Integer Programming
ISE 418

Lecture 1

Dr. Ted Ralphs

Reading for This Lecture

- N&W Sections I.1.1-I.1.4
- Wolsey Chapter 1
- CCZ Chapters 1-2

What is mathematical optimization?

Mathematical Optimization

- *Mathematical optimization* is a formal language for describing and analyzing (optimization) problems.
- The essential elements of an optimization problem are
  - a system whose operating state can be specified numerically by specifying the values of certain *variables*;
  - a set of states considered *feasible* for the given system that are contained in a set we can describe; and
  - an *objective function* that defines a preference ordering of the states.
- Before applying mathematical optimization techniques, we must first create a *model*, which is then translated into a particular *formulation*.
- The formulation is a formal description of the problem in terms of mathematical functions and logical operators.
- The use of mathematical optimization as a language imposes constraints on how the system can be modeled.
- We often need to make simplifying assumptions and approximations in order to put the problem into the required form.
- Nevertheless, mathematical optimization is a *very* general language.

Modeling

- Our overall goal is to develop a *model* of a real-world system in order to analyze the system.
- The system we are modeling is typically (but not always) one we are seeking to control by determining its "operating state."
- The (independent) variables in our model represent aspects of the system we have control over.
- The values that these variables take in the model tell us how to set the operating state of the system in the real world.
- *Modeling* is the process of creating a conceptual model of the real-world system.
- *Formulation* is the process of constructing a mathematical optimization problem whose solution reveals the optimal state according to the model.
- This is far from an exact science.

The Problem Solving Process

- The process solving the original problem consists generally of the following steps.
  - Model: Determine the "real-world" state variables, system constraints, and goal(s) or objective(s) for operating the system.
  - Formulate: Translate these variables and constraints into the form of a mathematical optimization problem (the "formulation").
  - Solve: Solve the mathematical optimization problem.
  - Interpret: Interpret the solution in terms of the real-world system.
- This process presents many challenges.
  - Simplifications may be required in order to ensure the eventual mathematical optimization problem is "tractable."
  - The mappings from the real-world system to the model and back are sometimes not very obvious.
  - Variables that don't appear in the conceptual model may be needed to enforce logical conditions or simplify the form of the constraints.
  - There may be more than one valid "formulation."
- All in all, an intimate knowledge of mathematical optimization definitely helps during the modeling process.

Example: Sudoku

Challenge: Fill in the grid squares with numbers 0-9 such that

- All squares in the same column have different values, and
- All squares in the same row have different values.

	2			3			4	
6								3
		4				5		
			8		6			
8				1				6
			7		5			
		7				6		
4								8
	3			4			2	

- What should the decision variable be?
- What are the constraints?

Mathematical Optimization Problems

Elements of the model:

- Decision variables: a vector of variables indexed 1 to n.
- Constraints: pairs of functions and right-hand sides indexed 1 to m.
- Objective Function
- Parameters and Data

The general form of a *mathematical optimization problem* is:

$$\begin{aligned} z_{\mathrm{MP}} = \sup \quad & f(x) \\ \text{s.t.} \quad & g_i(x) \le b_i, \ \ 1 \le i \le m \\ & x \in \mathbb{Z}^p \times \mathbb{R}^{n-p} \end{aligned} \tag{MP}$$

Note the use supremum here because the maximum may not exist.

Feasible Region

- The *feasible region* of (MP) is

$$\mathcal{F} = \{x \in \mathbb{Z}^p \times \mathbb{R}^{n-p} \mid g_i(x) \leq b_i, \;\; 1 \leq i \leq m\}$$

- The feasible region is *bounded* when

$$\mathcal{F} \subseteq \{x \in \mathbb{R}^m \mid \|x\|_1 \leq M\}$$

  and *unbounded* otherwise.

- We take $z_{\mathrm{MP}} = -\infty$ when $\mathcal{F} = \emptyset$ and say the problem is *infeasible* in this case.

- We may also have $z_{\mathrm{MP}} = \infty$ when the problem is *unbounded*, e.g., f is a linear function and $\exists \hat{x} \in \mathcal{F}$ and $d \in \mathbb{R}^n$ such that
  - $x + \lambda d \in \mathcal{F}$ for all $\lambda \in \mathbb{R}_+$,
  - $f(d) > 0$.

- Note that there is a difference between the *feasible region* being unbounded and the *problem* being unbounded.

Solutions

- A *solution* is an assignment of values to variables.
- A solution can hence be thought of as an n-dimensional vector.
- A *feasible solution* is an assignment of values to variables such that all the constraints are satisfied, i.e., a member of $\mathcal{F}$.
- The *objective function value* of a solution is obtained by evaluating the objective function at the given point.
- An *optimal solution* (assuming maximization) is one whose objective function value is greater than or equal to that of all other feasible solutions.
- Note that a mathematical optimization problem may not have an optimal solution.
- Question: What are the different ways in which this can happen?

Possible Outcomes

- When we say we are going to "solve" a mathematical optimization problem, we mean to determine
  - whether it has an optimal *value* (meaning z_{MP} is finite), and
  - whether it has an optimal *solution* (the supremum can be attained).
- Note that the supremum may not be attainable if, e.g., $\mathcal{F}$ is an open set.
- We may also want to know some other things, such as the status of its "dual" or about sensitivity.

Types of Mathematical Optimization Problems

- The type of a mathematical optimization problem is determined primarily by
  - The form of the objective and the constraints.
  - Whether there are integer variables or not.
- In 406, you learned about linear models.
  - The objective function is linear.
  - The constraints are linear.
- The most important determinants of whether a mathematical optimization problem is "tractable" are the convexity of
  - The objective function.
  - The feasible region.

Types of Mathematical Optimization Problems (cont'd)

- Mathematical optimization problems are generally classified according to the following dichotomies.
  - Linear/nonlinear
  - Convex/nonconvex
  - Discrete/continuous
  - Stochastic/deterministic
- See the NEOS guide for a more detailed breakdown.
- This class concerns (primarily) models that are discrete, linear, and deterministic (and as a result generally non-convex)

The Formal Setting for This Course

- We consider linear optimization problems in which we additionally impose that $x \in \mathbb{Z}^p_+ \times \mathbb{R}^{n-p}_+$.
- The general form of such a mathematical optimization problem is

$$z_{\mathsf{IP}} = \max\{c^\top x \mid x \in \mathcal{S}\}, \tag{MILP}$$

where for $A \in \mathbb{Q}^{m \times n}, b \in \mathbb{Q}^m, c \in \mathbb{Q}^n$. we have

$$\mathcal{P} = \{x \in \mathbb{R}^n \mid Ax \leq b\} \tag{FEAS-LP}$$

$$\mathcal{S} = \mathcal{P} \cap (\mathbb{Z}^p_+ \times \mathbb{R}^{n-p}_+) \tag{FEAS-MIP}$$

- This type of optimization problem is called a *mixed integer linear optimization problem* (MILP).
- If $p = n$, then we have a *pure integer linear optimization problem*, or an *integer optimization problem* (IP).
- If $p = 0$, then we have a *linear optimization problem* (LP).
- The first p components of x are the *discrete* or *integer* variables and the remaining components consist of the *continuous* variables.

Conventions and Notation

If not otherwise stated, the following conventions will be followed for lecture slides during the course:

- A will denote a matrix of dimension m by n (rational).
- b will denote a vector of dimension m (rational).
- x will denote a vector of dimension n.
- c will denote a vector of dimension n (rational).
- p will be the number of integer variables.
- $\mathcal{P}$ will denote a polyhedron contained in $\mathbb{R}^n$, usually given in the form

$$\mathcal{P} = \{x \in \mathbb{R}^n \mid Ax \leq b\}$$

- $\mathcal{S}$ will be $\mathcal{P} \cap (\mathbb{Z}^p_+ \times \mathbb{R}^{n-p}_+)$.
- An integer program is then described fully by the quadruplet (A, b, c, p).
- Vectors will be column vectors unless otherwise noted.
- When taking the product of vectors, we will sometimes leave off the transpose.

Additional Notation

- The notation A_N will denote a submatrix formed by taking the columns indexed by set $N \subseteq \{1, \ldots, n\}$.
- We will sometimes use the notation $I = \{1, \ldots, p\}$ and $C = \{p + 1, \ldots, n\}$.
- Then A_C is a matrix formed by the columns of A corresponding to the continuous variables.
- Similarly, A_I is a matrix formed by the columns of A corresponding to the integer variables.
- The i^{th} column of A will be denoted A_i.
- The i^{th} row of A will be denoted a_i.

Special Case: Binary Integer Optimization

- In many cases, the variables of an IP represent yes/no decisions or logical relationships.
- These variables naturally take on values of 0 or 1.
- Such variables are called *binary*.
- IPs involving only binary variables are called *binary integer optimization problems* (BIPs) or $0-1$ *integer optimization problems* ($0-1$ IPs).

Combinatorial Optimization

- A *combinatorial optimization problem* $CP = (N, \mathcal{F})$ consists of
  - A finite *ground set* N,
  - A set $\mathcal{F} \subseteq 2^N$ of *feasible solutions*, and
  - A *cost function* $c \in \mathbb{Z}^n$.
- The *cost* of $F \in \mathcal{F}$ is $c(F) = \sum_{j \in F} c_j$.
- The combinatorial optimization problem is then

$$\max\{c(F) \mid F \in \mathcal{F}\}$$

- There is a natural association with a $0-1$ IP.
- Many COPs can be written as BIPs or MILPs.

Some Notes

- The form of the problem we consider will be maximization by default, since this is the standard in the reference texts.
- I normally think in terms of minimization by default, so please be aware that this may cause some confusion.
- Also note that the definition of $\mathcal{S}$ includes non-negativity, but the definition of $\mathcal{P}$ does not.
- One further assumption we will make is that the constraint matrix is rational. Why?

Some Notes

- The form of the problem we consider will be maximization by default, since this is the standard in the reference texts.
- I normally think in terms of minimization by default, so please be aware that this may cause some confusion.
- Also note that the definition of $\mathcal{S}$ includes non-negativity, but the definition of $\mathcal{P}$ does not.
- One further assumption we will make is that the constraint matrix is rational. Why?
  - This is an important assumption since with irrational data, certain "intuitive" results no longer hold (such as what?)
  - A computer can only understand rational data anyway, so this is not an unreasonable assumption.

How Difficult is MILP?

- Solving general integer MILPs can be much more difficult than solving LPs.
- There in no known *polynomial-time* algorithm for solving general MILPs.
- Solving the associated *LP relaxation*, an LP obtained by dropping the integerality restrictions, results in an upper bound on z_{IP}.
- Unfortunately, solving the *LP relaxation* may not tell us much.
  - Rounding to a feasible integer solution may be difficult.
  - The optimal solution to the LP relaxation can be arbitrarily far away from the optimal solution to the MILP.
  - Rounding may result in a solution far from optimal.

Discrete Optimization and Convexity

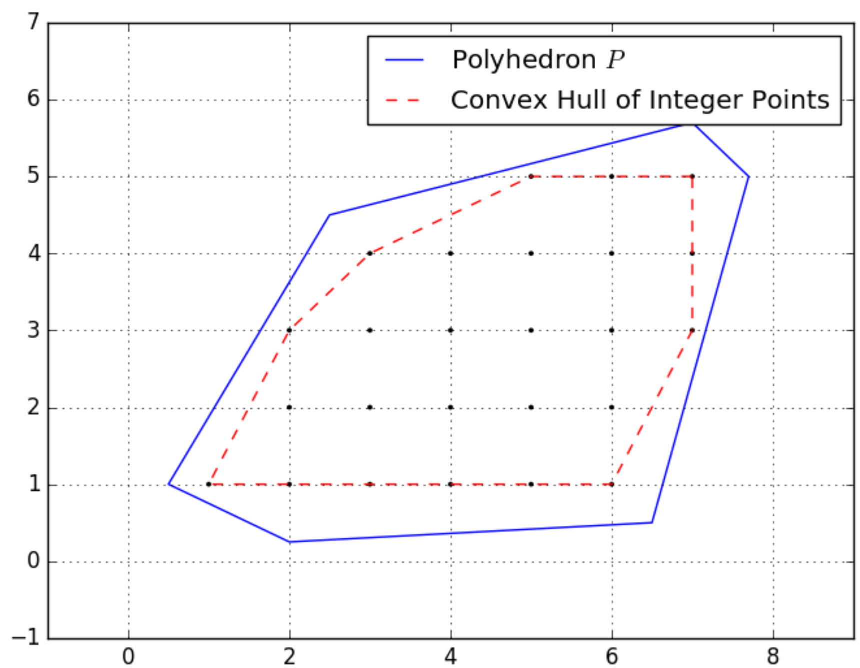
- One reason why convex problems are "easy" to solve is because convexity makes it easy to find *improving feasible directions*.
- Optimality criterion for a linear program are equivalent to "no improving feasible directions."
- The feasible region of an MILP is nonconvex and this makes it difficult to find feasible directions.
- The algorithms we use for LP can't easily be generalized.
- Although the feasible set is nonconvex, there is a convex set over which we can optimize in order to get a solution (why?).
- The challenge is that we do not know how to describe that set.
- Even if we knew the description, it would in general be too large to write down explicitly.
- Integer variables can be used to model other forms of nonconvexity, as we will see later on.

The Geometry of an MILP

- Let's consider again an integer optimization problem

$$\begin{aligned} \max \quad & c^\top x \\ \text{s.t.} \quad & Ax \leq b \\ & x \in \mathbb{Z}^n_+ \end{aligned}$$

- The feasible region is the integer points inside a polyhedron.



- Why does solving the LP relaxation not necessarily yield a good solution?

How General is Discrete Optimization?

- A natural question to ask is just how general this language for describing optimization problems is.
- Is this language general enough that we should spend time studying it?
- To answer this question rigorously requires some tools from an area of computer science called *complexity theory*.
- We can say informally, however, that the language of mathematical optimization is *very* general.
- One can show that almost anything a computer can do can be described as a mathematical optimization problem[1].
- Mixed integer linear optimization is not quite as general, but is complete for a broad class of problems called NP.
- We will study this class later in the course.

[1] Formally, mathematical optimization can be shown to be a "Turing-complete" language

Conjunction versus Disjunction

- A more general mathematical view that ties integer programming to logic is to think of integer variables as expressing *disjunction*.
- The constraints of a standard mathematical program are *conjunctive*.
  - All constraints must be satisfied.
  - In terms of logic, we have

$$g_1(x) \leq b_1 \text{ AND } g_2(x) \leq b_2 \text{ AND } \cdots \text{ AND } g_m(x) \leq b_m \tag{1}$$

  - This corresponds to *intersection* of the regions associated with each constraint.
- Integer variables introduce the possibility to model *disjunction*.
  - At least one constraint must be satisfied.
  - In terms of logic, we have

$$g_1(x) \leq b_1 \text{ OR } g_2(x) \leq b_2 \text{ OR } \cdots \text{ OR } g_m(x) \leq b_m \tag{2}$$

  - This corresponds to the *union* of the regions associated with each constraint.

MILP Representability

- The connection between integer programming and disjunction is captured most elegantly by the representability theorem.

Definition 1. *A set* $\mathcal{F} \subseteq \mathbb{R}^n$ *is* MILP representable *if there exist* $A \in \mathbb{Q}^{m \times n}, G \in \mathbb{Q}^{r \times n}, b \in \mathbb{Q}^m$ *such that for*

$$\mathcal{S} = \left\{(x, y) \in (\mathbb{Z}^p \times \mathbb{R}_+^{n-p}) \times (\mathbb{Z}_+^t \times \mathbb{R}_+^{r-t}) \mid Ax + Gy \leq b\right\},$$

we have that $\mathcal{F} = \mathrm{proj}_x(\mathcal{S})$.

Theorem 1. *(MILP Representability Theorem) A set* $\mathcal{F} \subseteq \mathbb{R}^n$ *is MILP representable if and only if there exist rational polytopes* $\mathcal{P}_1, \ldots, \mathcal{P}_k$ *and vectors* $r^1, \ldots, r^t \in \mathbb{Z}^n$ *such that*

$$\mathcal{F} = \bigcup_{i=1}^{k} (\mathcal{P}_i + \mathrm{intcone}\{r^1, \ldots, r^t\})$$

- Roughly speaking, we are optimizing over a union of polyhedra all of which have the same recession cone.
- This class of problem can also be obtained simply by introducing a disjunctive logical operator to the language of linear programming.

Connection with Other Fields

- Integer programming can be studied from the point of view of a number of fundamental mathematical disciplines:
  - Algebra
  - (Projective) Geometry
  - Topology
  - Combinatorics
    - Matroid theory
    - Graph theory
  - Logic
    - Set theory
    - Formal systems and proof theory
    - Computability/complexity theory
- There are also (many) other related disciplines:
  - Constraint programming
  - Answer set programming
  - Logic programming
  - Satisfiability
  - Planning and artificial intelligence

Basic Themes

Our goal will be to expose the geometrical structure of the feasible region (at least near the optimal solution). We can do this by

- Convexification
- Outer/Inner approximation
- Lifting and Projection

An important component of the algorithms we consider will be mechanisms for computing bounds by either

- Relaxation
- Duality

When all else fails, we will employ a basic principle: divide large, difficult problems into smaller ones.

- Logic (conjunction/disjunction)
- Implicit enumeration
- Decomposition