, if a problem/tableau has the special property that the $\mathtt{m}$ columns of the $\mathtt{m} \times \mathtt{m}$ identity matrix appear as $\mathtt{m}$ of the $\mathtt{n}$ columns of $\mathtt{A}$ then we **can** read off the current point.

We need a new idea, a **Canonical Form** problem.

Definition A.2 (Canonical Form) *An LP is in C.F. if:*

1. *the vector* $\mathtt{b}$ *has no negative elements*
2. *the* $\mathtt{m}$ *columns of the* $\mathtt{m} \times \mathtt{m}$ *identity matrix appear as* $\mathtt{m}$ *of the* $\mathtt{n}$ *columns of* $\mathtt{A}$ *(not necessarily in the right order)*
3. *the components of* $\mathbf{c}$ *corresponding to the* $\mathtt{m}$ *identity matrix columns are all zero.*

Significance of Canonical Form

- When a problem is in canonical form, we can choose the columns in A that correspond to the columns of I_m, the $m \times m$ identity matrix, to form the matrix B.
- And the basis $\mathcal{B}$ to be the set of indices of these columns.
- So $x_B = b \geq 0$ and we have a feasible point.
- The significance of Property (3) is that if we choose a new feasible point (using the pivoting operation explained below)
  - by reducing one of the basic variables to zero and
  - increasing a non-basic variable from zero

  the objective function value is only affected by the increase from zero of the non-basic variable.

A Canonical Form Example Finally, an example!

Example A.1 *Consider the problem*

$$\begin{aligned} \min -3x_1 - 2x_2 \; & \textit{subject to} \\ x_1 + x_2 + x_3 &= 5, \\ 2x_1 + (1/2)x_2 + x_4 &= 8, \\ x &\geq 0. \end{aligned}$$

We can write the Simplex Tableau corresponding to the problem as:

0	−3	−2	0	0
5	1	1	1	0
8	2	1/2	0	1

The basis is $\mathcal{B} = \{3, 4\}$ *and we can read off the value of* $x_B \equiv (x_3, x_4)^T = (5, 8)^T$. *Also* $z = d = 0$ *as* $c_3 = c_4 = 0$.

More Examples

Example A.2 *A tableau in canonical form:*

-9		0	-3	-7	0	0	4	0
2		0	2	1	1	0	-1	0
6		-1	-1	3	0	0	3	1
4		0	4	2	0	1	-2	0

$\mathcal{B} = \{4, 5, 7\}$ *and* $x_B = (2, 6, 4)^T$ *so that* $x^T = (0, 0, 0, 2, 6, 0, 4)^T$.

- *Why is* $d = 9$*??*
- *Answer: this tableau may not be the initial tableau so the value of* $z = d$ *will have been updated from the initial zero value.*

Example A.3 *Another tableau in canonical form:*

0	0	0	−5	0	−7	−6	−3
50	0	0	1	1	3	1	0
180	0	1	1	0	1	2	2
80	1	0	1	0	4	1	1

$\mathcal{B} = \{4, 2, 1\}$ *and* $x_B = (50, 180, 80)^T$ *so that* $x^T = (80, 180, 0, 50, 0, 0, 0)^T$.

We will use this example to illustrate the Simplex Method in the following.

A.2 Tableau-based Simplex Method

There are two key questions underlying the Simplex Method.

1. What currently non-basic variable should we consider increasing from zero — to get the maximum reduction in the objective function?
2. By how much can we increase a non-basic variable from zero (making it basic) without violating a constraint?

1. Let's answer the first question by looking at the Simplex Tableau in Example A.3 above.
   - The numbers in columns 2–8 of row 1 are the objective function coefficients for $x_1, \ldots, x_7$.
   - A unit increase in the non-basic variables $x_3, \ldots, x_7$ from zero results in a corresponding reduction of $-5, 0, -7, -6, -3$ in the value of the objective function.
   - So the obvious decision is to select the non-basic variable x_5 (column 6) to increase as the corresponding reduction in the the value of the objective function is -7.

2. Having chose the "entering variable" x_5 we need to determine by how much can we increase it without violating a constraint. We will "derive" a rule that answers this by referring again to the Simplex Tableau in Example A.3 above.

   - We have decided to increase x_5 from zero so set $x_5 = t > 0$ and substitute into the constraint equations represented by the tableau.
   - Leaving $x_3 = x_6 = x_7 = 0$ and substituting $x_5 = t$ and writing the basic variables x_1, x_2 and x_4 with the appropriate coefficients:

$$\begin{aligned} 50 &= 0x_1 + 0x_2 + 0 + x_4 + 3t + 0 + 0 \\ 180 &= 0x_1 + x_2 + 0 + 0x_4 + 1t + 0 + 0 \\ 80 &= x_1 + 0x_2 + 0 + 0x_4 + 4t + 0 + 0 \end{aligned}$$

- Solving for x_4, x_2 & x_1 gives:

$$\begin{aligned} x_4 &= 50 - 3t \text{ which is } \geq 0 \quad \text{for } t \leq 50/3 \\ x_2 &= 180 - t \text{ which is } \geq 0 \quad \text{for } t \leq 180 \\ x_1 &= 80 - 4t \text{ which is } \geq 0 \quad \text{for } t \leq 20. \end{aligned}$$

- So provided $t \leq 50/3$, all three currently basic variables remain non-negative.
- This motivates the general rule: "look down the **entering column** and select the row such that the ratio of the coefficient in Col 1 and that in the entering column is minimal as the **leaving row**".

- The only exception: "if the coefficient in a particular row of the **entering column** is negative then that **entering variable** may be increased indefinitely without making the **leaving variable** negative — so we exclude that row from consideration when selecting a leaving variable".
- If all the coefficients in the **entering column** (below the first row) are negative then the problem is unbounded and has no solution.

A.3 Tableau Update Rules

In the light of the answers to the two questions

1. What currently non-basic variable should we consider increasing from zero — to get the maximum reduction in the objective function?
2. By how much can we increase a non-basic variable from zero (making it basic) without violating a constraint?

we can list the **Tableau Update Rules**.

Tableau Update Rules.

1. Select the Entering Column.
2. Check for Unboundedness.
3. If Not Unbounded Select the Leaving Row.

But what next?

- How do we "update" the tableau to implement theses choices?
- The key insight is that a (Canonical Form) tableau represents a set of linear equations.

- The tableau:

$-d$	c_1	c_2	$\ldots$	c_n
b_1	A_{11}	A_{12}	$\ldots$	A_{1n}
b_2	A_{21}	A_{22}	$\ldots$	A_{2n}
$\vdots$	$\vdots$	$\vdots$	$\ldots$	$\vdots$
b_m	A_{m1}	A_{m2}	$\ldots$	A_{mn}

- represents the system of linear equations:

$$\begin{aligned}
z - d &= c_1x_1 &&+c_2x_2 &&+\ldots &&+c_nx_n \\
b_1 &= A_{11}x_1 &&+A_{12}x_2 &&+\ldots &&+A_{1n}x_n \\
b_2 &= A_{21}x_1 &&+A_{22}x_2 &&+\ldots &&+A_{2n}x_n \\
\vdots &= \vdots &&+ \vdots &&+\ldots &&+ \vdots \\
b_m &= A_{m1}x_1 &&+A_{m2}x_2 &&+\ldots &&+A_{mn}x_n
\end{aligned}$$

Observations

- The Tableau represents a set of linear equations.
- The Rows other than the First represent the constraints.
- The Top Row represents the current Objective value.
- We can perform **Row Operations** (Multiply a row by a non-zero factor or Add a Multiple of one Row to another) on the constraint Rows without changing the feasible region.
- We are essentially performing Gauss-Jordan Elimination on the Tableau (matrix).

- Remember that a Canonical Form Tableau has zeroes in the first row in the columns corresponding to the basic variables.
- Or equivalently $\mathbf{z} - \mathbf{d}$ is expressed in terms of the non-basic (currently zero) variables.
- So any row operation applied to the Constraint Rows (in order to make a currently basic variable non-basic and vice versa) must also be applied to the Top Row (to express $\mathbf{z} - \mathbf{d}$ in terms of the new non-basic variables.
- To remove a currently Basic variable from the basis we simply:
  - Divide the Leaving Row by the Pivot Element (where Entering Column crosses Leaving Row).
  - Add suitable multiples of Leaving Row to the other Rows (incl the Top Row) so as to introduce zeroes above and below the Pivot Element.

Full Tableau Update Rules

1. Select the Entering Column as that with the largest negative (for a min problem) coefficient in the Top Row.
2. Check for Unboundedness if all elements in Entering Column are negative).
3. If Not Unbounded Select the Leaving Row by finding the Row (other than the Top Row) with the smallest ratio of the coefficient in the Entering Colum and the coefficient in the Left Hand Column.

4. Divide the Leaving Row by the Pivot Element (where Entering Column crosses Leaving Row).
5. Add suitable multiples of Leaving Row to the other Rows (incl the Top Row) so as to introduce zeroes above and below the Pivot Element.
6. Repeat until all objective coefficients are non-negative (optimal solution found for min problem) or all entries in Entering column are negative (problem unbounded).
7. (For brevity from now on I'll refer to the First/coefficient Row as Row 0.)

Example A.4 *Let's see how this works with the Example above: Example A.3*

1.

0	0	0	−5	0	−7	−6	−3
50	0	0	1	1	3	1	0
180	0	1	1	0	1	2	2
80	1	0	1	0	4	1	1

The Entering Column is the x_5 *column and the Leaving Row is Row 1.*

- *Divide Row 1 across by* 3 *to get a* 1 *in the pivot position.*
- *Add suitable multiples of the new Row 1 to Row 0, & Rows 2 & 3 to introduce zeros above and below the* 1 *in the* x_5 *column.*

2.

350/3	0	0	−8/3	7/3	0	−11/3	−3
50/3	0	0	1/3	1/3	1	1/3	0
490/3	0	1	2/3	−1/3	0	5/3	2
40/3	1	0	−1/3	−4/3	0	−1/3	1

The Entering column is the x_6 *column and the Leaving Row is Row 1.*

- *Divide Row 1 across by* 1/3 *to get a* 1 *in the pivot position.*
- *Add suitable multiples of the new Row 1 to Row 0, & Rows 2 & 3 to introduce zeros above and below the* 1 *in the* x_6 *column.*

3.

300	0	0	1	6	11	0	−3
50	0	0	1	1	3	1	0
80	0	1	−1	−2	−5	0	2
30	1	0	0	−1	1	0	1

The Entering column is the x_7 *column and the Leaving Row is Row 3.*

- *Divide Row 3 across by* 1 *to get a* 1 *in the pivot position!*
- *Add suitable multiples of the (unchanged) Row 3 to Row 0, & Rows 2 & 3 to introduce zeros above the* 1 *in the* x_7 *column.*

4.

390	3	0	1	3	14	0	0
50	0	0	1	1	3	1	0
20	−2	1	−1	0	−7	0	0
30	1	0	0	−1	1	0	1

- *This is the final (Optimal) tableau.*
- *It is clearly optimal as no further reduction in the objective is possible.*
- *The final value of* z *is given by* $z = d$ *with* $-d = 390$ *so* $z = -390$.
- *The optimal value of* x *is* $x^T = (0, 20, 0, 0, 0, 50, 30)^T$.
- *The optimal basis is* $\mathcal{B} = \{2, 6, 7\}$.

We can now write the Simplex method formally as an **algorithm** on the next Slide.

With a little practice you will not need to refer to it.

Algorithm A.1 (Simplex Method)

(1) begin (Start with a Canonical tableau s.t. $\mathbf{b} \geq \mathbf{0}$.)
(2) while NOT finished do
(3) if $c_j \geq 0$ for all j
(4) then **STOP** (Tableau is optimal.)
(5) else Select j s.t. $c_j < 0$.
(6) fi
(7) if $a_{ij} \leq 0$ for all $i = 1, \ldots, m$
(8) then **STOP** (Problem is unbounded.)
(9) fi
(10) Select k such that:
(11) $\frac{b_k}{a_{kj}} = \min_i \left\{ \frac{b_i}{a_{ij}} \text{such that } a_{ij} > 0 \right\}$ (k attains the min.)
(12) Pivot on a_{kj}. (Divide Row k across by a_{kj} and add
(13) end ... multiples of Row k to the rows above & below
(14) end ... ... introducing zeros into column j.)

A.4 Finding a Canonical Tableau

Up to now we have assumed that we can find a Canonical tableau (an "all-slack initial feasible point"). In practice this may not be easy — for many problems it takes as much work to find one as it does to sovle the problem starting from the Canonical tableau. This process usually involves solving an auxiliary "Part 1" problem which minimises an auxiliary objective function (that penalises infeasibility).

See "Operations Research" by Taha or your OR1 notes for more on this topic. We will not need these ideas in this module.

A.5 Dual Simplex Method

It can often happen in Linear programming (and particularly when using LP relaxations as part of the Branch & Bound Method for IP's) that, after solving an LP by finding a tableau in optimal form, we need to add an extra constraint.

If the extra constraint has a negative b_i (element of LH Column) the "augmented" tableau will not be in Canonical form and work must be done to achieve this.(The negative LH Column element means that the current solution is infeasible).

The Dual Simplex method allows us to start with an **infeasible** starting tableau (based on the optimal tableau for the original problem).

We apply **Dual Tableau Update Rules** that **increase** the objective (for a min problem) and **reduce** the infeasibility at each stage.

The rules are similar to the standard **Tableau Update Rules** on Slide 229. Typically this is much faster than starting from scratch.

I'll illustrate the technique with an example.

Dual Simplex Example

Example A.5 (Dual Simplex Example) *Suppose that the Simplex Method has been used to solve an LP and that the following optimal tableau has been found:*

	x_1	x_2	x_3	x_4	x_5
70	0	2	1	0	4
50	0	1	−2	1	−2
5	1	1	1	0	−1

Now suppose that the optimal solution represented by this tableau ($(x_1, x_2, x_3, x_4, x_5)^T = (5, 0, 0, 50, 0)^T$) *is not acceptable and we add the extra constraint* $x_2 + x_3 \geq 10$.

Introducing a slack variable x_6 for this new constraint, the extended problem has the following Standard Form tableau:

	x_1	x_2	x_3	x_4	x_5	x_6
70	0	2	1	0	4	0
50	0	1	−2	1	−2	0
5	1	1	1	0	−1	0
−10	0	−1	−1	0	0	1

- *The tableau is no longer Canonical as the LHS Column has a negative element and in fact does not represent a feasible point as $b_3 = -10$.*
- *However it does still have all non-negative cost coefficients – the mark of an optimal solution.*
- *So the tableau represents an "infeasible but optimal" solution (strictly speaking not possible but useful shorthand).*

However, if a tableau has non-negative **costs** and a set of basic variables then

- While keeping the cost coefficients non-negative
- We can pivot to either
  - make the LH Column non-negative (so we have an optimal tableau) or
  - show that the problem is infeasible.

To see how we can derive a set of "Dual Simplex Pivot Rules", consider the following primal/dual pair where we take $c \geq 0$:

$$\begin{array}{ll} \min c^T x & \max -b^T y \\ \text{subject to } Ax \leq b & \text{subject to } -A^T x \leq c \\ x \geq 0 & y \geq 0. \end{array}$$

If we introduce non-negative "slack" vectors $\mathbf{u}$ and $\mathbf{s}$ to the primal & dual problem respectively we can represent the two problems as Standard Form Simplex tableaux:

$\mathsf{P} =$

	$\mathbf{x}$	$\mathbf{u}$
0	$\mathbf{c}^{\mathbf{T}}$	$\mathbf{0}$
$\mathbf{b}$	$\mathbf{A}$	$\mathbf{I}$

and $\mathsf{D} =$

	$\mathbf{y}$	$\mathbf{s}$
0	$\mathbf{b}^{\mathbf{T}}$	$\mathbf{0}$
$\mathbf{c}$	$-\mathbf{A}$	$\mathbf{I}$

- If $\mathbf{b} \geq 0$ then the primal tableau P is optimal as we have taken $\mathbf{c} \geq \mathbf{0}$.
- The dual tableau D is **already** in canonical form as $\mathbf{c} \geq \mathbf{0}$.
- To transform to a tableau with $\mathbf{b} \geq 0$ while keeping $\mathbf{c} \geq \mathbf{0}$, just pivot using the Simplex Pivot Rules on Slide 229!

- In the dual tableau T, we know that pivoting using the Simplex Pivot Rules will end in either an optimal or an unbounded tableau.
- Because these rules arise from applying the Simplex Algorithm to the Dual, this pivoting method is called the **Dual Simplex Method**.
- On the next Slide we will write down explicitly what these rules are.

- First note that the column in D under b_i has entries $-a_{ij}$ for $j = 1, \ldots, n$.
- So the minimum ratio for the b_i column is given by:

$$\min_j \left\{ \frac{c_j}{-a_{ij}} \text{such that } -a_{ij} > 0 \right\}$$

- or (obviously) equivalently by:

$$\max_j \left\{ \frac{c_j}{a_{ij}} \text{such that } a_{ij} < 0 \right\}$$

- So applying the Simplex Algorithm to the Dual tableau D gives us the following [**Dual Simplex Method** (notice the similarity to the Simplex Method Alg. A.1 above):

Algorithm A.2 (Dual Simplex Method)

(1) begin (Start with a tableau s.t. $\mathbf{c} \geq \mathbf{0}$.)
(2) while NOT finished do
(3) if $b_i \geq 0$ for all i
(4) then **STOP** (Tableau is optimal.)
(5) else Select i s.t. $b_i < 0$.
(6) fi
(7) if $-a_{ij} \leq 0$ for all $j = 1, \ldots, n$
(8) then **STOP** (Dual unbounded $\equiv$ Primal infeasible.)
(9) fi
(10) Select k such that:
(11) $\frac{c_k}{a_{ik}} = \max_j \left\{ \frac{c_j}{a_{ij}} \text{such that } a_{ij} < 0 \right\}$ (k attains max.)
(12) Pivot on a_{ik}. (Divide Row k across by a_{ik} and add
(13) end ... multiples of Row k to the rows above & below
(14) end ... ... introducing zeros into column i.)

- By writing the rules in this form we can see that if we perform pivots on the primal tableau **using these rules** the pivots maintain non-negativity of the cost coefficients **c** and either achieve non-negativity of the LH colum **b** or end with an infeasible tableau.
- We illustrate with Example A.5 above — writing the tableaux P & T.
- To make the example easier to follow we first rearrange the columns of P so that the columns of the Identity matrix are on the right.

Shuffle the columns of

	x_1	x_2	x_3	x_4	x_5	x_6
70	0	2	1	0	4	0
50	0	1	−2	1	−2	0
5	1	1	1	0	−1	0
−10	0	−1	−1	0	0	1

to give P=

	x_2	x_3	x_5	x_4	x_1	x_6
70	2	1	4	0	0	0
50	1	−2	−2	1	0	0
5	1	1	−1	0	1	0
−10	−1	−1	0	0	0	1

- Writing P & D together:

P=

	x_2	x_3	x_5	x_4	x_1	x_6
70	2	1	4	0	0	0
50	1	−2	−2	1	0	0
5	1	1	−1	0	1	0
−10	−1	−1	0	0	0	1

and D=

	y_1	y_2	y_3	y_4	y_5	y_6
−70	50	5	−10	0	0	0
2	−1	−1	1	1	0	0
1	2	−1	1	0	1	0
4	2	1	0	0	0	1

- The dual simplex pivot rules select the **RED** position in D.
- More important, applying these rules gives the **RED** position in P.

- If we perform these pivots in P & D we find (again shuffling the columns of P & D so that the identity is on the right):

P=

	x_2	x_6	x_5	x_4	x_1	x_3
60	1	1	4	0	0	0
70	3	−2	−2	1	0	0
−5	0	1	**−1**	0	1	0
10	1	1	0	0	0	1

and D=

	y_1	y_2	y_5	y_4	y_3	y_6
−60	70	−5	10	0	0	0
1	−3	0	−1	1	0	0
1	2	−1	1	0	1	0
4	2	**1**	0	0	0	1

- Finally, we perform these pivots in P & D and we find (again shuffling the columns of P & D so that the identity is on the right) **both tableaux are optimal**:

P=

	x_2	x_6	x_1	x_4	x_5	x_3
40	1	5	4	0	0	0
80	3	−4	−2	1	0	0
5	0	−1	−1	0	1	0
10	1	1	0	0	0	1

and D=

	y_1	y_6	y_5	y_4	y_3	y_2
−40	80	5	10	0	0	0
1	−3	0	−1	1	0	0
5	4	1	1	0	1	0
4	2	1	0	0	0	1

- Of course in practice we **only** pivot on P using the Dual Simplex Algorithm above.
- The sequence of P/D (Primal/Dual) tableau pairs does illustrate the fact that pivoting with the Simplex Method on the Dual tableau D is equivalent to pivoting with the Dual Simplex Method on the Primal tableau P!
- In practice we also don't shuffle the columns so that the identity I appears on the right
  - We did it here to make it easier to see the relationship between P & T.

One Last Dual Simplex Example

- The following example has non-negative ("optimal") cost coefficients and a set of basic variables.
- In this example we will not rearrange the columns so that the Identity columns appear on the right — there is no need in general as noted above.
- The following tableau would be optimal except for the fact that there are two rows with $b_i < 0$.
- We arbitrarily select the first row as the pivot row; $i = 1$.

	x_1	x_2	x_3	x_4	x_5	x_6
-10	0	0	3	1	2	0
-5	1	0	-1	0	-1	0
2	0	0	2	3	0	1
-7	0	1	2	-1	-1	0

- Now choose a column using the Rule in Line 10 of the Algorithm, the $x_3 (j = 3)$ and $x_5 (j = 5)$ colums both have negative a_{ij} but $\frac{2}{-1} > \frac{3}{-1}$ so we choose the x_5 column.
- The pivot element is shown in **RED** on previous Slide.
- Pivoting gives:

	x_1	x_2	x_3	x_4	x_5	x_6
-20	2	0	1	1	0	0
5	-1	0	1	0	1	0
2	0	0	2	3	0	1
-2	-1	1	3	**-1**	0	0

- Choose Row 3 and the x_4 column as $\frac{1}{-1} > \frac{2}{-1}$.

	x_1	x_2	x_3	x_4	x_5	x_6
-22	1	1	4	0	0	0
5	-1	0	1	0	1	0
-4	**-3**	3	11	0	0	1
2	1	-1	-3	1	0	0

- Choose Row 2 and the x_1 column as -3 is the only negative element in Row 2.

	x_1	x_2	x_3	x_4	x_5	x_6
$-70/3$	0	2	$23/3$	0	0	$1/3$
$19/3$	0	-1	$-8/3$	0	1	$-1/3$
$4/3$	1	-1	$-11/3$	0	0	$-1/3$
$2/3$	0	0	$2/3$	1	0	$1/3$

Optimal!

Exercise A.1 *Work through the last Example, taking Row 3 rather than Row 1 as the pivot row.*

A.6 Proof of Thm. 2.2

Proof: RTP that all extreme points of conv(X) lie in $X = \{x^{(1)}, \ldots, x^{(t)}\}$. Assume not.

- Then conv(X) contains an extreme point $\mathbf{v}$ say, which is not equal to any of the $x^{(i)}$, $i = 1, \ldots, t$.
- I also have that $\mathbf{v} \in$ conv(X) by definition of an extreme point (Def. 2.5).
- So $\mathbf{v} = \sum_{i=1}^{t} \lambda_i x^{(i)}$, s.t. $\sum_{i=1}^{t} \lambda_i = 1$ for some non-negative set of numbers λ_i, $i = 1, \ldots, t$.
- As $\mathbf{v}$ is not equal to any of the $x^{(i)}$, $i = 1, \ldots, t$, I must have $\lambda_i < 1$ for each i.
- Write $\mathbf{v} = \sum_{i:\lambda_i > 0} \lambda_i x^{(i)}$ and divide this sum into two parts, say $i < M$ and $i \geq M$.

- It must be possible to do this as otherwise $\mathbf{v}$ is equal to one of the $\mathbf{x}^{(i)}$.
- So write $\mathbf{v} = \sum_{i:\lambda_i>0, i<M} \lambda_i \mathbf{x}^{(i)} + \sum_{i:\lambda_i>0, i\geq M} \lambda_i \mathbf{x}^{(i)}$.
- Now write $L = \sum_{i:\lambda_i>0, i<M} \lambda_i$ and $R = \sum_{i:\lambda_i>0, i\geq M} \lambda_i$ — so that $L + R = 1$.
- Finally, $\mathbf{v} = L \sum_{i:\lambda_i>0, i<M} \frac{\lambda_i}{L} \mathbf{x}^{(i)} + R \sum_{i:\lambda_i>0, i\geq M} \frac{\lambda_i}{R} \mathbf{x}^{(i)}$ and so (check) $\mathbf{v}$ is on a line joining two elements of $\text{conv}(X)$ — contradicting the fact that $\mathbf{v}$ is an extreme point of $\text{conv}(X)$. ■