15-780 – Mixed integer programming

J. Zico Kolter

February 12, 2014
Overview

• Introduction to mixed integer programs

• Examples: Sudoku, planning with obstacles

• Solving integer programs with branch and bound

• Extensions
Overview

• Introduction to mixed integer programs

• Examples: Sudoku, planning with obstacles

• Solving integer programs with branch and bound

• Extensions
Introduction

- Recall optimization problem

\[
\begin{align*}
\text{minimize} & \quad f(x) \\
\text{subject to} & \quad g_i(x) \leq 0 \quad i = 1, \ldots, m
\end{align*}
\]

“easy” when \(f, g_i\) convex, “hard” otherwise

- But how hard? How do we even go about solving (locally or globally) these problems?

- We’ve seen how to solve discrete non-convex optimization problems with search, can we apply these same techniques for mathematical optimization?
**Mixed integer programs**

- A special case of non-convex optimization methods that lends itself to a combination of search and convex optimization

\[
\begin{align*}
\text{minimize} \quad & f(x, z) \\
\text{subject to} \quad & g_i(x, z) \leq 0 \quad i = 1, \ldots, m
\end{align*}
\]

- \( x \in \mathbb{R}^n \), and \( z \in \mathbb{Z}^p \) are optimization variables

- \( f : \mathbb{R}^n \times \mathbb{Z}^p \rightarrow \mathbb{R} \) and \( g_i : \mathbb{R}^n \times \mathbb{Z}^p \rightarrow \mathbb{R} \) convex objective and constraint functions

- *Not* a convex problem (set of all integers is not convex)

- Note: some ambiguity in naming, some refer to MIPs as only *linear* programs with integer constraints
Mixed binary integer programs

• For this class, we’ll focus on a slightly more restricted case

$$\min_{x,z} f(x, z)$$

subject to $g_i(x, z) \leq 0 \quad i = 1, \ldots, m$

$z_i \in \{0, 1\}, \quad i = 1, \ldots, p$

• Still an extremely powerful class of problems (i.e., binary integer programming is NP-complete)
Overview

• Introduction to mixed integer programs

• Examples: Sudoku, planning with obstacles

• Solving integer programs with branch and bound

• Examples (solved)

• Extensions
Example: Sudoku

• The ubiquitous Sudoku puzzle

\[
\begin{array}{cccccccc}
5 & 3 & & & & 7 & & \\
6 & & 1 & 9 & 5 & & & \\
9 & 8 & & & & 6 & & \\
8 & & 6 & & 3 & & & \\
4 & 8 & 3 & & 1 & & & \\
7 & & 2 & & 6 & & & \\
& 6 & & & 2 & 8 & & \\
& 4 & 1 & 9 & & 5 & & \\
& & 8 & & 7 & 9 & & \\
\end{array}
\]

• Can be encoded as binary integer program: let \( z_{i,j} \in \{0, 1\}^9 \) denote the “indicator” of number in the \( i, j \) position

\[
z_{6,3} = [0 \ 0 \ 1 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0]^T \iff
\begin{array}{cccccccc}
5 & 3 & & & & 7 & & \\
6 & & 1 & 9 & 5 & & & \\
9 & 8 & & & & 6 & & \\
8 & & 6 & & 3 & & & \\
4 & 8 & 3 & & 1 & & & \\
7 & & 2 & & 6 & & & \\
& 6 & & & 2 & 8 & & \\
& 4 & 1 & 9 & & 5 & & \\
& & 8 & & 7 & 9 & & \\
\end{array}
\]
• Each square can have only one number
\[
\sum_{k=1}^{9} (z_{i,j})_k = 1, \quad i, j = 1, \ldots, 9
\]

• Every row must contain each number
\[
\sum_{j=1}^{9} z_{i,j} = 1, \text{ (all ones vector)} \quad i = 1, \ldots, 9
\]

• Every column must contain each number
\[
\sum_{i=1}^{9} z_{i,j} = 1, \quad j = 1, \ldots, 9
\]

• Every 3x3 block must contain each number
\[
\sum_{k,\ell=1}^{3} z_{i+k,j+\ell} = 1, \quad i, j \in \{0, 3, 6\}
\]
Final optimization problem (note that objective is irrelevant, as we only care about finding a feasible point)

\[
\begin{align*}
\text{minimize} & \quad \sum_{i,j=1}^{9} \max_k (z_{i,j})_k \\
\text{subject to} & \quad z_{i,j} \in \{0, 1\}^9, \; i, j = 1, \ldots, 9 \\
& \quad \sum_{k=1}^{9} (z_{i,j})_k = 1, \; i, j = 1, \ldots, 9 \\
& \quad \sum_{j=1}^{9} z_{i,j} = 1, \; i = 1, \ldots, 9 \\
& \quad \sum_{i=1}^{9} z_{i,j} = 1, \; j = 1, \ldots, 9 \\
& \quad \sum_{k,\ell=1}^{3} z_{i+k,j+\ell} = 1, \; i, j \in \{0, 3, 6\}
\end{align*}
\]
Example: path planning with obstacles

- Find path from start to goal that avoids obstacles

\[
\begin{align*}
\text{Start} & \quad \text{Goal} \\
\square & \quad \bigcirc
\end{align*}
\]

- Represent path as set of points \( x_i \in \mathbb{R}^2, i = 1, \ldots, m \) and minimize squared distance between consecutive points

- Obstacle is defined by \( a, b \in \mathbb{R}^2 \)

\[
O = \{ x : a_1 \leq x_1 \leq b_1, a_2 \leq x_2 \leq b_2 \}
\]
• Constraint that we \textit{not} hit obstacle can be represented as

\[(x_i)_1 \leq a_1 \lor (x_i)_1 \geq b_1 \lor (x_i)_2 \leq a_2 \lor (x_i)_2 \geq b_2, \ i = 1, \ldots, m\]

• How can we represent this using binary variables?
• The trick: “big-M” method

• Let $M \in \mathbb{R}$ be some big number and consider the constraint

\[(x_i)_1 \leq a_1 + zM\]

for $z \in \{0, 1\}$; if $z = 0$, this is the same as the original constraint, but if $z = 1$ then constraint will always be satisfied

• Introduce new variables $z_{i1}, z_{i2}, z_{i3}, z_{i4}$ for each $x_i$

\[
\begin{align*}
(x_i)_1 &\leq a_1 + z_{i1}M \\
(x_i)_1 &\geq b_1 - z_{i2}M \\
(x_i)_2 &\leq a_2 + z_{i3}M \\
(x_i)_2 &\geq b_2 - z_{i4}M \\
z_{i1} + z_{i2} + z_{i3} + z_{i4} &\leq 3
\end{align*}
\]
Final optimization problem

\[
\begin{align*}
\text{minimize} & \quad \sum_{i=1}^{m-1} \|x_{i+1} - x_i\|_2^2 \\
\text{subject to} & \quad (x_i)_1 \leq a_1 + z_{i1}M \\
& \quad (x_i)_1 \geq b_1 - z_{i2}M \\
& \quad (x_i)_2 \leq a_2 + z_{i3}M \\
& \quad (x_i)_2 \geq b_2 - z_{i4}M \\
& \quad z_{i1} + z_{i2} + z_{i3} + z_{i4} \leq 3 \\
& \quad z_{ij} \in \{0, 1\}, \quad i = 1, \ldots, m, \quad j = 1, \ldots, 4 \\
x_1 = \text{start}, \quad x_m = \text{goal}
\end{align*}
\]
Overview

- Introduction to mixed integer programs
- Examples: Sudoku, planning with obstacles
- Solving integer programs with branch and bound
- Examples (solved)
- Extensions
Solution via enumeration

- Recall that optimization problem

\[
\begin{align*}
\text{minimize } & \quad f(x, z) \\
\text{subject to } & \quad g_i(x, z) \leq 0 \quad i = 1, \ldots, m \\
& \quad z_i \in \{0, 1\}, \quad i = 1, \ldots, p
\end{align*}
\]

is easy for a fixed \( z \) (then a convex problem)

- So, just enumerate all possible \( z \)’s and solve optimization problem for each

- \( 2^p \) possible assignments, quickly becomes intractable
Branch and bound

- Branch and bound is simply a search algorithm (best-first search) applied to finding the optimal $z$ assignment

- In the worst case, still exponential (have to check every possible assignment)

- In many cases much better
Convex relaxations

• The key idea: *convex relaxation* of non-convex constraint

\[
\begin{align*}
\text{minimize} & \quad f(x, z) \\
\text{subject to} & \quad g_i(x, z) \leq 0 \quad i = 1, \ldots, m \\
& \quad z_i \in \{0, 1\}, \quad i = 1, \ldots, p
\end{align*}
\]
Convex relaxations

• The key idea: \textit{convex relaxation} of non-convex constraint

\[
\begin{align*}
\text{minimize} \quad & f(x, \bar{z}) \\
\text{subject to} \quad & g_i(x, \bar{z}) \leq 0 \quad i = 1, \ldots, m \\
& \bar{z}_i \in [0, 1], \quad i = 1, \ldots, p
\end{align*}
\]
Convex relaxations

• The key idea: *convex relaxation* of non-convex constraint

\[
\begin{align*}
\text{minimize} & \quad f(x, \bar{z}) \\
\text{subject to} & \quad g_i(x, \bar{z}) \leq 0 \quad i = 1, \ldots, m \\
& \quad \bar{z}_i \in [0, 1], \quad i = 1, \ldots, p
\end{align*}
\]

• Key point: if the optimal solution \( \bar{z}^* \) to the relaxation is integer valued, then it is an optimal solution to the integer program

• Furthermore, all solutions to relaxed problem provide *lower bounds* on optimal objective

\[
f(x^*, \bar{z}^*) \leq f(x^*, z^*)
\]
Simple branch and bound algorithm

• Idea of approach
  1. Solve relaxed problem

  2. If there are variables $\tilde{z}_i^*$ with non-integral solutions, pick one of the variables and recursively solve each relaxation with $\tilde{z}_i = 0$ and $\tilde{z}_i = 1$

  3. Stop when a solution is integral

• By using best-first search (based upon lower bound given by relaxation), we potentially need to search many fewer possibilities than for enumeration
function \((f, x^*, \bar{z}^*, C) = \text{Solve-Relaxtion}(C)\)
// solves relaxation plus constraints in \(C\)

\[
q \leftarrow \text{Priority-Queue}()
\]
\[
q.\text{push}(\text{Solve-Relaxtion}(\{\}))
\]

**while** (q not empty):
\[
(f, x^*, \bar{z}^*, C) \leftarrow q.\text{pop}()
\]

**if** \(\bar{z}^*\) integral:
\[
\text{return } (f, x^*, \bar{z}^*, C)
\]

**else:**

Choose \(i\) such that \(\bar{z}_i\) non-integral
\[
q.\text{push}(\text{Solve-Relaxtion}(C \cup \{\bar{z}_i = 0\}))
\]
\[
q.\text{push}(\text{Solve-Relaxtion}(C \cup \{\bar{z}_i = 1\}))
\]
• A common modification: in addition to maintaining lower bound from relaxation, maintain an upper bound on optimal objective

• Common method for computing upper bound: round entries in $\bar{Z}_i$ to nearest integer, and solve optimization problem with this fixed $\bar{Z}$

• (May not produce a feasible solution)
**function** \((f, x^*, \bar{z}^*, C) = \text{Solve-Relaxtion}(C)\)

\(/ / \text{solves relaxation plus constraints in } C\)

q ← Priority-Queue()
q2 ← Priority-Queue()
q.push(Solve-Relaxtion(\{\}))

**while** (q not empty):
   \((f, x^*, \bar{z}^*, C) ← q.pop()\)
   q2.push(Solve-Relaxtion(\{\bar{z} = \text{round}(\bar{z}^*)\}))

   **if** q2.first() − \(f < \epsilon\):
      **return** q2.pop()
   **else**:
      Choose \(i\) such that \(\bar{z}_i\) non-integral
      q.push(Solve-Relaxtion(\(C \cup \{\bar{z}_i = 0\}\)))
      q.push(Solve-Relaxtion(\(C \cup \{\bar{z}_i = 1\}\)))
Simple example (from Boyd and Mattingley)

\[ \begin{align*}
\text{minimize} & \quad 2z_1 + z_2 - 2z_3 \\
\text{subject to} & \quad 0.7z_1 + 0.5z_2 + z_3 \geq 1.8 \\
& \quad z_i \in \{0, 1\}, \quad i = 1, 2, 3
\end{align*} \]
Simple example (from Boyd and Mattingley)

\[
\begin{align*}
\text{minimize} & \quad 2z_1 + z_2 - 2z_3 \\
\text{subject to} & \quad 0.7z_1 + 0.5z_2 + z_3 \geq 1.8 \\
& \quad z_i \in [0, 1], \quad i = 1, 2, 3
\end{align*}
\]

Search tree

\[
\{\}
\]

Queue

\[
(-0.143, [0.43, 1, 1], \{\})
\]
Simple example (from Boyd and Mattingley)

\[
\begin{align*}
\text{minimize} & \quad 2z_1 + z_2 - 2z_3 \\
\text{subject to} & \quad 0.7z_1 + 0.5z_2 + z_3 \geq 1.8 \\
& \quad z_i \in [0, 1], \quad i = 1, 2, 3
\end{align*}
\]

Search tree

\[
\begin{array}{c}
\{\}\rightarrow z_1 = 0 \\
& \rightarrow z_1 = 1
\end{array}
\]

Queue

\[
(0.2, [1, 0.2, 1], \{z_1 = 1\}) \\
(\infty, -, \{z_1 = 0\})
\]
Simple example (from Boyd and Mattingley)

\[
\begin{align*}
\text{minimize} & \quad 2z_1 + z_2 - 2z_3 \\
\text{subject to} & \quad 0.7z_1 + 0.5z_2 + z_3 \geq 1.8 \\
& \quad z_i \in [0, 1], \quad i = 1, 2, 3
\end{align*}
\]

Search tree

\[
\begin{array}{c}
\{\} \\
| \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad
\end{array}
\]

Queue

\[
\begin{align*}
(1, [1, 1, 1], \{z_1 = 1, z_2 = 1\}) \\
(\infty, -, \{z_1 = 0\}) \\
(\infty, -, \{z_1 = 1, z_2 = 0\})
\end{align*}
\]
Sudoku revisited

- The hard part with Sudoku is finding puzzles where the initial linear programming relaxation is not already tight.

- Branch and bound solves this problem after 27 steps.
minimize \( \sum_{i,j=1}^{9} \max_k (z_{i,j})_k \)

subject to \( z \in \text{Valid-Sudoku} \)

\( z_{i,j} \in \{0, 1\}^9, \quad i,j = 1, \ldots, 9 \)
Path planning with obstacles

- Final optimization problem

\[
\begin{align*}
\text{minimize} & \quad \sum_{i=1}^{m-1} \|x_{i+1} - x_i\|^2_2 \\
\text{subject to} & \quad (x_i)_1 \leq a_1 + z_{i1}M \\
& \quad (x_i)_1 \geq b_1 - z_{i2}M \\
& \quad (x_i)_2 \leq a_2 + z_{i3}M \\
& \quad (x_i)_2 \geq b_2 - z_{i4}M \\
& \quad z_{i1} + z_{i2} + z_{i3} + z_{i4} \leq 3 \\
& \quad z_{ij} \in \{0, 1\}, \quad i = 1, \ldots, m \\
& \quad x_1 = \text{start}, \quad x_m = \text{goal}
\end{align*}
\]
Overview

- Introduction to mixed integer programs
- Examples: Sudoku, planning with obstacles
- Solving integer programs with branch and bound
- Extensions
Extensions to MIP

• How to incorporate actual integer (instead of just binary) constraints?
  – When solution is non-integral, split after adding constraints
    \[ \{ \bar{z}_i \leq \text{floor}(\bar{z}_i^*) \}, \{ \bar{z}_i \geq \text{ceil}(\bar{z}_i^*) \} \]

• More advanced splits, addition of “cuts” that rule out non-integer solutions (branch and cut)

• Solve convex problems more efficiently, many solvers can be sped up given a good initial point, and many previous solutions will be good initializations
Take home points

• Integer programs are a power subset of non-convex optimization problems that can solve many problems of interest

• Combining search and numerical optimization techniques, we get an algorithm that solve many problems much more efficiently than the “brute force” approach

• Performance will still be exponential in the worst case, and problem dependent, but can be reasonable for many problems of interest