Lecture 18: (Mixed) Integer Linear Programming Cont.

Sam McCauley

November 12, 2021

Williams College

Admin

- How was Assignment 6?
- MM3 written; out once I do some testing (should be soon)
- No TA office hours this coming week
- No class Monday! Extra office hours instead (in TCL 306)

Upcoming schedule

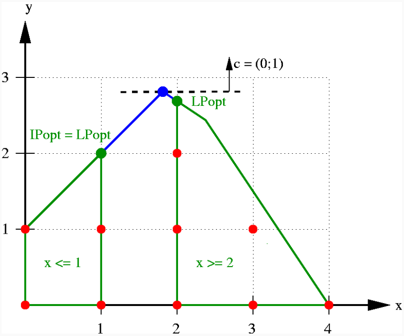
- Current plan: Assignment 7 due *Tuesday* before Thanksgiving
- Assignment 7 is back to the usual: half coding in C, half problem set questions
- Idea would be that it's significantly shorter (in terms of time spent) than most other assignments
- Is that difficult with your plans?

Plan for Today

- Wrap up branch and bound
- More mixed/integer linear programming examples!

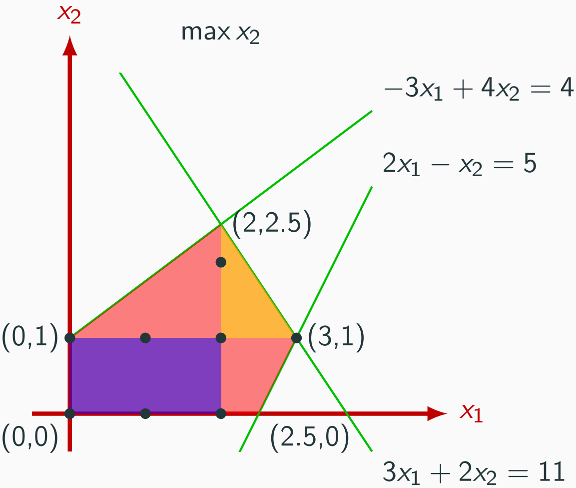
Main MIP Solving Method: Branch and Bound

Branching



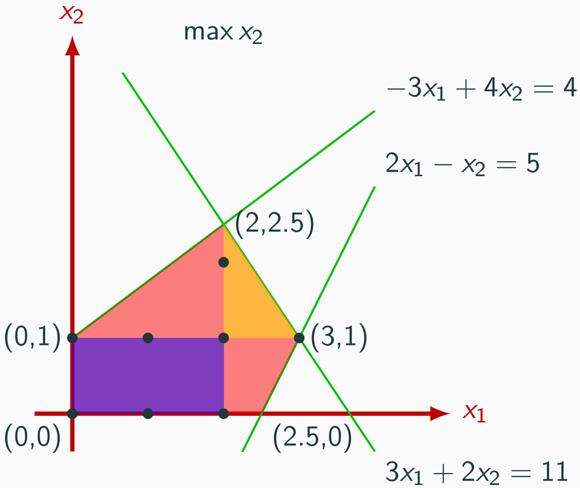
- First, we divide the problem into several subproblems
- Visualization is useful: just partition the feasible region into several pieces
- So far, still need to search through all of them (same as brute force)

Branching and Bounding



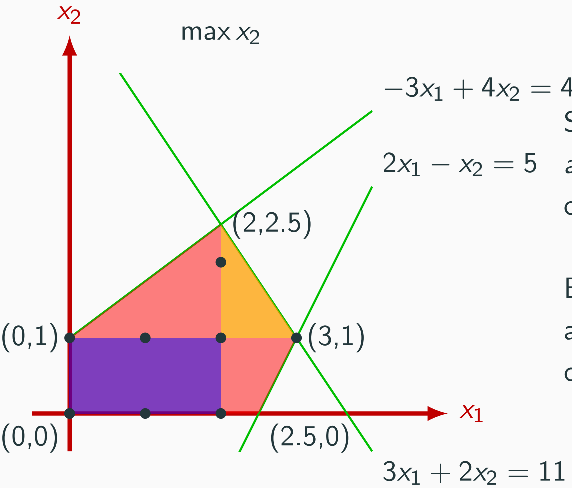
- Partition region
- Find best solution in orange piece
- When can we avoid searching in purple?

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, and come back to refine the solution once we've ruled some others out.
- 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)
    - (Outside scope of class) Duality can help

Branch and Bound Intuition

- Let us rule out big parts of the polytope (that is to say: lets us avoid searching massive numbers of potential solutions.)
- "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 using GLPK...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 than CPLEX or COIN-OR.

More ILP and MIP Examples

Scheduling

- (Aside: scheduling is a major application of ILPs. Lots of different techniques; this is just one example.)
- Assign n unit-cost jobs to machines.
- Each job j_i has a type t_i. Two jobs of the same type cannot be assigned to the same machine.
- How can we schedule the jobs with the minimum number of machines?

Scheduling Jobs with Types

- n jobs, job i has type t_i
- Two jobs of same type cannot be assigned to the same machine
- Min number of machines

- What variables do we want?
- Probably: keep track of what job is assigned to what machine
- $s_{i,m} = 1$ if job i is assigned to machine m
- How many machines do we need?
- At most n. So have n^2 variables: $s_{i,m} \in \{0, 1\}$, for $1 \leq i \leq n$ and $1 \leq m \leq n$.

Scheduling Jobs with Types

- n jobs, job i has type t_i
- Two jobs of same type cannot be assigned to the same machine
- Min number of machines
- $s_{i,m} = 1$ if job i assigned to machine m

- Constraints?
- Want every job assigned to exactly one machine
- For all $1 \leq i \leq n$, $\sum_{m=1}^{n} s_{i,m} = 1$

Scheduling Jobs with Types

- n jobs, job i has type t_i
- Two jobs of same type cannot be assigned to the same machine
- Min number of machines
- $s_{i,m} = 1$ if job i assigned to machine m

- Constraints?
- Two jobs of the same type can't be assigned to the same machine
- Rephrased: for every machine m, no two jobs of the same type can be assigned to m

Scheduling Jobs with Types

- n jobs, job i has type t_i
- Two jobs of same type cannot be assigned to the same machine
- Min number of machines
- $s_{i,m} = 1$ if job i assigned to machine m

- Constraints?
- For every machine i, no two jobs of the same type can be assigned to i
- For all $1 \leq m \leq n$,for all jobs i_1 and i_2 with the same type $t_{i_1} = t_{i_2}$, $s_{i_1,m} + s_{i_2,m} \leq 1$
- (Up to n^3 constraints. Also: constraints depend on the input.)

Scheduling Jobs with Types

- n jobs, job i has type t_i
- Two jobs of same type cannot be assigned to the same machine
- Min number of machines
- $s_{i,m} = 1$ if job i assigned to machine m

- Objective?
- Let c_m be the cost of machine m. Want $c_m = 1$ if there is a job assigned to machine i, $c_m = 0$ otherwise.
- $\min \sum_{m=1}^{n} c_m$
- Constraint for c_m?
- For all jobs i and all machines m, $c_m \geq s_{i,m}$

Scheduling Jobs with Types

Objective: $\min \sum_{m=1}^{n} c_m$

Constraints:

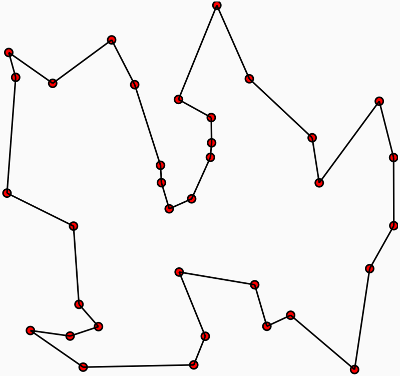
$c_m \geq s_{i,m}$

For all $1 \leq m \leq n$,for all jobs i_1 and i_2 with the same type $t_{i_1} = t_{i_2}$,
$s_{i_1,m} + s_{i_2,m} \leq 1$

For all $1 \leq i \leq n$, $\sum_{m=1}^{n} s_{i,m} = 1$

$s_{i,m} \in \{0, 1\}$ for all $1 \leq i \leq n$, $1 \leq m \leq n$.

Travelling Salesman



- Find minimum-length cycle through vertices such that each is visited exactly once
- Given: set of n points, for each pair of points i and j the cost $c_{i,j}$ to get from i to j. Have $c_{j,i} = c_{i,j}$

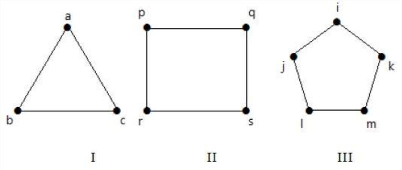
Travelling Salesman

- Variables?
- $e_{i,j} = 1$ if the TSP tour has an edge from point i to point j
- $e_{i,j} \in \{0, 1\}$ for $1 \leq i \leq n$ and $1 \leq j \leq n$.
- Objective?
- $\sum_{i=1}^{n} \sum_{j=1}^{n} e_{i,j} c_{i,j}$

Travelling Salesman

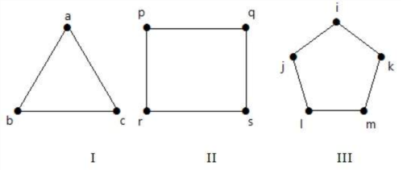
- Constraints?
- Need to ensure that the edges with $e_{i,j} = 1$ form a cycle through all points
- Observation: in a cycle, all points have one edge coming in, and one edge going out
- For all i, $\sum_{j \neq i} e_{i,j} = 1$ and $\sum_{\ell \neq i} e_{\ell,i} = 1$
- Is this sufficient?

Travelling Salesman



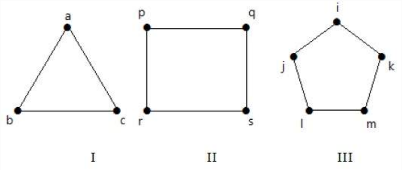
- Unfortunately, no—one in/one out just means a set of cycles.
- Can we give another constraint to fix this?
- Yes, but it's nontrivial:
- Add $n-1$ new variables u_i (for $i = 2, \ldots, n$)
- $u_i - u_j + ne_{i,j} \leq n-1$ for $2 \leq i \neq j \leq n$, and
- $1 \leq u_i \leq n-1$ for $2 \leq i \leq n$

Travelling Salesman



- $u_i - u_j + ne_{i,j} \leq n - 1$ for $2 \leq i \neq j \leq n$, and
- $1 \leq u_i \leq n - 1$ for $2 \leq i \leq n$
- single cycle → LP solution:
- If we have a simple cycle visiting every vertex, can we create an assignment that satisfies the constraints?
- Yes: if i is the kth visited city (after city 1), set $u_i = k$

Travelling Salesman (High Level)

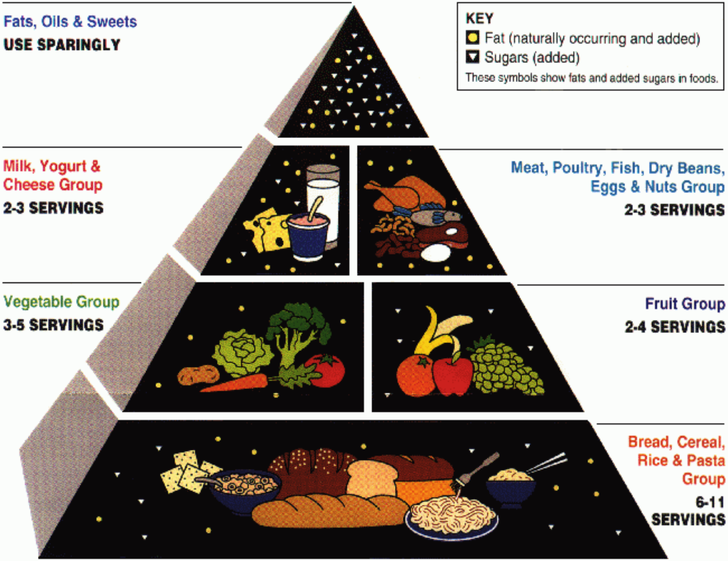


- $u_i - u_j + ne_{i,j} \leq n - 1$ for $2 \leq i \neq j \leq n$, and
- $1 \leq u_i \leq n - 1$ for $2 \leq i \leq n$
- LP solution $\rightarrow$ single cycle:
- Sum the above inequalities for any k-length cycle not including city 1

One last example

- Idea here: we talked about how LPs can only really "AND" constraints
- With ILP and MIP, can do something much more like "OR":
  - One of these constraints must be satisfied, or
  - Pick one of these items (in an assignment)
- Simple example: optimal eating while being able to choose your diet

Food Pyramid



Choice of diet

- You need to satisfy one of the three following diet goals:
  - 46 grams of protein and 130 grams of carbs every day; or
  - 20 grams of protein and 200 grams of carbs every day; or
  - 100 grams of protein and 30 grams of carbs every day
- 100g Peanuts: 25.8g of protein, 16.1g carbs, $1.61
- 100g Rice: 2.5g protein, 28.7g carbs, $.79
- 100g Chicken: 13.5g protein, 0g carbs, $.70

What is the cheapest way you can hit one of these diet goals?

MIP for Choice of Diet

- How to encode which diet I choose?
- $x_1 = 1$ if I choose the first diet; $x_2 = 1$ if I choosed the second diet; $x_3 = 1$ if I choose the third diet
- Make sure I choose exactly one diet?
- $x_i \in \{0, 1\}$
- $x_1 + x_2 + x_3 = 1$

MIP for Choice of Diet

- You need to satisfy one of the three following diet goals:
  - 46 grams of protein and 130 grams of carbs every day; or
  - 20 grams of protein and 200 grams of carbs every day; or
  - 100 grams of protein and 30 grams of carbs every day
- How can I encode this?
- Previously: $25.8p + 2.5r + 13.5c \geq 46 \ldots$
- Hint: if $x_1 = 0$, I want to do something to these constraint so that they're *always* satisfied
- $25.8p + 2.5r + 13.5c + 46(1 - x_1) \geq 46$

Choice of diet LP

- Diet options:
  - 46 g protein; 130 g carbs; or
  - 20 g protein; 200 g carbs; or
  - 100 g protein; 30 g carbs
- 100g Peanuts: 25.8g protein, 16.1g carbs, \$1.61
- 100g Rice: 2.5g protein, 28.7g carbs, \$.79
- 100g Chicken: 13.5g protein, 0g carbs, \$.70

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

- $25.8p + 2.5r + 13.5c + 46(1 - x_1) \geq 46;$
- $16.1p + 28.7r + 130(1 - x_1) \geq 130$
- $25.8p + 2.5r + 13.5c + 20(1 - x_2) \geq 20;$
- $16.1p + 28.7r + 200(1 - x_2) \geq 200$
- $25.8p + 2.5r + 13.5c + 100(1 - x_3) \geq 100;$
- $16.1p + 28.7r + 30(1 - x_2) \geq 30$
- $x_1 + x_2 + x_3 = 1$
- $p, r, c \geq 0$; $p, r \in \mathbb{Z}$; $x_i \in \{0, 1\}$

Technique summary

- When want to choose one of several constraints to satisfy:
- multiply the indicator variable for whether or not you choose by a large enough constant to make the constraint trivial
- Need to be able to bound the constraint to do this!
- What happens with rounding when you use this technique?

Conclusion

Takeaways

- What is an ILP/MIP?
- When do you need an ILP/MIP? When does an LP suffice?
- Using ILP/MIPs to describe a computational problem
- Some ILP/MIP techniques
- Branch and bound: how can a heuristic give you an optimal solution?