Computational Integer Programming

Lecture 1: Introduction

Dr. Ted Ralphs

Quick Overview

- Course web site

  http://coral.ie.lehigh.edu/~ted/teaching/computational-mip

- Course structure: five days, nine sessions, alternating between
  - Lecture sessions
  - Computational exercises
- Slides will be posted on-line each day
- Material is drawn from
  - http://coral.ie.lehigh.edu/~ted/teaching/ie418
  - http://coral.ie.lehigh.edu/~ted/teaching/coin-or
  - http://co-at-work.zib.de/#schedule
  - https://github.com/tkralphs/FinancialModels/
- Please ask questions!!

Computational Tools

- We'll install these free optimization suites.
  - SCIP
  - COIN-OR
- We'll work with these modeling and programming environments.
  - AMPL
  - ZIMPL
  - Python-based
    * PuLP
    * Pyomo
    * DiPPy
- Solver Studio (?)
- Any OS should work!!

Computational Exercises

- We'll have to see how the computational exercises go.
- With so many students and just one instructor, it may be difficult.
- Please bear with me and plan to try some things at home.
- Please bring questions back the next day.

Survey Says...

- Laptop?
- Preferred OS?
- Programming Experience?
- Programming Environment?
- Optimization Background?

Reference Texts

- Nemhauser and Wolsey
- Wolsey
- Conforti, Corneujols, and Zambelli
- See also more extensive list at

  http://coral.ie.lehigh.edu/~ted/teaching/ie418

References for This Lecture

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

Mathematical Optimization Problems

- *Mathematical optimization* is a framework for formulating and analyzing optimization problems.
- The essential elements of an optimization problem are
  - a system whose operating states can be described numerically by specifying the values of certain *variables*;
  - a set of states considered *feasible* for the given system; 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 framework for formulation imposes constraints on what aspects of the system can be modeled.
- We often need to make simplifying assumptions and approximations in order to put the problem into the required form.

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 Modeling Process

- The modeling process consists generally of the following steps.
  - Determine the "real-world" state variables, system constraints, and goal(s) or objective(s) for operating the system.
  - Translate these variables and constraints into the form of a mathematical optimization problem (the "formulation").
  - Solve the mathematical optimization problem.
  - 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.
  - There may be more than one valid "formulation".
- All in all, an intimate knowledge of mathematical optimization definitely helps during the modeling process.

Formalizing: Mathematical Optimization Problems

Elements of the model:

- Decision variables
- Constraints
- Objective Function
- Parameters and Data

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

$$\begin{array}{lrcl} \text{min or max } f(x) & & & \\ \text{s.t.} & g_i(x) & \left\{ \begin{array}{c} \leq \\ = \\ \geq \end{array} \right\} & b_i \\ & x & \in & X \end{array}$$

where $X \subseteq \mathbb{R}^n$ might be a discrete set (what is a discrete set?)

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.
- 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 a feasible 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 is feasible, and
  - whether it has an optimal solution.
- 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.
  - The form of the set X.
- The most basic case in the linear optimization problem (LP) (this course assumes basic knowledge of linear optimization).
  - 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)

Our Formal Setting

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

$$z_{IP} = \max\{c^\top x \mid x \in \mathcal{S}\}, \qquad \text{(MILP)}$$

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

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

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

- This type of optimization problem is called a *mixed integer linear optimization problem*, or simply a *mixed integer optimization problem* (MIP).
- 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.

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 optimization problems*.

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 math program.
- Many COPs can be written as BIPs or MIPs.

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 nonnegativity, but the definition of $\mathcal{P}$ does not.
- One further assumption we will make is that the constraint matrix is rational.
  - 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 Discrete Optimization?

- Solving general integer programs can be much more difficult than solving linear programs.
- There in no known *polynomial-time* algorithm for solving general MIPs.
- Solving the associated *linear programming relaxation* results in an upper bound on the optimal solution to the MIP.
- In general, solving the *LP relaxation*, an LP obtained by dropping the integerality restrictions, does 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 MIP.
  - Rounding may result in a solution far from optimal.
  - We can bound the difference between the optimal solution to the LP and the optimal solution to the MIP (how?).

Integer Programming and Convexity

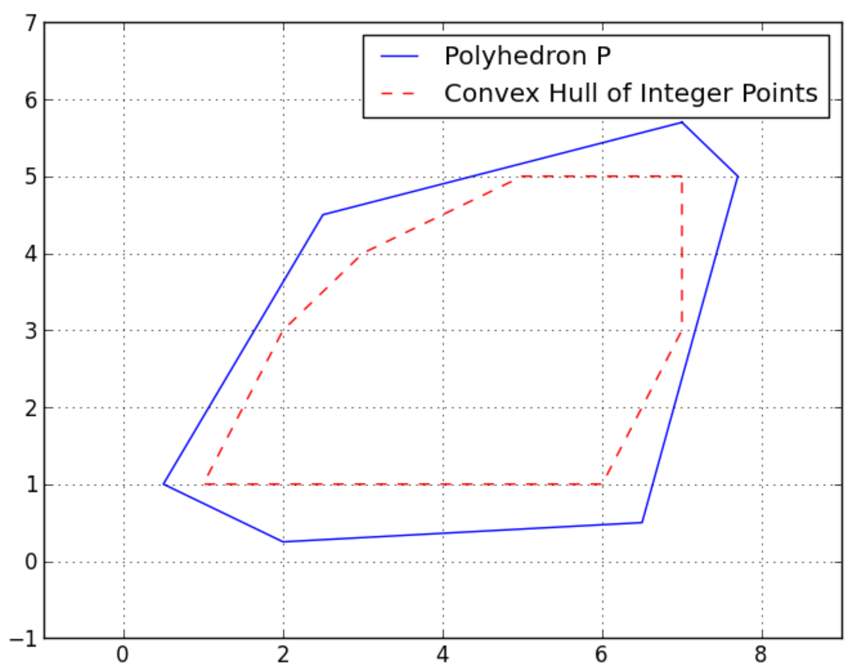
- The feasible region of an integer program is nonconvex.
- The nonconvexity is of a rather special form, though.
- Although the feasible set is nonconvex, there is a convex set over which we can optimize in order to get a solution (why?).
- The challenge is that we do not know how to describe that set.
- 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 Integer Programming

- Let's consider again an integer linear program

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

- 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 MILP?

- 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 *union* of the regions associated with each constraint.

Representability Theorem

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

Theorem 1. *(MILP Representability Theorem) A set* $\mathcal{F} \subseteq \mathbb{R}^n$ *is MIP 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, which can 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
  - Geometry
  - Topology
  - Combinatorics
    - Matroid theory
    - Graph theory
  - Logic
    - Set theory
    - Proof theory
    - Computability/complexity theory
- There are also a number of other related disciplines
  - Constraint programming
  - Satisfiability
  - Artificial intelligence