15-780 – Graduate Artificial Intelligence: Integer programming

J. Zico Kolter (this lecture) and Nihar Shah
Carnegie Mellon University
Spring 2019
Outline

Introduction

Integer programming

Solving integer programs

Extensions and discussion
Logistics

Deadline for forming group project teams is Feb 14

More information about the project to be posted this week, but you are still encouraged to form groups in the meantime
Outline

Introduction

Integer programming

Solving integer programs

Extensions and discussion
A large factory makes tables and chairs. Each table returns a profit of $200 and each chair a profit of $100. Each table takes 1 unit of metal and 3 units of wood and each chair takes 2 units of metal and 1 unit of wood. The factory has 6K units of metal and 9K units of wood. How many tables and chairs should the factory make to maximize profit?
A large factory makes tables and chairs. Each table returns a profit of $200 and each chair a profit of $100. Each table takes 1 unit of metal and 3 units of wood and each chair takes 2 units of metal and 1 unit of wood. The factory has 6K units of metal and 9K units of wood. How many tables and chairs should the factory make to maximize profit?
A large factory makes tables and chairs. Each table returns a profit of $200 and each chair a profit of $100. Each table takes 1 unit of metal and 3 units of wood and each chair takes 2 units of metal and 1 unit of wood. The factory has 6 units of metal and 9 units of wood. How many tables and chairs should the factory make to maximize profit?
A large factory makes tables and chairs. Each table returns a profit of $200 and each chair a profit of $100. Each table takes 1 unit of metal and 3 units of wood and each chair takes 2 units of metal and 1 unit of wood. The factory has 6 units of metal and 9 units of wood. How many tables and chairs should the factory make to maximize profit?
Challenges of integer programming

The above example was “easy” in that the rounded solution to the LP happened to also be a solution to the integer program.

In general, integer solution can be arbitrarily far from the LP solution.

Can be hard to even find a feasible solution that is integer valued, e.g., imagine the task of finding an integer solution to some arbitrary set of linear equations $Ax = b$. 
Many applications (see next lecture)

Path planning with obstacles

Many problems in game theory

Constraint satisfaction problems

(Exact) most likely assignment in graphical models

Scheduling and unit commitment

Kidney exchange
Outline

Introduction

Integer programming

Solving integer programs

Extensions and discussion
Integer linear programming

An optimization problem like linear programming, except that variables are required to take on integer values (e.g., in inequality form)

\[
\begin{align*}
\text{minimize} & \quad c^T x \\
\text{subject to} & \quad Gx \leq h \\
& \quad x \in \mathbb{Z}^n \quad \text{(integers)}
\end{align*}
\]

Not a convex problem, because of integer constraint (set of all integers is not a convex set)

Can also consider mixed integer linear programming, with both integer and non-integer variables

\[
\begin{align*}
\text{minimize} & \quad c^T x \\
\text{subject to} & \quad Gx \leq h \\
& \quad x_i \in \mathbb{Z}, \quad i \in J \subseteq \{1, \ldots, n\}
\end{align*}
\]
Other variations

For simplicity, in this lecture we will focus on binary integer programming, where \( x \) variables are in \( \{0, 1\} \)

\[
\text{minimize} \quad c^T x \\
\text{subject to} \quad Gx \leq h \\
x \in \{0, 1\}^n
\]

This is just for ease of presentation, we will discuss how to adapt all these methods for general integer variables.

Techniques we present are actually largely applicable to any mixed integer programming problem with convex objective and constraints (other than integer constraint)
Difficult of binary integer programming

**Theorem:** Binary integer programming is NP-hard

**Proof:** We show this by reduction from 3SAT

Recall the 3SAT satisfiability problem: given binary variables \( x_1, \ldots, x_n \in \{True, False\} \) determine if there is some assignment that satisfies a set of clauses in conjunctive normal form, e.g.,

\[
(x_1 \lor x_2 \lor \neg x_3) \land (x_2 \lor \neg x_4 \lor x_5) \land \cdots
\]

Formulate this as a binary integer program: \( x_1, \ldots, x_n \in \{0,1\} \), with e.g.,

\[
(x_1 \lor x_2 \lor \neg x_3) \iff x_1 + x_2 + (1 - x_3) \geq 1
\]

Finding feasible solution to BIP equivalent to finding satisfying assignment
Outline

Introduction

Integer programming

Solving integer programs

Extensions and discussion
Solving integer programs

How can we go about finding the solution to the binary integer program

\[
\begin{align*}
\text{minimize} & \quad c^T x \\
\text{subject to} & \quad Gx \leq h \\
& \quad x \in \{0,1\}^n
\end{align*}
\]

Naïve solution: \(2^n\) possible assignments of all \(x\) variables, just try each one, return solution with minimum objective value out of those that satisfy constraints

In the worst case, we can’t do any better than this, but often it is possible to solve the problem \textit{much} faster in practice
Key idea: relaxing integer constraints

Consider alternate optimization problem where we relax the constraint $x_i \in \{0,1\}$ to be $x_i \in [0,1] \equiv 0 \leq x_i \leq 1$, forming the linear program:

minimize $c^T x$
subject to $Gx \leq h$

$x \in [0,1]^n$

Key point #1: if the solution to this linear program $x^*$ has all integer values, then it is also the solution to the integer program

Key point #2: the optimal objective for the linear program will be lower than that of the binary integer program

Both points follow trivially from the fact that $\{0,1\}^n \subset [0,1]^n$
Integer solutions

Integer solutions are more common than you may naively expect, will happen whenever the vertices of the polytope all have integer values.

E.g., consider trivial optimization problem

\[
\begin{align*}
\text{minimize} & \quad c^T x \\
\text{subject to} & \quad x \in [0,1]^n
\end{align*}
\]

Solution is:

\[
x^*_i = \begin{cases} 
1 & \text{if } c_i \leq 0 \\
0 & \text{otherwise}
\end{cases}
\]
Consider the linear program, with $c$ and $h$ chosen so problem is feasible:

$$\begin{align*}
\text{minimize} \quad & c^T x \\
\text{subject to} \quad & x \in [0,1]^n \\
& x_i - x_{i+1} \leq h_i, \quad i = 1, \ldots, n-1
\end{align*}$$

Will the solutions (optimal $x$ values) to this LP always take on integer values?

1. Yes, for all values of $c \in \mathbb{R}^n$ and $h \in \mathbb{R}^{n-1}$

2. Yes, for any $c \in \mathbb{R}^n$ but only for $h \in \mathbb{Z}^{n-1}$

3. Yes, but only for $c \in \mathbb{Z}^n$ and $h \in \mathbb{Z}^{n-1}$

4. Not necessarily, even for $c \in \mathbb{Z}^n$ and $h \in \mathbb{Z}^{n-1}$
Branch and bound

LP relaxation is an quickly-computable approximation which gives us a lower bound on the true solution: sounds a lot like an admissible heuristic…

This leads us to the branch and bound algorithm: this is just greedy informed search (i.e., $f(s) = h(s)$, no path cost, just heuristic cost), applied using LP relaxation as the heuristic

Repeat:
1. Choose relaxed problem from frontier with lowest cost
2. If solution is not integer valued, pick a non-integer variable $x_i$ and add problems with additional constraints $x_i = 0$ and $x_i = 1$
3. If solution is integer valued, return
Branch and bound in more detail

A more detailed description of branch and bound

Function: Solve-Relaxation(\(C\)):
- Solve linear program plus additional constraints in \(C\)
- Return (objective value \(f^*\), solution \(x^*\), and constraint set \(C\))

Algorithm: Branch-and-Bound
- Push Solve-Relaxation(\(\{\}\)) on to frontier set
- Repeat while frontier is not empty:
  1. Get lowest cost solution from frontier: \((f, x, C)\)
  2. If \(x\) is integer valued, return \(x\)
  3. Else, choose some \(x_i\) not integer valued and add
     Solve-Relaxation(\(C \cup \{x_i = 0\}\)), Solve-Relaxation(\(C \cup \{x_i = 1\}\)),
     to the frontier
Branch and bound example

minimize \[ 2x_1 + x_2 - 2x_3 \]
subject to \[ 0.7x_1 + 0.5x_2 + x_3 \geq 1.8 \]
\[ x_i \in \{0,1\}, \quad i = 1,2,3 \]
Branch and bound example

\[
\begin{align*}
\text{minimize} & \quad 2x_1 + x_2 - 2x_3 \\
\text{subject to} & \quad 0.7x_1 + 0.5x_2 + x_3 \geq 1.8 \
& \quad x_i \in [0,1], \quad i = 1,2,3
\end{align*}
\]

Search tree  \hspace{2cm} \text{Frontier}

\[
\begin{array}{l}
\{\} \\
\end{array}
\]

\[
(f^* = -0.143, x^* = [0.43,1,1], \mathcal{C} = \{\})
\]
Branch and bound example

minimize \( x \) \( 2x_1 + x_2 - 2x_3 \)
subject to \( 0.7x_1 + 0.5x_2 + x_3 \geq 1.8 \)
\( x_i \in [0,1], \quad i = 1,2,3 \)

Search tree

\[
\begin{array}{c}
\{ \} \\
\{ x_1 = 0 \} \\
\{ x_1 = 1 \}
\end{array}
\]

Frontier

\[
\begin{align*}
(f^* = 0.143, x^* = [0.43,1,1], C = \{\}) \\
(f^* = 0.2, x^* = [1,0.2,1], C = \{x_1 = 1\}) \\
(f^* = \infty, x^* = \emptyset, C = \{x_1 = 0\})
\end{align*}
\]
Branch and bound example

\[
\begin{align*}
\text{minimize} & \quad 2x_1 + x_2 - 2x_3 \\
\text{subject to} & \quad 0.7x_1 + 0.5x_2 + x_3 \geq 1.8 \\
& \quad x_i \in [0, 1], \quad i = 1, 2, 3
\end{align*}
\]

Search tree

Frontier

\[
\begin{align*}
(f^* = 0.143, x^* = [0.43, 1, 1], C = \{\}) \\
(f^* = 0.2, x^* = [1, 0.2, 1], C = \{x_1 = 1\}) \\
(f^* = 1, x^* = [1, 1, 1], C = \{x_1 = 1, x_2 = 1\}) \\
(f^* = \infty, x^* = \emptyset, C = \{x_1 = 0\}) \\
(f^* = \infty, x^* = \emptyset, C = \{x_1 = 1, x_2 = 0\})
\end{align*}
\]
Branch and bound example

minimize \( x \) \[ 2x_1 + x_2 - 2x_3 \]
subject to \( 0.7x_1 + 0.5x_2 + x_3 \geq 1.8 \)
\( x_i \in [0,1], \quad i = 1,2,3 \)

Search tree

Frontier

\[ (f^* = 0.143, x^* = [0.43,1,1], C = \{\}) \]
\[ (f^* = 0.2, x^* = [1,0.2,1], C = \{x_1 = 1\}) \]
\[ (f^* = 1, x^* = [1,1,1], C = \{x_1 = 1, x_2 = 1\}) \]
\[ (f^* = \infty, x^* = \emptyset, C = \{x_1 = 0\}) \]
\[ (f^* = \infty, x^* = \emptyset, C = \{x_1 = 1, x_2 = 0\}) \]
Upper bounds

Often want to also maintain an upper (feasible) bound when possible

Algorithm: Branch-and-Bound-2
- Push Solve-Relaxation(\{\}) on to frontier set
- Set $\bar{f} = \infty$
- Repeat while frontier is not empty:
  1. Get lowest cost solution from frontier: $(f, x, C)$
  2. Set $\hat{x} = \text{round}(x)$, if feasible and $c^T\hat{x} < \bar{f}$: $\bar{f} = c^T\hat{x}$, $\bar{x} = \hat{x}$
  3. If $\bar{f} - f \leq \epsilon$, return $\bar{x}$
  4. Else, choose some $x_i$ not integer valued and add
     Solve-Relaxation($C \cup \{x_i = 0\}$),Solve-Relaxation($C \cup \{x_i = 1\}$),
     to the frontier
Next class, and on the homework, we’ll see how to formulate sudoku problems as integer programs.

“World’s hardest sudoku”, let’s see how branch and bound fares.
Extensions

Branch and bound solves problem after expanding 27 nodes

In fact, it’s not that easy to find a sudoku problem where the initial LP relaxation is not already tight…
Outline

Introduction

Integer programming

Solving integer programs

Extensions and discussion
Extension to non-binary / mixed problems

For problems with general integer constraint (not just binary constraints), the algorithm is virtually identical.

Only difference is that we pick non-integer $\tilde{x}_j$ and then add
to the frontier

\[ \text{Solve-Relaxation}(\mathcal{C} \cup \{ x_i \geq \lceil \tilde{x}_j \rceil \}), \text{Solve-Relaxation}(\mathcal{C} \cup \{ x_i \leq \lfloor \tilde{x}_j \rfloor \}) \]

Can deal with mixed integer problems (some non-integer variables), by simply not branching on non-integer variables, and by resolving over non-integer variables after rounding integer variables.
Unusual to use pure branch and bound to solve real-world problems

Real solvers additionally use a concept called cutting planes to further restrict the allowable set of non-integer solutions (“Branch and cut”)

Example: Gomory cut, in simplex method, consider row in \( \tilde{A} = A_j^{-1} A \), where corresponding entry in \( \tilde{x}_j = A_j^{-1} b \) is not integer valued (call the row \( \tilde{a}_j \)); add the constraint

\[
(\tilde{a}_j - \lfloor \tilde{a}_j \rfloor)^T x \geq \tilde{x}_j - \lfloor \tilde{x}_j \rfloor
\]

We won’t prove it, but not too hard to show that this constraint rules out the current (non-integer) solution, while not excluding any integer solutions
Off-the-shelf solvers

Extremely well-developed set of commercial solvers are available (free for academic use), two most well known are CPLEX and Gurobi

Extremely well-vetted set of “pre-solve” problem simplification methods, simplex and other LP solvers, branch and bound, and cutting plane generation methods

Open source notably lags behind in this area, but SCIP solver (http://scip.zib.de/) is the best one I’m aware of (“only” ~7X worse than CPLEX, Gurobi on benchmark running times)