Lecture 17: (Mixed) Integer Linear Programming

Sam McCauley

November 8, 2021

Williams College

Admin

- Assignment 6 due Wednesday; Mini-midterm 3 out Thursday
- MM3 will look very much like Assignment 6, but will also have integer-constrained problems
- I'll try to grade Assignment 6 very quickly so that you get feedback for MM3. (I'll grade out of order if necessary.)
- No class next Monday!
  - Chance to focus on MM3; reset a bit before part 4 of the course.

Plan for Today

- Wrap up last lecture
- What is an integer linear program?
- Lots of examples!
- Some information about how solvers work

Assignment 6 Questions?

- Problem 5: I really thought it was pretty clear when I did the problem but there are definitely a couple really ambiguous points
  - Plan: I'm writing a clearer version of the problem. I'll post the clarified version on the website after class. (I'll announce on slack but not email.)
  - Shouldn't change the contents of the problem but you should double-check it if you've already worked on it
- Other questions?

One more LP

- Let's solve a difficult optimization problem with our LP solver
- Idea: a middle school closed. Students from 6 different areas need to be assigned to the three other existing middle schools in the area. How can we do that?
- Need to assign all students; make sure students are reasonably well-balanced; minimized cost of transportation
- One problem: will wind up with fractional number of students assigned. How can we resolve this?
  - Round!
  - Does make our solution not optimal. But we'll discuss: by how much?

Setting up school problem

- Three schools with capacities 900, 1100, 1000
- Grades 6, 7, and 8
- Each grade assigned to a school must consist of between 30% and 36% of the school's total assignment. (Can't give one school all eighth graders.)
- Let's look at numbers in terms of what students are from each area, and how much it costs to get students from an area to a school.

School Problem Numbers

Area	6th	7th	8th	School 1	School 2	School 3
1	144	171	135	$300	0	$700
2	222	168	210	-	$400	$500
3	165	176	209	$600	$300	$200
4	98	140	112	$200	$500	-
5	195	170	135	0	-	$400
6	153	126	171	$500	$300	0

- Three schools with capacities 900, 1100, 1000
- Each grade assigned to a school must consist of between 30% and 36% of the school's total assignment. (Can't give one school all eighth graders.)
- minimize total cost

Our solution is fractional!

- What can we do?
- Round up or down; make sure constraints are still met
- How much can this affect our cost?
- Each school will probably end up with ≈ 1 student away from optimal. Unlikely to be more than \$1000 or so off.
- When is this strategy not a good idea?
  - When rounding changes the solution by a larger amount

Definitions

- Integer Linear Program (ILP): has linear constraints and objectives, but all variables are required to be *integers*
- Mixed Integer Linear Program (MIP): linear constraints and objectives. Some variables are required to be integers, some variables are continuous

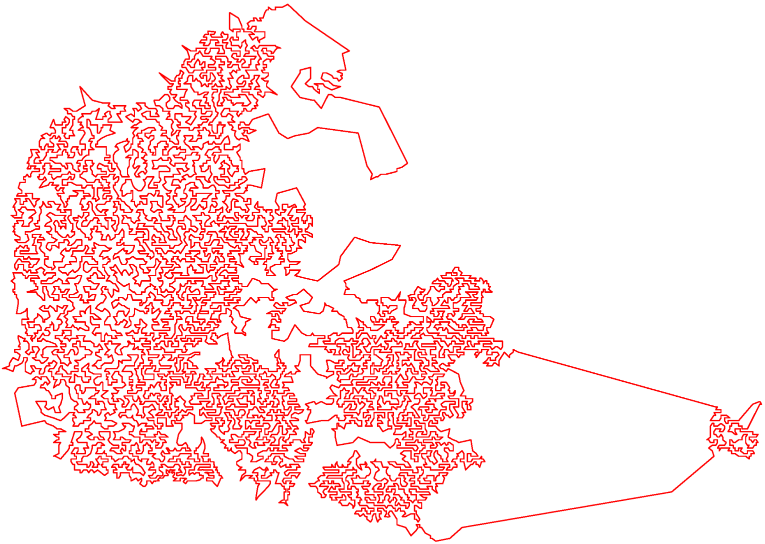
Why it's Useful

- Benefits from some structure (not as much as LP)
- Efficient solvers in practice
- Extremely widely applicable

Some good and bad news

- Solving an ILP or an MIP is NP-hard
- Bad news: can't guarantee to solve an ILP efficiently
- Good news: if an ILP solver tends to be efficient in practice, we can solve NP hard problems
- Can guarantee optimal solutions!

Travelling Salesman



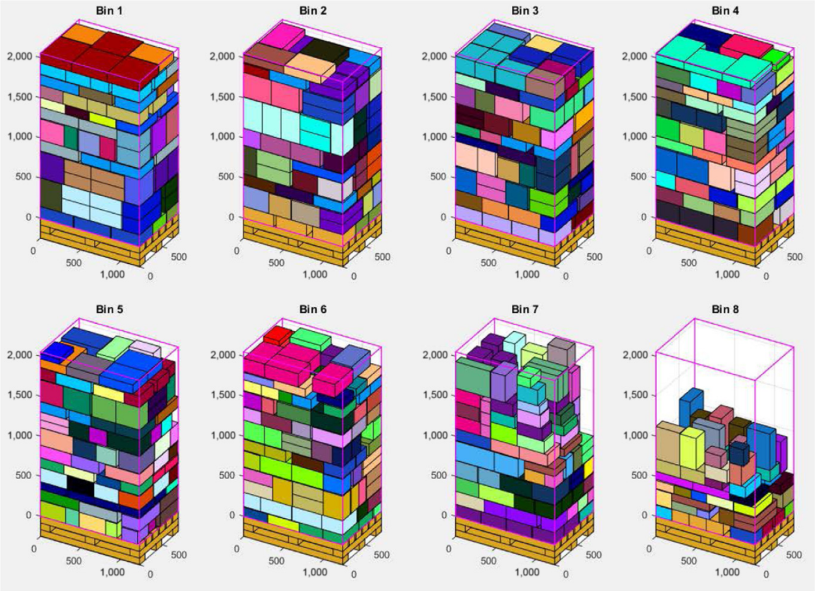
This is an *optimal* TSP instance with tens of thousands of points.

(Literally) Packaging Items

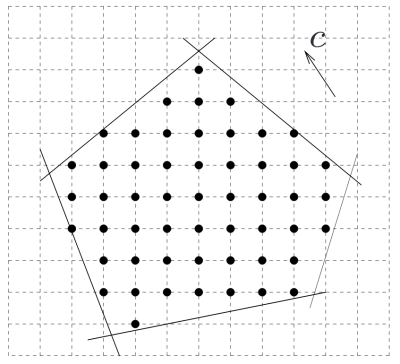
- Pack items onto pallets (bins)
- 3 dimensional
- May not have integral sizes
- Other constraints for what goes on what

Packaging Items

From Elhedhli, Gzara, and Yildiz 2019:



Visualization of an ILP



(Figure from L. Vandenberghe)

- Still a polytope given by inequalities
- But now, we're restricted to integer grid points

ILP and MIP Examples

Simple MIP

Diet problem from last lecture, but peanuts and rice only come in 100g bags. Chicken we may order as many grams as we want.

Objective: $\min 1.61p + .79r + .7c$

This means p and r are integers.

$$25.8p + 2.5r + 13.5c \geq 46$$
$$16.1p + 28.7r \geq 130$$
$$p \geq 0, r \geq 0, c \geq 0$$
$$p, r \in \mathbb{Z}$$

Using GLPK for ILP and MIP

- after “bounds” section (or after “constraints” section if no bounds)
- can write `general`, `integer`, or `binary`
- Then list variables of that type. (Binary variables must be 0 or 1, general are just normal LP variables)
- Default: general

Two Towers!

- Get a list of heights (let's forget about taking square roots—it's OK if the heights are not integers) $h_1, \ldots h_n$

- Want to divide into two towers T_1 and T_2 to minimize $|\sum_{i \in T_1} h_i - \sum_{j \in T_2} h_j|$.

Two Towers as an ILP

- Idea: build the smaller tower, make it as large as possible (but less than half total height)
- Variables $x_1, \ldots, x_n$. We have $x_i \in \{0,1\}$ for all i. Goal: $x_i = 1$ if i is in the smaller tower
- Objective: $\max \sum_{i=1}^{n} x_i h_i$
- Constraints:

$$\sum_{i=1}^{n} x_i h_i \leq \frac{1}{2} \sum_{i=1}^{n} h_i$$

$$x_i \in \{0,1\} \text{ for all } i = 1, \ldots, n$$

Why does this work?

- Every x_i is 0 or 1
- The total height of all items i with $x_i = 1$ is less than half the height (so it's the smaller tower), and is as large as possible
- So every assignment of 0 and 1 to x_i corresponds to a two tower solution. The ILP solution picks the best one.

First attempt: rounding

- Can we solve this as an LP and then round the solution?

- No! LP is trivially solvable with all but one variable being an integer.

How is GLPK going to do on two towers?

- Who thinks it will do better? Worse? (GLPK uses a direct ILP solver; does not use rounding)
- Answer: it takes a long time ($\approx$4.5 minutes) and gives a pretty bad solution
- Appear to be some precision issues.
- We'll come back to some ideas about why it doesn't do well in a minute

Doctor Assignments

- Let's say we have n doctors and n hospitals
- Want to match doctors to hospitals
- Doctor i lives distance $d_{i,j}$ from hospital j
- Goal: match doctors with hospitals to minimize total driving distance
- (Other methods? Yes, but this generalizes easily)

Doctor Assignments: ILP

- What should our variables be?
- $x_{i,j} = 1$ if doctor i is assigned to hospital j, $x_{i,j} = 0$ otherwise
- Constraints?
    - $x_{i,j} \in \{0, 1\}$
    - For all i: $\sum_{j=1}^{n} x_{i,j} = 1$ (every doctor has one hospital)
    - For all j: $\sum_{i=1}^{n} x_{i,j} = 1$ (every hospital has one doctor)

Doctor Assignments: ILP

Constraints:

- $x_{i,j} \in \{0, 1\}$
- For all i: $\sum_{j=1}^{n} x_{i,j} = 1$
- For all j: $\sum_{i=1}^{n} x_{i,j} = 1$

Objective? (Recall: goal is minimize total distance)

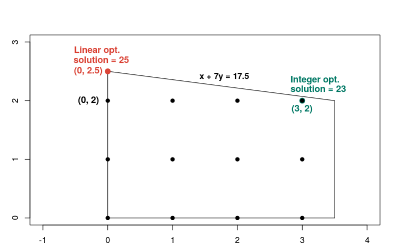
- $\min \sum_{i=1}^{n} \sum_{j=1}^{n} x_{i,j} d_{i,j}$

Solving MIPs

First thought: Can we Round?

- *LP relaxation:* just remove the integer constraints
- $e_{i,j} \in \{0, 1\}$ becomes $e_{i,j} \geq 0$ and $e_{i,j} \leq 1$.
- How badly can this do?

Rounding MIPs

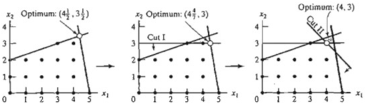


From Google OR Tools Documentation

- Can do arbitrarily badly, even for simple ILPs
- May work effectively if the problem has a special structure that makes rounding effective
- Example: the diet problem is probably solved fairly well by rounding (will only be off by 1 unit of each food)

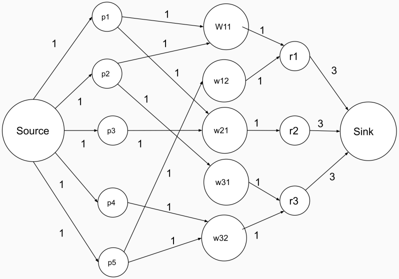
Second Method: Cutting ILPs

FIGURE 9.10
Illustration of the use of cuts in ILP



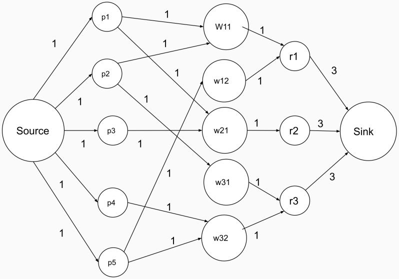
- Cut the LP without removing integer solutions
- After enough cuts, can round and get a good solution!
- Not always possible, but surprisingly effective methods in practice for some types of problem
- Many MIP solvers find these cuts for you

Third Method: Prove the LP has integral soln



- Broad class of LPs are guaranteed to give optimal solutions
- We won't go over in this class except on this slide!
- Example for linear algebra people: if your constraint matrix is totally unimodular then there exists an optimal integer solution

Third Method: Prove the LP has integral soln



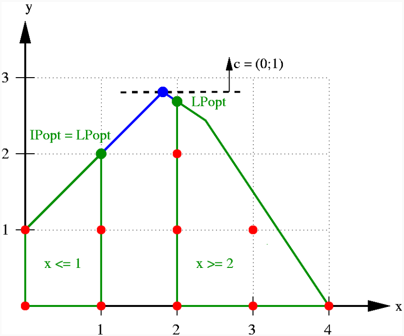
- Broad class of LPs are guaranteed to give optimal solutions
- We won't go over in this class except on this slide!
- Example for flow-reduction-lovers: if you write a flow problems as an LP where all constraints are integers, there exists an optimal integer solution

Main MIP Solving Method: Branch and Bound

Branch and Bound

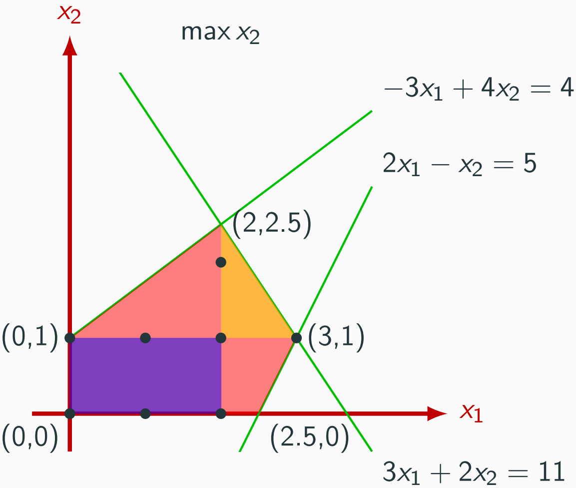
- Two towers: really wanted to "rule out" some of the search space
- Maintain worst-case guarantees
- This is the idea behind branch and bound
- This is a large *class* of algorithms; I'm giving a high level description of the idea

Branching



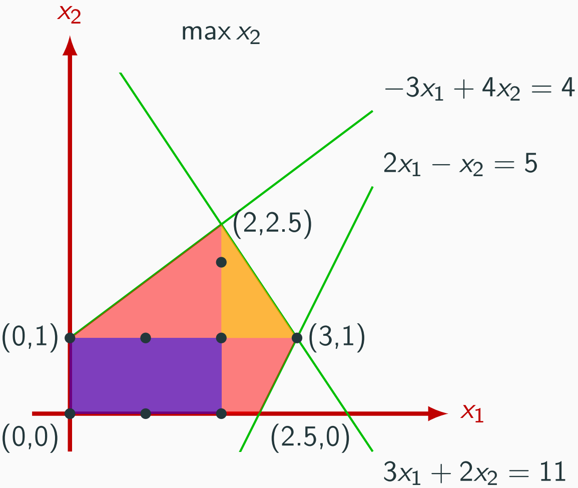
- First, we divide the problem into several subproblems
- Visualization is useful: just partition the feasible region into several pieces
- So far, still need to search through all of them (same as brute force)

Branching and Bounding



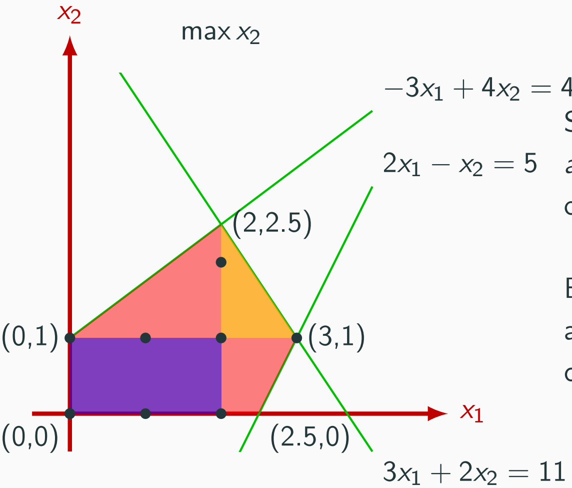
- Partition region
- Find best solution in orange piece
- When can we avoid searching in purple?

Branching and Bounding



- Upper bound best solution in purple
- If best possible soln in purple is worse than best soln in orange, can skip

Branching and Bounding



Safe to skip: *always* gives optimal solution.

But, can't skip anything in worst case.

What do we need?

- Way to get a good solution in orange region: recurse!
- Or: can just do a simple greedy method.
- Way to upper bound best solution in purple region??
  - Relax to an LP! Might not give a good upper bound, but will give *an* upper bound (Recall: LPs are relatively fast to solve)
  - Duality can help

Branch and Bound Intuition

- Let us rule out big parts of the polytope
- "Everything in here has a bad objective function, so we can skip it." (This is the *bound* part)
- Many practical problems have large parts that are easy to skip. (If we're stacking groceries on pallets, no need to spend time looking at solutions with bread on the bottom.)
- The more we branch (find good solutions), the more we can bound (rule out parts of the search space whose solutions are suboptimal)

Branch and Bound in Practice

- Advanced methods to figure out what parts of the polytope to search, and how accurately to bound them
- The better your choices, the more you can rule out
- Other methods (greedy, LP cuts, duality, heuristic search, etc.) can be integrated into this method

Branch and Bound in Practice

- Solvers are generally optimized for a given problem
- Dedicated solvers for TSP, Knapsack, that make branching decisions and use bounding methods particularly effective for that problem
- This is how you get the optimal, giant TSP tours
- Also some general-purpose solvers

Branch and Bound Summary

- Always gives an optimal solution
- May not find it quickly on tricky problems
- Two Towers performance was not great...any ideas why that is?

Solvers

These solvers have both LP and MIP solvers (using different algorithms):

- GLPK (simplex, branch and bound). Open source. Standalone program is fairly easy to use; can also access from C.
- CPLEX - IBM software for MIPs. Old but reliable. Proprietary. Effective, but can be difficult to work with
- COIN-OR - open source solver
- Google OR tools - wrapper for COIN-OR. Has a really nice TSP and Knapsack solvers. More user friendly