Inmas 2021: Modeling and Optimization

Session 3: How do solvers work?

Alexander Estes

Solving LPs

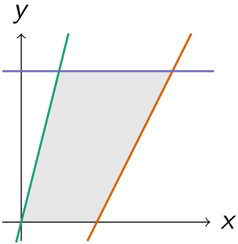
How do solvers work?

- Several methods
- One of the oldest methods is *simplex method*
- Simplex method still widely in use today

Simplex method I

Important facts about LPs:

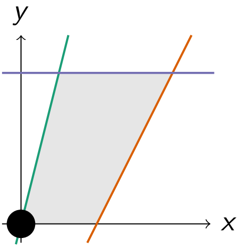
- The feasible region of an LP is a polyhedron.
- The optimal solution is a vertex of the polyhedron (if it exists).



$$
\begin{aligned}
y - 4x &\leq 0 \\
2x - y &\leq 2 \\
y &\leq 2 \\
x &\geq 0 \\
y &\geq 0
\end{aligned}
$$

Simplex method II

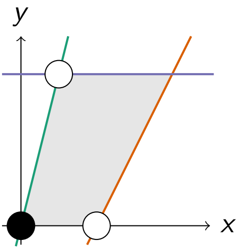
Start at some vertex:



$$
\begin{aligned}
y - 4x &\leq 0 \\
2x - y &\leq 2 \\
y &\leq 2 \\
x &\geq 0 \\
y &\geq 0
\end{aligned}
$$

Simplex method III

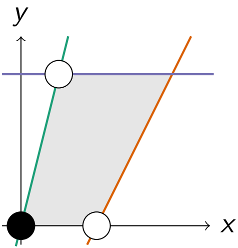
See if any neighboring vertex has a better objective (this can be done in $O(m)$ time where m is number of constraints)



$$\begin{aligned} y - 4x &\leq 0 \\ 2x - y &\leq 2 \\ y &\leq 2 \\ x &\geq 0 \\ y &\geq 0 \end{aligned}$$

Simplex method IV

If a neighboring vertex has a better objective move to that vertex (this can be done in $O(mn)$ time). If not, then solution is optimal.



$$
\begin{aligned}
y - 4x &\leq 0 \\
2x - y &\leq 2 \\
y &\leq 2 \\
x &\geq 0 \\
y &\geq 0
\end{aligned}
$$

How good is the Simplex Algorithm?

- The number of vertices on a polygon can be exponential in the number of constraints. For example:

  $$0 \leq x_i \leq 1$$

  for $i = 1, \ldots n$ has $2n$ constraints and 2^n vertices.
- In some cases, the simplex algorithm can visit every vertex!
- These types of instances are extremely unlikely to occur naturally.
- Simplex is very effective in practice.

Alternatives to Simplex

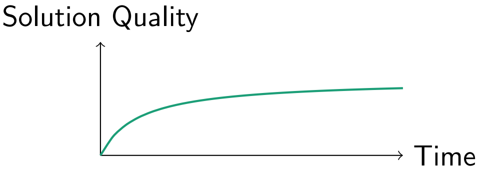
Interior point algorithms.

- Solutions follow a path in interior of polyhedron.
- Polynomial time.
- On some instances (especially extremely large, sparse instances), better than simplex.
- Also available in Gurobi and other commercial solvers.
- *Linear Optimization* by Bertsimas and Tsitsiklis
- *Convex Optimization* by Boyd and Vandenbergh

Duality I

LP methods often find “good” solution much faster than optimal solution.

- Can save a lot of time if we stop once we are “close” to optimal.
- How do we know we are close to optimal?



Duality II

Every LP has a dual:

Primal	Dual
$\max_x c^\intercal x$	$\min_\lambda b^\intercal \lambda$
$Ax \leq b$	$\lambda^\intercal A \geq c$
$x \geq 0$	$\lambda \geq 0$

Duality III

Weak duality:

- Let x be feasible solution to primal
- Let λ be feasible solution to dual.
- Then $c^\intercal x \leq b^\intercal \lambda$.
- Proof: look at $\lambda^\intercal A x$.
  - $(\lambda^\intercal A)x \geq c^\intercal x$ since $x \geq 0$ and $\lambda^\intercal A \geq c$.
  - $\lambda^\intercal (Ax) \leq b^\intercal \lambda$ since $\lambda \geq 0$ and $Ax \leq b$.

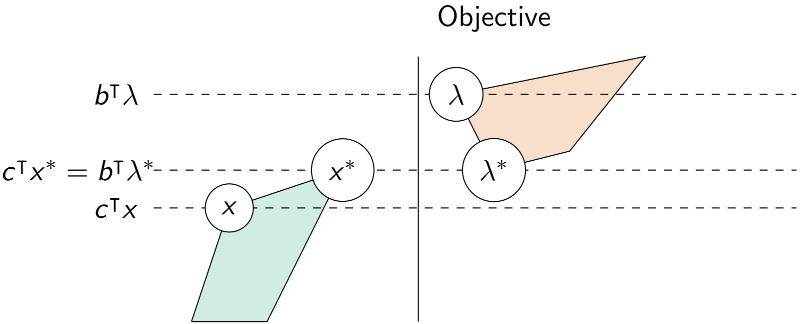
Duality IV

Strong duality:

- If either the primal or the dual has an optimal solution, then both problems do and the objective values are equal.

Duality V

Visualization:



Duality VI

Implications:

- If objective of dual solution is close to that of primal solution, then both solutions are almost optimal.
- If objectives equal, then both solutions are optimal.
- Simplex and interior point methods produce optimal primal and dual solutions.
- Interior point methods typically maintain feasible primal and dual solutions in every iteration.
- Simplex does not produce a feasible dual solution until iteration in which optimal solution is found.

Duality VII

Example primal problem:

$$\max \quad 4x + 5y + z$$

s.t.

$$\begin{aligned} 2x + 3y &\leq 60 \\ x + 2y &\leq 20 \\ y + z &\leq 10 \\ x + y + z &\leq 15 \\ x, y, z &\geq 0 \end{aligned}$$

Duality VIII

Example dual problem:

$$\min \quad 60\lambda_1 + 20\lambda_2 + 10\lambda_3 + 15\lambda_4$$

s.t.

$$\begin{aligned} 2\lambda_1 + \lambda_2 + \lambda_4 &\geq 4 && [x] \\ 3\lambda_1 + 2\lambda_2 + \lambda_3 + \lambda_4 &\geq 5 && [y] \\ \lambda_3 + \lambda_4 &\geq 1 && [z] \\ \lambda &\geq 0 \end{aligned}$$

Duality: Beyond LPs

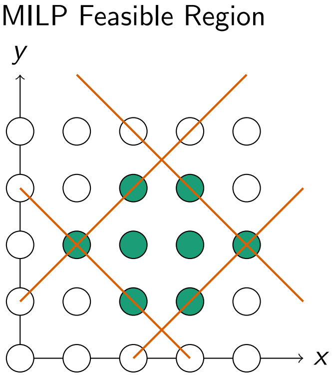
- Many types of optimization problems admit versions of duality, both strong and weak.
- Many convex optimization problems admit strong duality (see *Convex Optimization*, Boyd and Vandenberghe)
- Integer linear programs admit weak duality (see *Integer Programming*, Wolsey)
- Integer linear programs can also admit strong duality, but this is more complicated and less frequently used than in LPs (e.g. Williams 1996. "Duality in mathematics and linear and integer programming." Journal of Optimization Theory and Applications. 90:2, 257-278)

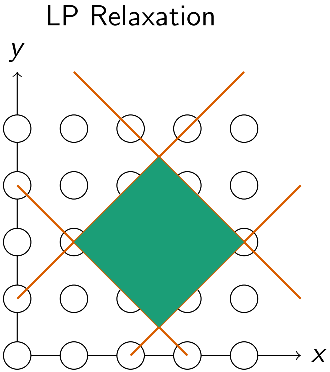
Solving IPs I

How do IP solvers work?

- Start with LP relaxation.
- LP relaxation replaces integer variables with continuous variables.

Solving IPs II





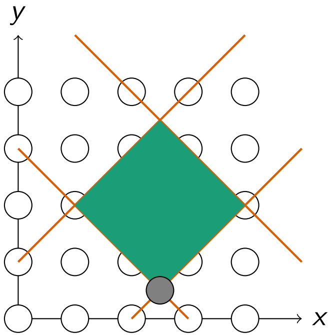
Solving IPs III

Key facts:

- For minimization problem, optimal value of LP relaxation is a lower bound.
- For maximization problem, optimal value of LP relaxation is an upper bound.
- If optimal solution to LP relaxation is integral, then it is also an optimal solution to the IP

Solving IPs IV

Get solution to LP relaxation:



It's not integral. Now what?

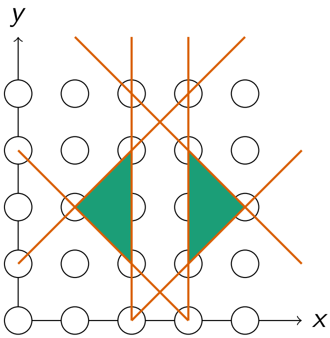
Branch-and-bound I

Option A: Branch.

- Pick a variable x with fractional value v.
- Divide feasible region into two smaller regions:
  1. $x \leq \lfloor v \rfloor$
  2. $x \geq \lceil v \rceil$
- Repeat procedure for smaller regions until solution is acquired for each region.
- See which solution is better.

Branch-and-bound II

Branching on x:



Note that in this case, both smaller regions have only integral points. Thus, simplex will produce integral values for those points.

Branch-and-bound III

This is usually arranged as a tree.

- The root node corresponds with the entire problem.
- Every time we branch, we create two children nodes.
- We will prune nodes that we can prove are suboptimal.

Branch-and-bound IV

Example: suppose that we have some IP with 5 variables with objective:

$$\max_x \sum_{i=1}^{5} x_i$$

The exact constraints will not be important for this example.

Branch-and-bound V

Our tree will start with a single node, corresponding to entire problem:

Root

We solve the LP relaxation, and we get the following solution:

x_1	x_2	x_3	x_4	x_5
10.5	0	11.5	5.5	3

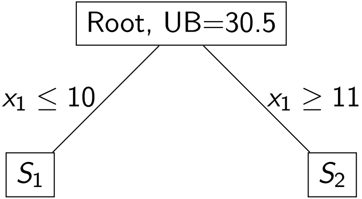
- The objective value is 30.5.
- The objective of the LP relaxation is an upper bound on the IP
- We associate this upper bound with the root node:

Root, UB=30.5

We choose a variable to branch on; let's choose x_1.

Branch-and-bound VI

We add two nodes, $x_1 \leq 10$ and $x_1 \geq 11$:



We now have a choice of which node to evaluate next. Let's choose S_1.

Branch-and-bound VII

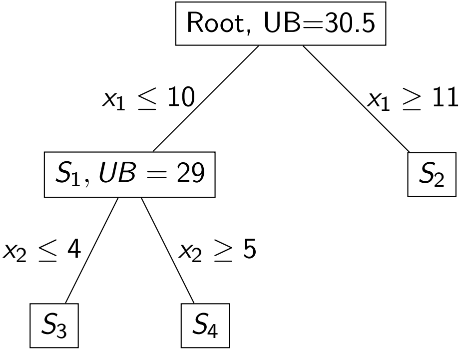
We solve the the LP relaxation of the subproblem with $x_1 \leq 10$, which gives us

x_1	x_2	x_3	x_4	x_5
10	4.5	6	6	2.5

- Again, use the objective value 29 as an upper bound associated with this node.
- Choose some variable to branch on; suppose we choose x_2.
- Add child nodes corresponding to $x_2 \leq 4$ and $x_2 \geq 5$.

Branch-and-bound VIII

We now have the tree:



We choose another node to evaluate; let's choose S_2.

Branch-and-bound IX

We solve the LP of the subproblem with $x_1 \geq 11$, which gives us:

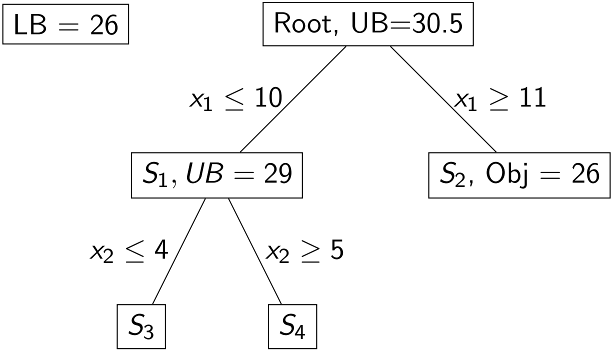
x_1	x_2	x_3	x_4	x_5
14	0	3	3	6

This is a feasible solution.

- We do not need to branch further from S_2.
- The value 26 is the optimal value of the subproblem S_2.
- A feasible solution gives us a lower bound for the entire IP; we store both this solution and the objective value.

Branch-and-bound X

We now have the tree:



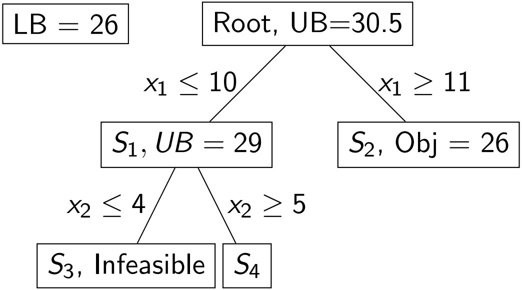
Let's evaluate the node S_3. Suppose that

Branch-and-bound XI

We solve the LP of the subproblem with $x_1 \leq 10$ and $x_2 \leq 4$.

- Suppose that this LP is infeasible.
- Then, we can simply remove this node.

This gives us the tree:



Let's evaluate the node S_4.

Branch-and-bound XII

We solve the LP relaxation of the subproblem with $x_1 \leq 10$ and $x_2 \geq 5$. Suppose that we get the following solution:

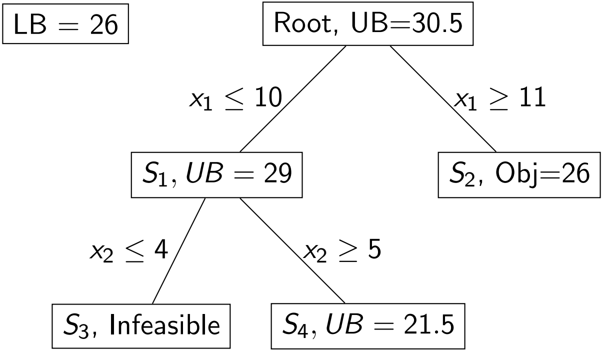
x_1	x_2	x_3	x_4	x_5
10	5	2.5	2.5	1.5

This has an objective value of 21.5

- This is lower than the current lower bound of 26
- We can be certain that further branching will not produce an optimal solution.
- We can prune this node.

Branch-and-bound XIII

The completed tree:



The optimal objective value is 26, which we found at node S_2.

Branch-and-bound XIV

In summary:

- Choose a node.
- Solve the LP.
- If the LP is infeasible, there is no need to branch.
- If the solution is integral, you have solved the subproblem and there is no need to branch.
- If the LP objective is higher than the current LB, then the subproblem is suboptimal and there is no need to branch.
- Otherwise, the LP has a fractional optimal solution; choose a variable and branch, creating two children.

Branch-and-bound Effectiveness I

How effective is branch-and-bound?

- Branch-and-bound is guaranteed to reach an optimal solution eventually.
- However, in the worst case, branch-and-bound may require enumerating every solution (or nearly every solution).
- In practice, branch-and-bound is very effective for many problems.
- For a while, branch-and-bound was the dominant method of solving integer programs.

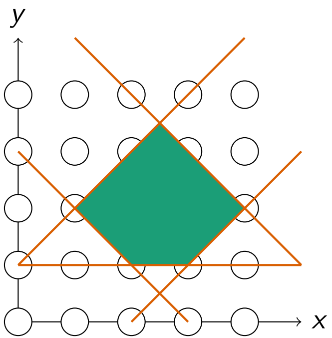
Cut Generation I

Option B: Cut generation

- Add a constraint that “cuts” the fractional point.
- Make sure not to cut any feasible integral solutions.

Cut Generation II

Cut generation:



Cut Generation III

There are many ways of generating cuts:

- Some types of cutting planes can be applied to any MILP problem.
- Some cutting planes apply only to specific constraints.

Effectiveness of cut generation

How effective are cut generation methods?

- Some cut generation methods are guaranteed to reach an optimal solution eventually.
- However, these methods tend to converge very slowly in practice
- It is difficult to solve practical problems purely with cut generation.

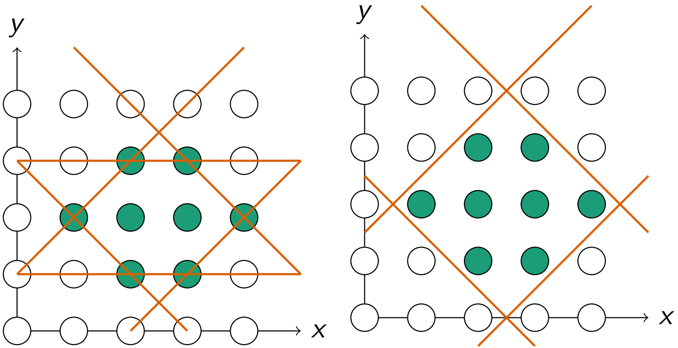
Branch-and-cut

Branch-and-bound can be combined with cutting plane algorithms:

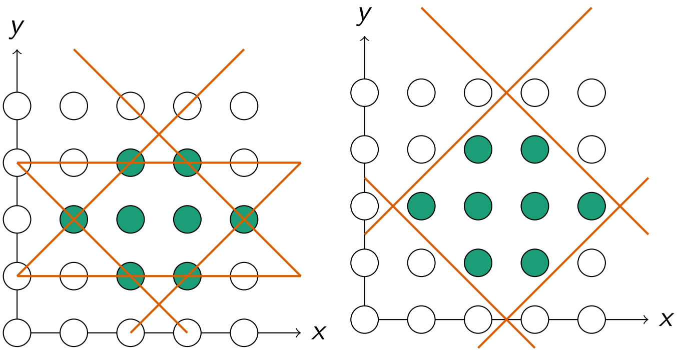
- At each node, we look to add some cutting planes before we branch.
- This is currently the method used by the most effective integer programming solvers.

Strong and weak formulations I

Consider same IP. Consider these formulations:



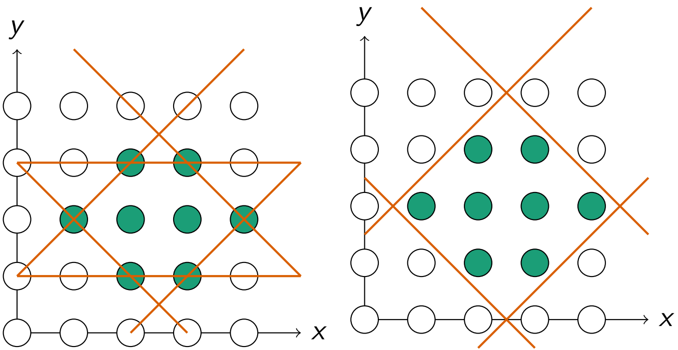
Strong and weak formulations II



Left formulation:

- Constraints very close to integral points.
- Less possible fractional solutions.
- Will likely require less branching and/or cut generation.

Strong and weak formulations III



Right formulation:

- Constraints not as close to integral points.
- More possible fractional solutions.
- Will likely require more branching and/or cut generation.

Strong and weak formulations IV

Definition of strength:

- If polyhedron associated with formulation A strictly contains the polyhedron associated with formulation B, then A is *weaker* than B and B is *stronger* than A

To show that a formulation B is stronger than A:

- Show that every feasible solution from LP relaxation of B is feasible for LP relaxation of A
- Show that LP relaxation A has some feasible solution that is infeasible for the LP relaxation of B.

Strong and weak formulations V

Example:

- We have a set with n items
- If we select one item from the set then we cannot select any other item from the set

Formulation A

$$x_i + x_j \leq 1 \text{ for all } i, j$$
$$x \geq 0$$

Formulation B:

$$\sum_{i=1}^{n} x_i \leq 1$$
$$x \geq 0$$

Strong and weak formulations VI

Formulation B is stronger than formulation A.

- If $\sum_{i=1}^{n} x_i \leq 1$ and $x \geq 0$ then $x_i + x_j \leq \sum_{i=1}^{n} x_i \leq 1$ for any i and j.
- So, every solution to LP relaxation of B is a solution to LP relaxation of A.
- The solution in which $x_i = 1/2$ for all i is feasible for LP relaxation of A but not for LP relaxation of B.

Strong and weak formulations VII

If every vertex of LP relaxation is integral, then you have found *convex hull of integral solutions.*

- Simplex produces optimal solution.
- This is the unique strongest formulation.

Strong and weak formulations VIII

Rule of thumb: look for as strong a formulation as possible!

- Caveat: sometimes stronger formulations require much more constraints, which can slow down solution.
- Usually, strength of formulation is more important than size of formulation.