Lecture 9: Linear and integer programming algorithms

- Solving MIPs:
  Relaxations
  Branch and bound search
- Solving LPs:
  Simplex algorithm

Solving MIPs

- A typical approach is use branch and bound search with a suitable relaxation.
- A relaxation of a problem removes constraints in order to get an easier to solve problem.
- Given a MIP P, its relaxation $R(P)$ is a problem satisfying the following conditions (for a minimization problem P):
  - the optimal solution value to $R(P)$ is no more than that of P,
  - if the solution to $R(P)$ is feasible to P, then it is optimal for P,
  - if $R(P)$ is infeasible, then so is P.
- A useful relaxation of a MIP P satisfying these condition is the linear relaxation $LR(P)$ of P which is obtained by removing the integrality constraints from P.
- Linear relaxation is computationally interesting because it is a strong relaxation, it provides a global view on the constraints, and it generates "dual values" for the constraints.

Branch and bound

Given a MIP P and its relaxation $R(P)$, branch and bound works as follows:

1. Solve $R(P)$ to get an optimal relaxation solution x^*.
2. If $R(P)$ is infeasible, then so is P
   else if x^* is feasible to P, then x^* is optimal to P
   else create new problems $P_1, \ldots, P_k$ by branching and solve recursively. Stop examining a subproblem if it cannot be optimal to P (bounding).

Branching

- Branching creates new subproblems based on an optimal solution x^* to $R(P)$ that is infeasible to P.
- The subproblems $P_1, \ldots, P_k$ must satisfy the following properties:
  - Every feasible solution to P is feasible to at least one of $P_1, \ldots, P_k$.
  - x^* is infeasible in each of $R(P_1), \ldots, R(P_k)$.
- For the linear relaxation, x^* is infeasible iff there is variable x_j that has a fractional value x_j^* in x^*.
- In this case we can create two new problems:
  - one with the additional constraint $x_j \leq \lfloor x_j^* \rfloor$;
  - one with the additional constraint $x_j \geq \lfloor x_j^* \rfloor + 1$.

Bounding

- Bounding also uses relaxation.
- Suppose we have generated a feasible solution to some subproblem with objective value z^*. This could be optimal to a subproblem but we do not yet know whether it is optimal to P.
- If we now have a subproblem with relaxation objective value $z' \geq z^*$, then we can cease examining this subproblem (bounding).
- This is because further branching can only increase the objective value.

Improving Effectiveness

- Careful formulation
  - Strong relaxations typically work well but are often bigger in size.
  - Break symmetries.
  - Multiple “big-M” values often lead to performance problems.
  - Deciding which formulation works better needs often experimentation.
- Cutting plans
  These are constraints that are added to a relaxation to “cut off” the optimal relaxation solution x^*. Often are problem specific but there are also general techniques (e.g. Gomory cuts).
- Special branching rules
  In many systems, for example, Special Ordered Sets are available.

Solving Linear Relaxation

- Linear Relaxation of a MIP gives a linear program (LP)
- There are a number of well-known techniques for solving LPs
  - Simplex
    The oldest and most widely used method with very mature implementation techniques.
    Worst-case time complexity exponential but seems to work extremely well in practice.
  - Interior point methods
    A newer approach; polynomial time worst case time complexity; implementation techniques advancing
- Next Simplex method is reviewed as an example.

Simplex Method

- Assume that the linear program is in standard form:
  $$\begin{aligned}&\min cx \text{ s.t.}\\ &Ax = b\\ &x \geq 0\end{aligned}$$
- The basic idea: start from a basic feasible solution (“a corner point”) and look at the adjacent ones. If an improvement in cost is possible by moving to an adjacent solution, we do so. An optimal solution has been found if no improvement is possible.
- Next we briefly review the basic concepts needed:
  - basic feasible solutions (bfs)
  - move from one bfs to another (pivoting)
  - the overall simplex algorithm

Basic Feasible Solutions

- Assume an LP in standard form.

$$\begin{array}{ll} \min cx \text{ s.t.} & \\ Ax = b & \text{where } A \text{ is a } m \times n \text{ matrix and } m < n. \\ x \geq 0 & \end{array}$$

- A basis of A is a linearly independent collection $\{A_{j_1}, \ldots, A_{j_m}\}$ of column vectors A_{j_i} of A. The basis can be treated as a $m \times m$ nonsingular matrix $B = [A_{j_i}]$.
- The basic solution corresponding to B is a vector x such that
  - $x_j = 0$ if A_j is not in the basis and
  - x_{j_k} = the kth component of $B^{-1}b$ otherwise.
- A basic feasible solution (bfs) is a basic solution such that $x_i \geq 0$ for all i.
- Note that a basic solution is obtained by setting $n - m$ variables to zero and solving the resulting set of equations for the remaining m variables. If there is a unique solution, this is gives a basic solution.

Example

- Consider the LP

$$\begin{array}{l} \min\ 2x_2 + x_4 + 5x_7 \\ \begin{array}{ccccccccccccccc} x_1 & + & x_2 & + & x_3 & + & x_4 & & & & & & & = & 4 \\ x_1 & & & & & & & + & x_5 & & & & & = & 2 \\ & & & & x_3 & & & & & + & x_6 & & & = & 3 \\ & & 3x_2 & + & x_3 & & & & & & & + & x_7 & = & 6 \end{array} \\ x_1, \ldots, x_7 \geq 0 \end{array}$$

A possible basis: $\{A_4, A_5, A_6, A_7\}$ and the corresponding matrix:

$$B = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Now the basic solution is $x0 = (0,0,0,4,2,3,6)$ since $x0_1 = x0_2 = x0_3 = 0$ and

$$B^{-1}b = b = \begin{pmatrix} 4 \\ 2 \\ 3 \\ 6 \end{pmatrix}$$

Moving from bfs to bfs

- The idea is to remove one variable from the basis and replace it with another. This is called pivoting.
- In simplex this is organized as a manipulation of a tableau where, for instance, a set of equations

$$\begin{array}{ccccccccccc} 3x_1 & + & 2x_2 & + & x_3 & & & & & = & 1 \\ 5x_1 & + & x_2 & + & x_3 & + & x_4 & & & = & 3 \\ 2x_2 & + & 5x_2 & + & x_3 & & & + & x_5 & = & 4 \end{array}$$

is represented as

	x_1	x_2	x_3	x_4	x_5
1	3	2	1	0	0
3	5	1	1	1	0
4	2	5	1	0	1

Tableaux

- Pivoting is handled by keeping the set of equations diagonalized with respect to the basic variables.
- This can be achieved using elementary row operations: multiplying a row with a non-zero constant; adding a row to another.

Example.
Consider the set of equations

	x_1	x_2	x_3	x_4	x_5
1	3	2	1	0	0
3	5	1	1	1	0
4	2	5	1	0	1

Given a basis $B = \{A_3, A_4, A_5\}$, we can transform the tableau to a diagonalized form w.r.t. the basic variables x_3, x_4, x_5 by multiplying Row 1 with -1 and adding it to Rows 2 and 3:

	x_1	x_2	x_3	x_4	x_5
1	3	2	1	0	0
2	2	-1	0	1	0
3	-1	3	0	0	1

Tableaux—cont'd

- Notice that in the diagonalized form column 0 gives the values of the basic variables in the bfs $x0$ in question:

$$x0_{B(i)} = x_{i,0}, i = 1, \dots, m$$

where $B(i)$ denotes the ith basic variable (column).

- **Example**. Consider the set of equations:

	x_1	x_2	x_3	x_4	x_5
1	3	2	1	0	0
2	2	-1	0	1	0
3	-1	3	0	0	1

Given the basis $B = \{A_3, A_4, A_5\}$, $B(1) = 3, B(2) = 4, B(3) = 5$ and for its basic solution $x0$ holds: $x0_3 = 1, x0_4 = 2, x0_5 = 3$

Pivoting

- In pivoting variable x_j enters the basis and variable x_i leaves it.
- In the tableau this defines a pivot element $x_{l,j}$ where column j corresponds to the entering variable x_j and row l to the leaving variable x_i such that $B(l) = i$. We say that we pivot on $x_{l,j}$.
- In pivoting the tableau is brought to the diagonalized form w.r.t. the new basis using elementary row operations (Gaussian elimination):
  - for the pivot row l, all elements are divided by the pivot element and, hence, the pivot element in the new tableau is 1;
  - for other rows i, the resulting pivot row multiplied by $x_{i,j}$ is subtracted from the row, and, hence all elements in column j (except the pivot element) are 0 in the new tableau.
- This means that

$$\begin{array}{ll} x'_{l,q} = \frac{x_{l,q}}{x_{l,j}} & q = 0, \dots, n \\ x'_{i,q} = x_{i,q} - x'_{l,q} x_{i,j} & i = 1, \dots, m; i \neq l \\ & q = 0, \dots, n \end{array}$$

where $x_{i,j}$ and $x'_{i,j}$ are the old and new tableaux, respectively.

Example

- Consider the tableau

	x_1	x_2	x_3	x_4	x_5
1	3	2	1	0	0
2	2	-1	0	1	0
3	-1	3	0	0	1

- and the case where x_1 enters and x_3 leaves the basis.
- Now the pivot element is $x_{1,1}$ as $B(1) = 3$, and
- the new tableau

	x_1	x_2	x_3	x_4	x_5
$\frac{1}{3}$	1	$\frac{2}{3}$	$\frac{1}{3}$	0	0
$\frac{4}{3}$	0	$-\frac{7}{3}$	$-\frac{2}{3}$	1	0
$\frac{10}{3}$	0	$\frac{11}{3}$	$\frac{1}{3}$	0	1

- and the new basis is x_1, x_4, x_5 and, hence, $B(1) = 1, B(2) = 4, B(3) = 5$.

Cost Function in the Tableau

- A cost function $z = cx$ can added as an extra equation $-z + cx = 0$ to the tableau (no need to add a column for z).
- To start, we need a bfs and to make zero the c_js for the basic columns.
- This can be done using elementary row operations.
- Consider the example with 3, 4, and 5 as the basic columns.
- After transformation to the diagonalized form, subtract the resulting Rows 1, 2, 3 from Row 0, to get the desired form:
- Our running example and a cost function $z = x_1 + x_2 + x_3 + x_4 + x_5$ lead to a tableau:

	x_1	x_2	x_3	x_4	x_5
0	1	1	1	1	1
1	3	2	1	0	0
3	5	1	1	1	0
4	2	5	1	0	1

	x_1	x_2	x_3	x_4	x_5
-6	-3	-3	0	0	0
1	3	2	1	0	0
2	2	-1	0	1	0
3	-1	3	0	0	1

Choosing a Profitable Column

- It turns out that the cost function can be improved if we move to a bfs containing a non-basic variable x_j where the corresponding value c_j in the tableau is negative.
- If no such c_j exists, then an optimal solution has been found.
- In the previous example the bfs corresponding to Columns 3, 4, and 5 is not optimal because $c_1 = c_2 = -3 < 0$.
- Hence, we could move to a new bfs with entering variable x_1 or x_2 to improve the cost function.
- But how to choose the leaving variable?

Choosing the Leaving Variable

- The idea is to move to an adjacent bfs containing the entering variable x_j.
- In order not to miss an adjacent bfs we need to choose a pivot element $x_{k,j}$ with the smallest positive ratio $\frac{x0_k}{x_{k,j}}$, that is, a $x_{k,j}$ such that
$$\frac{x0_k}{x_{k,j}} = \min_{\substack{i \\ x_{i,j}>0}} \left(\frac{x0_i}{x_{i,j}}\right)$$
where $x0$ is the current *bfs*.
- Then the leaving variable is $B(k)$.

Example

- Consider the tableau

	x_1	x_2	x_3	x_4	x_5
-6	-3	-3	0	0	0
1	3	2	1	0	0
2	2	-1	0	1	0
3	-1	3	0	0	1

- If x_2 is the entering variable, then the pivot element is $x_{1,2}$ because the smallest positive ratio $\frac{x0_k}{x_{k,2}}$ is $\frac{1}{2}$ for $k = 1$.

Simplex algorithm

```
procedure Simplex
opt := "no"; unbounded := "no";
while opt = "no" and unbounded = "no" do
    if $c_j \geq 0$ for all $j$ then opt := "yes"
    else
        choose any $j$ such that $c_j < 0$ ;
        if$x_{i,j} \leq 0$ for all $i$ then unbounded := "yes"
        else
            find $\min_{i, x_{i,j}>0} (\frac{x0_i}{x_{i,j}}) = \frac{x0_k}{x_{k,j}}$
            and pivot on $x_{k,j}$
        end if
    end if
end while.
```

Example

- Consider the tableau on the right (above).
- Running Simplex on this tableau, we notice that for Columns 1 and 2, $c_j < 0$.
- If we choose c_2, then we need to pivot on $x_{1,2}$ as argued in the previous example.
- Then the new tableau is on the right (below).
- Here all c_js are positive and, hence, an optimal solution $(0, \frac{1}{2}, 0, \frac{5}{2}, \frac{3}{2})$ has been found with cost $\frac{9}{2}$ $(-z = -\frac{9}{2})$.

	x_1	x_2	x_3	x_4	x_5
-6	-3	-3	0	0	0
1	3	2	1	0	0
2	2	-1	0	1	0
3	-1	3	0	0	1

	x_1	x_2	x_3	x_4	x_5
$-\frac{9}{2}$	$\frac{3}{2}$	0	$\frac{3}{2}$	0	0
$\frac{1}{2}$	$\frac{3}{2}$	1	$\frac{1}{2}$	0	0
$\frac{5}{2}$	$\frac{7}{2}$	0	$\frac{1}{2}$	1	0
$\frac{3}{2}$	$-\frac{11}{2}$	0	$-\frac{3}{2}$	0	1

Further Issues

For an efficient implementation of Simplex there are a number issues that need to be handled:

- Finding the first bfs to start Simplex:
  artificial variable method, two-phase method, . . .
- How to choose the entering variable:
  nonbasic gradient method (choosing the most negative c_j), greatest increment method, . . .
- How to choose the pivot element in case of a tie:
  avoiding cycling, . . .

Summary: Solving MIPs

- Experiment with different formulations as well as different systems and solving techniques to see which performs best.
- Avoid multiple “big-M” values.
- Try to break symmetries.
- Do not introduce unnecessary integer variables.
- Scale the coefficients in the constraint to as small as possible.