

15-780 – Graduate Artificial Intelligence: Integer programming

J. Zico Kolter (this lecture) and Nihar Shah
Carnegie Mellon University
Spring 2020

Outline

Introduction

Integer programming

Solving integer programs

Extensions and discussion

Outline

Introduction

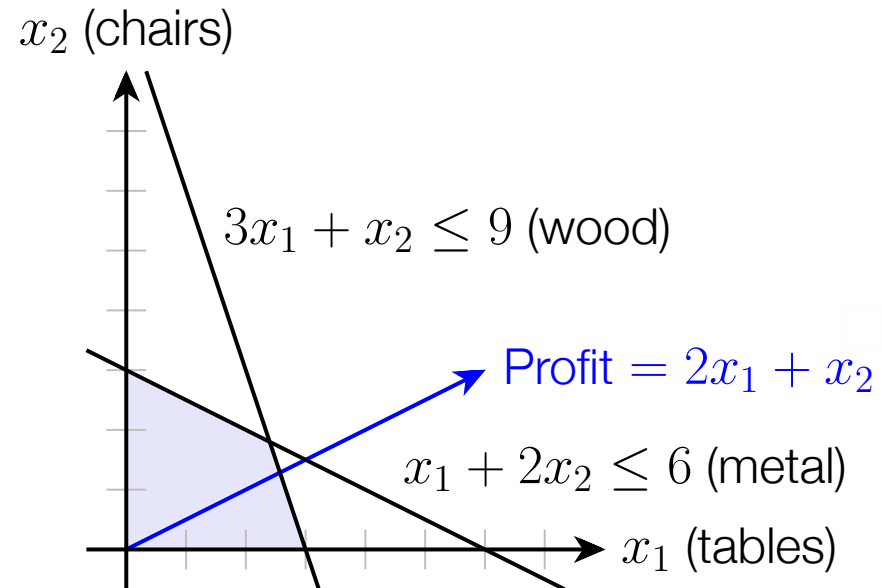
Integer programming

Solving integer programs

Extensions and discussion

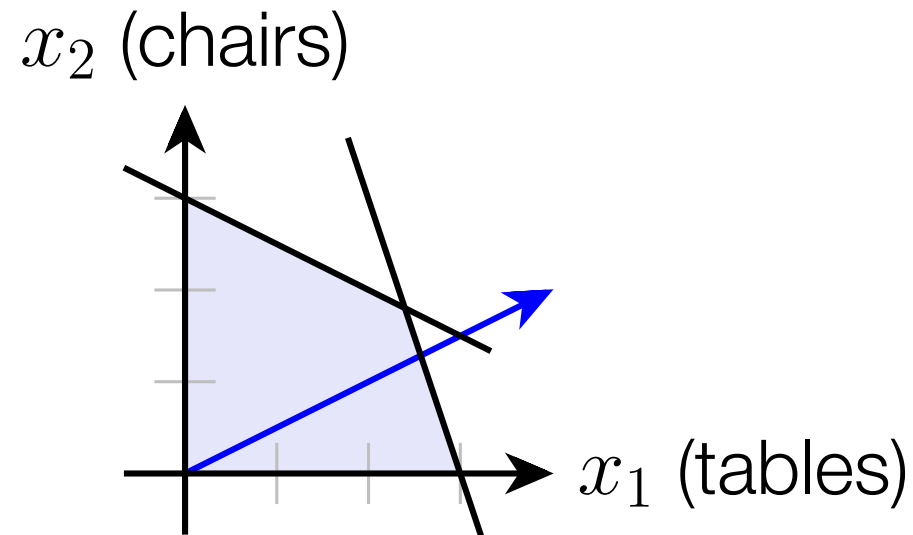
Recall linear program

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?



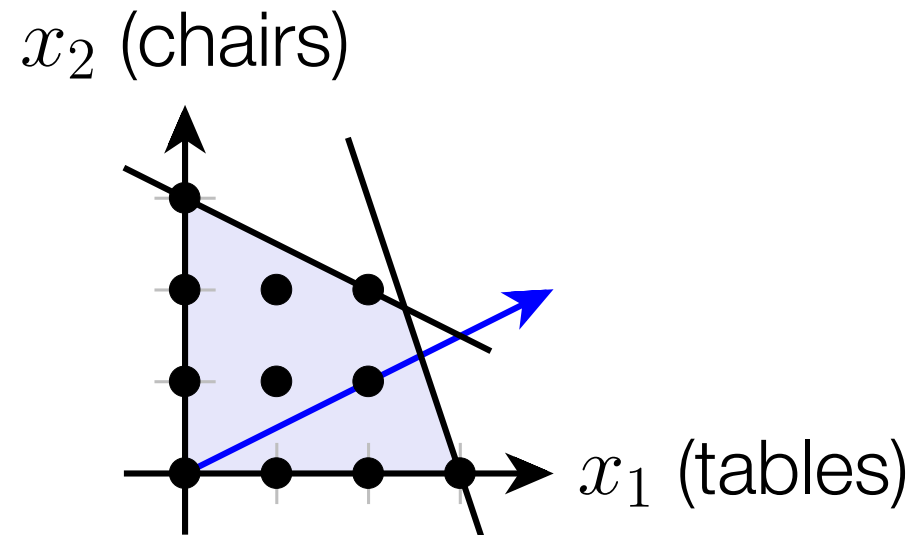
Recall linear program

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?



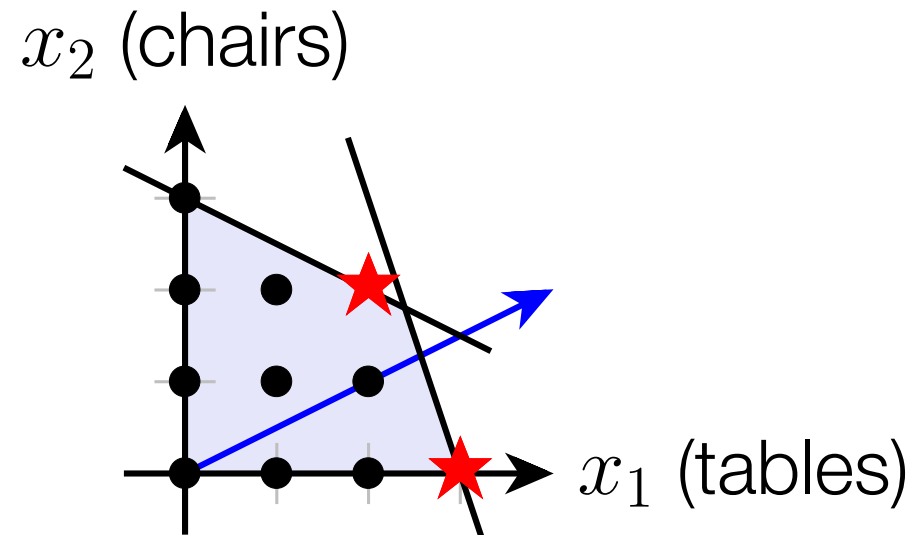
Recall linear program

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?



Recall linear program

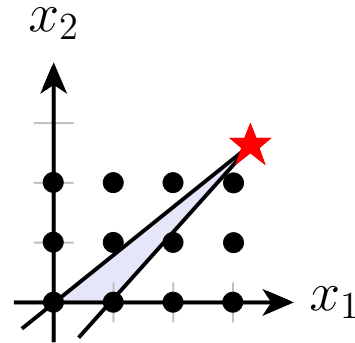
A large factory makes tables and chairs. Each table returns a profit of \$200 and each chair a profit of \$100. Each table takes 1 unit of metal and 3 units of wood and each chair takes 2 units of metal and 1 unit of wood. The factory has 6 units of metal and 9 units of wood. How many tables and chairs 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)

$$\begin{aligned} & \underset{x}{\text{minimize}} && c^T x \\ & \text{subject to} && Gx \leq h \\ & && x \in \mathbb{Z}^n \text{ (integers)} \end{aligned}$$

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

$$\begin{aligned} & \underset{x}{\text{minimize}} && c^T x \\ & \text{subject to} && Gx \leq h \\ & && x_i \in \mathbb{Z}, \quad i \in \mathcal{J} \subseteq \{1, \dots, n\} \end{aligned}$$

Other variations

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

$$\begin{aligned} & \underset{x}{\text{minimize}} && c^T x \\ & \text{subject to} && Gx \leq h \\ & && x \in \{0,1\}^n \end{aligned}$$

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 $x_1, \dots, x_n \in \{True, False\}$ determine if there is some assignment that satisfies a set of clauses in conjunctive normal form, e.g.,

$$(x_1 \vee x_2 \vee \neg x_3) \wedge (x_2 \vee \neg x_4 \vee x_5) \wedge \dots$$

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

$$(x_1 \vee x_2 \vee \neg x_3) \iff x_1 + x_2 + (1 - x_3) \geq 1$$

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

$$\begin{aligned} & \underset{x}{\text{minimize}} && c^T x \\ & \text{subject to} && Gx \leq h \\ & && x \in \{0,1\}^n \end{aligned}$$

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 *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:

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

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

$$\begin{array}{ll} \underset{x}{\text{minimize}} & c^T x \\ \text{subject to} & x \in [0,1]^n \end{array}$$

Solution is:

$$x_i^* = \begin{cases} 1 & \text{if } c_i \leq 0 \\ 0 & \text{otherwise} \end{cases}$$

Poll: integer solutions

Consider the linear program, with c and h chosen so problem is feasible:

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

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 a 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(\mathcal{C}):

- Solve linear program plus additional constraints in \mathcal{C}
- Return (objective value f^* , solution x^* , and constraint set \mathcal{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, \mathcal{C})
 2. If x is integer valued, return x
 3. Else, choose some x_i not integer valued and add
Solve-Relaxation($\mathcal{C} \cup \{x_i = 0\}$), Solve-Relaxation($\mathcal{C} \cup \{x_i = 1\}$), to the frontier

Branch and bound example

$$\begin{aligned} & \underset{x}{\text{minimize}} && 2x_1 + x_2 - 2x_3 \\ & \text{subject to} && 0.7x_1 + 0.5x_2 + x_3 \geq 1.8 \\ & && x_i \in \{0,1\}, \quad i = 1,2,3 \end{aligned}$$

Branch and bound example

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

Search tree

$$\boxed{\{}}$$

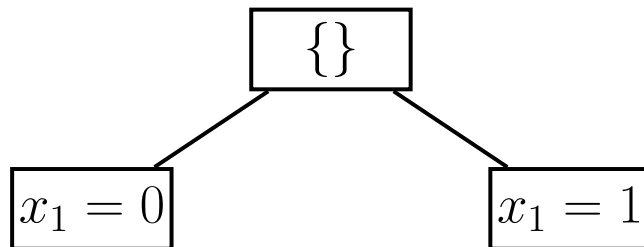
Frontier

$$(f^* = -0.143, x^* = [0.43, 1, 1], \mathcal{C} = \{\})$$

Branch and bound example

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

Search tree



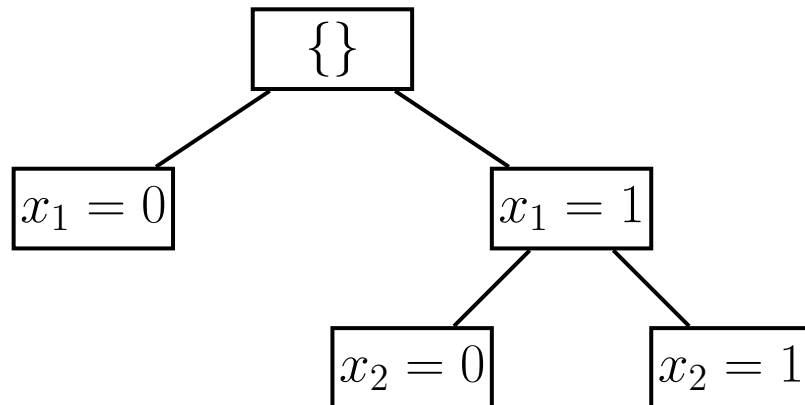
Frontier

$$\begin{aligned} & (\cancel{f^* = -0.143, x^* = [0.43, 1, 1], \mathcal{C} = \{\}}) \\ & (f^* = 0.2, x^* = [1, 0.2, 1], \mathcal{C} = \{x_1 = 1\}) \\ & (f^* = \infty, x^* = \emptyset, \mathcal{C} = \{x_1 = 0\}) \end{aligned}$$

Branch and bound example

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

Search tree



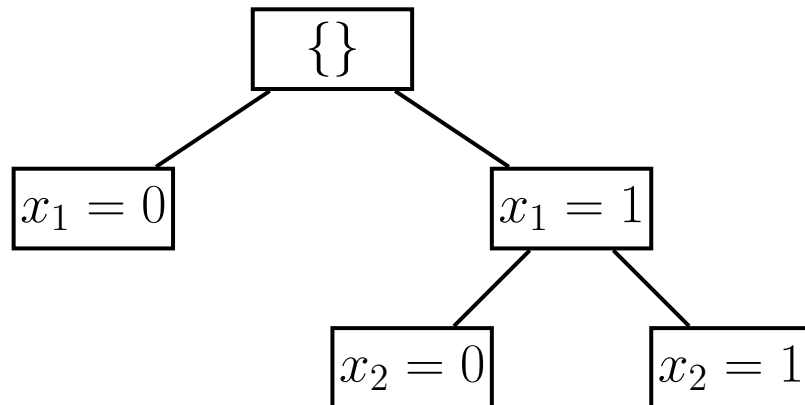
Frontier

$$\begin{aligned} & (\cancel{f^* = -0.143, x^* = [0.43, 1, 1], \mathcal{C} = \{\}}) \\ & (\cancel{f^* = 0.2, x^* = [1, 0.2, 1], \mathcal{C} = \{x_1 = 1\}}) \\ & (f^* = 1, x^* = [1, 1, 1], \mathcal{C} = \{x_1 = 1, x_2 = 1\}) \\ & (f^* = \infty, x^* = \emptyset, \mathcal{C} = \{x_1 = 0\}) \\ & (f^* = \infty, x^* = \emptyset, \mathcal{C} = \{x_1 = 1, x_2 = 0\}) \end{aligned}$$

Branch and bound example

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

Search tree



Frontier

$$\begin{aligned} & (\cancel{f^* = -0.143, x^* = [0.43, 1, 1], \mathcal{C} = \{\}}) \\ & (\cancel{f^* = 0.2, x^* = [1, 0.2, 1], \mathcal{C} = \{x_1 = 1\}}) \\ & (f^* = 1, x^* = [1, 1, 1], \mathcal{C} = \{x_1 = 1, x_2 = 1\}) \\ & (f^* = \infty, x^* = \emptyset, \mathcal{C} = \{x_1 = 0\}) \\ & (f^* = \infty, x^* = \emptyset, \mathcal{C} = \{x_1 = 1, x_2 = 0\}) \end{aligned}$$

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, \mathcal{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($\mathcal{C} \cup \{x_i = 0\}$), Solve-Relaxation($\mathcal{C} \cup \{x_i = 1\}$), to the frontier

Illustration: sudoku

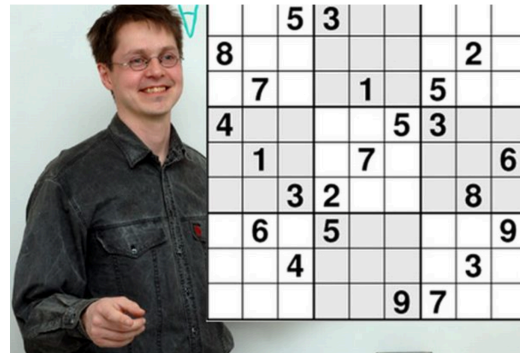
Next class, and on the homework, we'll see how to formulate sudoku problems as integer programs

News - Weird News

World's hardest sudoku: Can you solve Dr Arto Inkala's puzzle?

Aug 19, 2010 00:00 | By Mirror.co.uk | 0 Comments

Could this be the toughest sudoku puzzle ever devised?



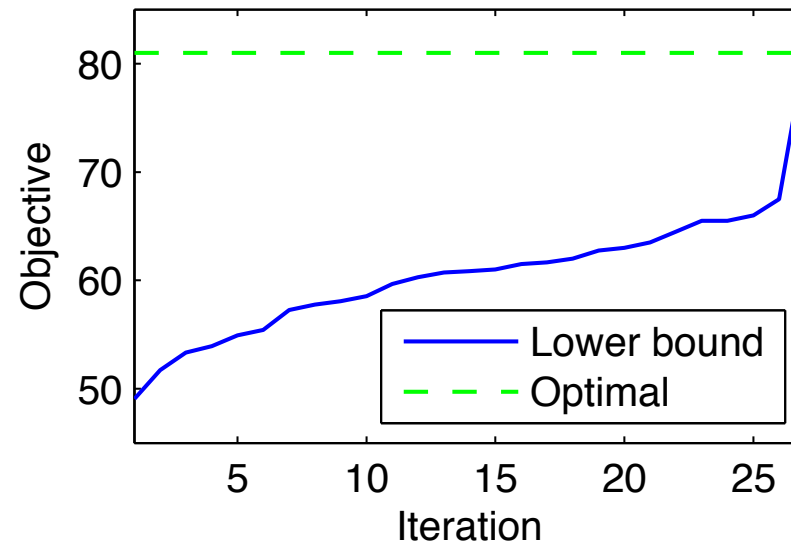
World's Hardest Sudoku

Could this be the toughest sudoku puzzle ever devised?

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

Illustration: sudoku

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...

Illustration: path planning

Want to plan a path from start to goal without hitting the obstacle

Represent path as a set of points $x^{(i)} \in \mathbb{R}^2, i = 1, \dots, m$ and minimize squared distance between points

Obstacle defined by $\ell \leq x^{(i)} \leq u$, for $a, b \in \mathbb{R}^2$

Constraint that we *not* hit obstacle given by condition

$$x_1^{(i)} \leq \ell_1 \vee x_2^{(i)} \leq \ell_2 \vee x_1^{(i)} \geq u_1 \vee x_2^{(i)} \geq u_2$$

How do we represent this “or” constraint as a (mixed) integer program?

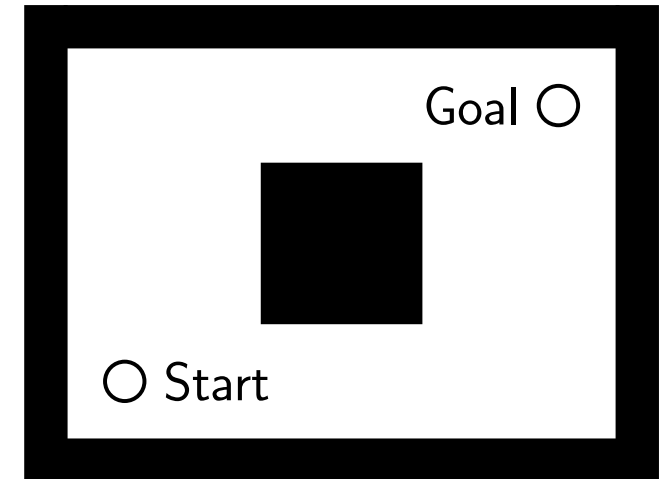


Illustration: path planning

The trick: “big-M” method

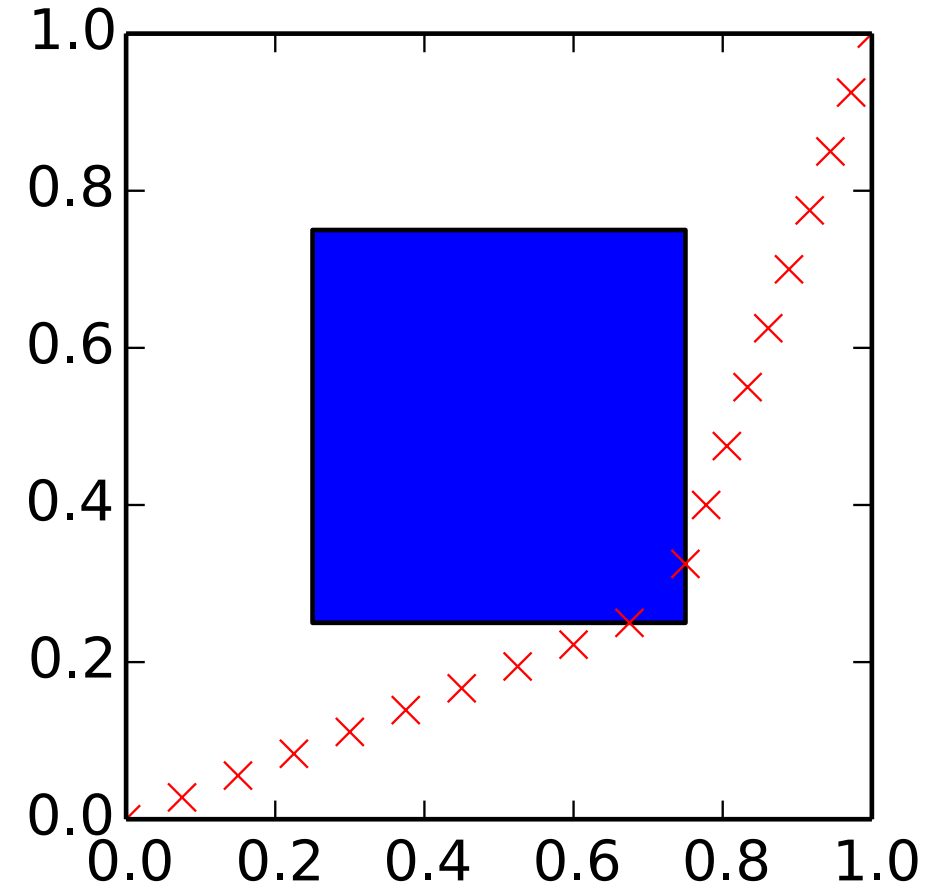
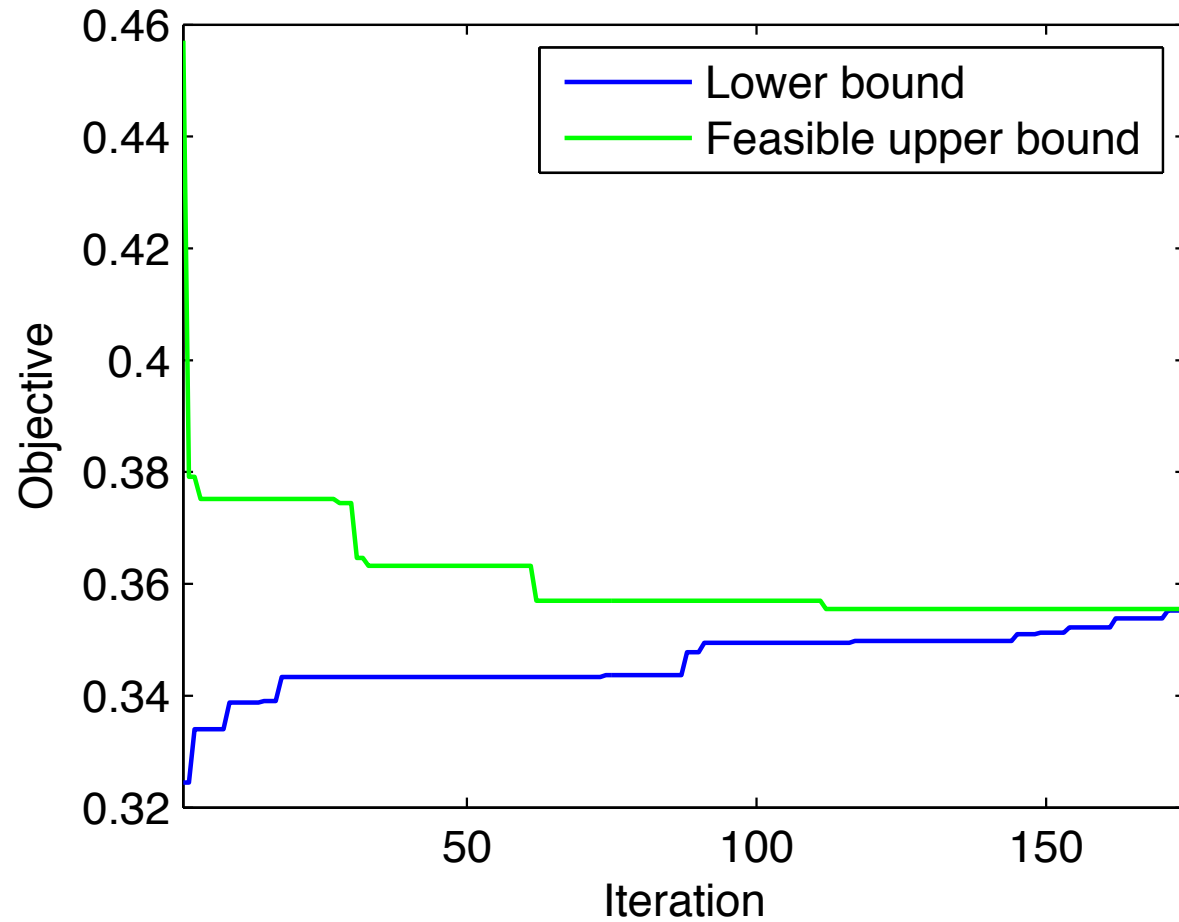
Let $M \in \mathbb{R}$ be some large number, let $z_1^{(i)} \in \{0,1\}$ and consider the constraint

$$x_1^{(i)} \leq \ell_1 + z_1^{(i)} M$$

If $z_1^{(i)} = 0$ this is the same as the original constraint, but if $z_1^{(i)} = 1$ this constraint will *always* be satisfied, so can write non-collision constraint as

$$\begin{aligned} x_1^{(i)} &\leq \ell_1 + z_1^{(i)} M, & x_2^{(i)} &\leq \ell_2 + z_2^{(i)} M \\ x_1^{(i)} &\geq u_1 + z_3^{(i)} M, & x_2^{(i)} &\geq u_2 + z_4^{(i)} M \\ z_1^{(i)} + z_2^{(i)} + z_3^{(i)} + z_4^{(i)} &\leq 3, & z^{(i)} &\in \{0,1\}^4 \end{aligned}$$

Illustration: path planning



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 Solve-Relaxation($\mathcal{C} \cup \{x_i \geq \lceil \tilde{x}_j \rceil\}$), Solve-Relaxation($\mathcal{C} \cup \{x_i \leq \lfloor \tilde{x}_j \rfloor\}$) to the frontier

Can deal with mixed integer problems (some non-integer variables), by simply not branching on non-integer variables, and by re-solving over non-integer variables after rounding integer variables

Cutting planes

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

CPLEX



IBM



GUROBI
OPTIMIZATION

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)