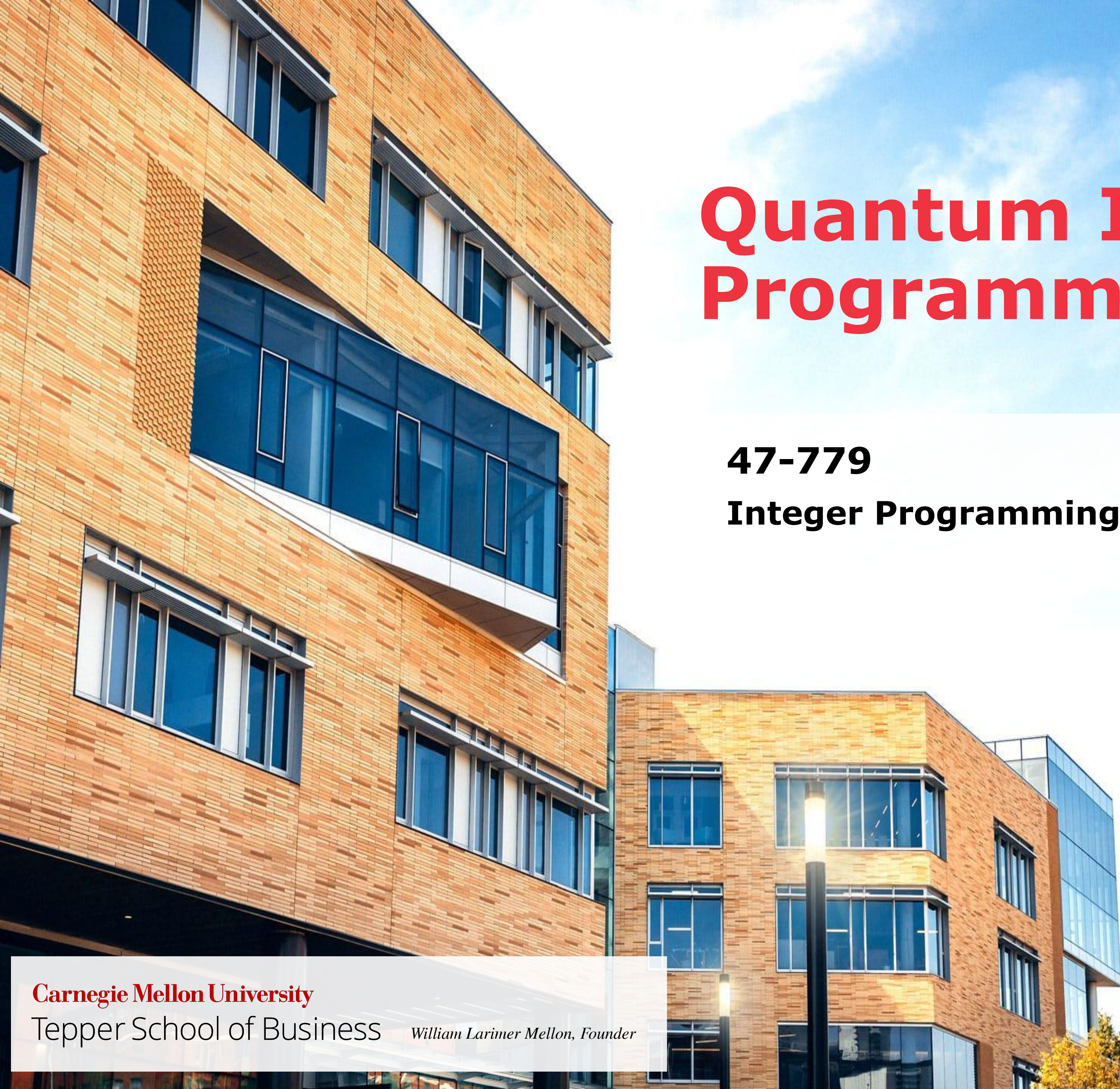


Quantum Integer Programming

47-779

Integer Programming

Carnegie Mellon University
Tepper School of Business
William Larimer Mellon, Founder

Agenda

- Constrained Optimization Models
- Linear Programming
  - Simplex and Interior Point methods
  - Duality brief overview
- (Mixed-)Integer Programming
  - Branch and bound and cutting planes
- (Mixed-)Integer Nonlinear Programming
- Nonlinearity
- Intro to Complexity

Constrained Optimization Models

- Exploration of real-world, looking for optimal solution can be time-consuming, expensive and prone to errors
- Instead we would like to have a model of the real-world
  - Represent our understanding of the real world
  - Incorporate assumptions and simplifications
  - Tradeoff between tractable and valid
- A useful paradigm is Mathematical Programming, where we write in mathematical equations
  - Objective(s)
  - Constraint(s)
    - All with respect to certain variables

$$\min_{x} f(x)$$
$$\text{s.t. } g(x) \leq 0$$

Operations Research approach

- Define the problem
- Formulate the model
  - Requirements
  - Simplifications
  - Assumptions
- Solve / Analyze the model
- Interpret the results

$$\min_{\mathrm{x}} f(x)$$
$$\mathrm{s.t.}\; g(x) \le 0$$

All steps are vital to provide a solution!

Simple example

Let’s propose a production plan that increases the profit of a company!

… we need more data than that.

The company only produces a finite set of products, each has its price. Besides, there are some production limitations.

To propose a model of this situation we need to identify certain key aspects of the problem.

- What are relevant parameters or data?
- Which decisions can we make? What is unknown? What is controllable?
- What limitations are relevant? What determines how a solution is valid (feasibility)?
- What is out goal?

Once those are clear, we can propose a model.

Simple example

After addressing those questions we reach the following problem statement.

Suppose there is a company that produces two different product, A and B, which can be sold at different values, \$5.5 and \$2.1 per unit respectively.
The company only counts with a single machine with electricity usage of at most 17kW/day; and producing each A and B consumes 8kW/day and 2kW/day, respectively.
Besides, the company can only produce at most 2 more units of A than B per day.

This is a valid model, but it would be easier to solve if we had a mathematical representation.
Assuming the units produced of A are x_1 and of B are x_2 we have

$$\begin{array}{ll} \max & 5.5x_1 + 2.1x_2 \\ & -x_1 + x_2 \leq 2 \\ & 8x_1 + 2x_2 \leq 17 \\ & x_1, x_2 \geq 0 \end{array}$$

[1] Adapted from Integer Programming (1st ed. 2014) by Michele Conforti, Gérard Cornuéjols, and Giacomo Zambelli

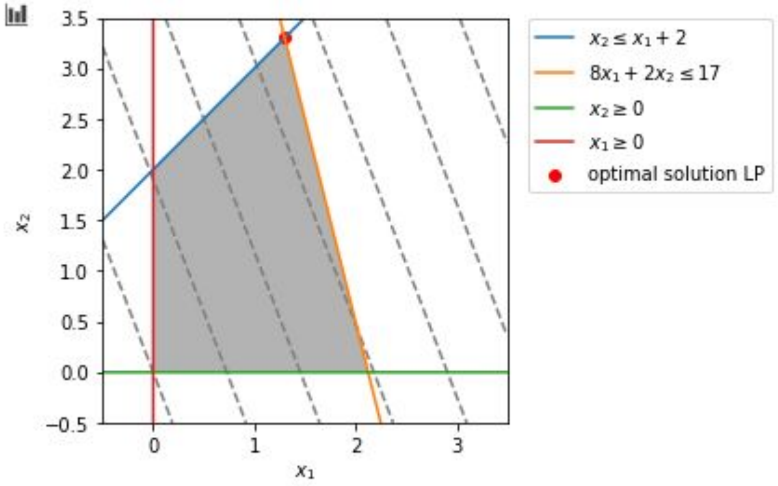
Linear Programming

A simplification of the general model presented before assumes that all the constraints and objective are linear, and the variables are continuous

$$\min_{\mathbf{x}} \mathbf{c}^\top \mathbf{x}$$
$$s.t.\ \mathbf{A}\mathbf{x} \leq \mathbf{b}$$

The feasible region of a Linear Program (LP) is a convex polyhedron

$$\begin{aligned} \max \quad & 5.5x_1 + 2.1x_2 \\ & -x_1 + x_2 \leq 2 \\ & 8x_1 + 2x_2 \leq 17 \\ & x_1, x_2 \geq 0 \end{aligned}$$



Interior-point methods

- Path through interior of polytope
- Polynomial time: $O(n^{3.5}L)$

Simplex methods

- Vertex hopping
- (Worst-case) Exponential time $O(2^n)$

Most solvers use simplex!

- 1e7 variables is tractable!

[1] Adapted from Integer Programming (1st ed. 2014) by Michele Conforti, Gérard Cornuéjols, and Giacomo Zambelli

Simplex (overview)

$$\min_{\mathbf{x}} \mathbf{c}^\top \mathbf{x}$$
$$s.t.\ \mathbf{A}\mathbf{x} = \mathbf{b}$$
$$\mathbf{x} \geq 0$$
$$\mathbf{x} \in \mathbb{R}^n$$

$$\Downarrow$$

$$\begin{bmatrix} 1 & -\mathbf{c}^\top & 0 \\ 0 & \mathbf{A} & \mathbf{b} \end{bmatrix}$$

$$\Downarrow$$

$$\begin{bmatrix} 1 & -\mathbf{c}_B^\top & -\mathbf{c}_N^\top & 0 \\ 0 & \mathbf{I} & \mathbf{D} & \mathbf{b} \end{bmatrix} \Rightarrow \begin{bmatrix} 1 & 0 & -\bar{\mathbf{c}}_N^\top & z_B \\ 0 & \mathbf{I} & \mathbf{D} & \mathbf{b} \end{bmatrix}$$

Relative costs

Objective value

Basic variables

Non-basic variables (assumed=0

Basic variables values

1. Write the “simplex tableau”
2. Convert to canonical form (select basis)
3. “Price out” basic variables
4. If solution can be improved we pivot (swap basic and non-basic variables)

Can efficiently restart from any feasible solution

[1] Adapted from Integer Programming (1st ed. 2014) by Michele Conforti, Gérard Cornuéjols, and Giacomo Zambelli

Linear Programming

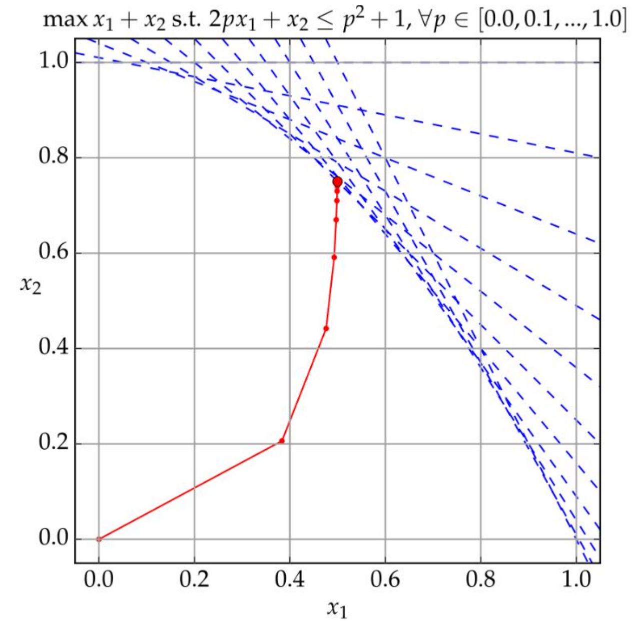
Modeling and Simplex method

Let’s jump to the code!
https://colab.research.google.com/github/bernalde/QuIP/blob/master/notebooks/Notebook%201%20-%20LP%20and%20IP.ipynb

Interior-point (brief overview)

- More details to it but the basics
- Intuition: starting from a feasible point, we approach the edges by having a monotonic barrier when close.
- Synonyms: Barrier method
- Not very efficient at restart
- **Very** useful when problems are **dual degenerate**

- **What is duality?**



[1] Adapted from Integer Programming (1st ed. 2014) by Michele Conforti, Gérard Cornuéjols, and Giacomo Zambelli
[2] https://en.wikipedia.org/wiki/Karmarkar%27s_algorithm

Linear Programming

Interior point method

Back to the code!
https://colab.research.google.com/github/bernalde/QuIP/blob/master/notebooks/Notebook%201%20-%20LP%20and%20IP.ipynb

Duality (very brief overview)

$$\min_{\mathbf{x}} \mathbf{c}^\top \mathbf{x}$$
$$s.t.\ \mathbf{A}\mathbf{x} = \mathbf{b}$$
$$\mathbf{x} \geq 0$$
$$\mathbf{x} \in \mathbb{R}^n$$

$$\min_{\mathbf{x}} \mathbf{c}^\top \mathbf{x} + \lambda^\top (\mathbf{b} - \mathbf{A}\mathbf{x})$$
$$s.t.\ \mathbf{x} \geq 0$$

Lagrangian or dual multipliers

$$\mathcal{L}(\lambda) = \min_{\mathbf{x} \geq 0} \mathbf{c}^\top \mathbf{x} + \lambda^\top (\mathbf{b} - \mathbf{A}\mathbf{x})$$
$$= \lambda^\top \mathbf{b} + \min_{\mathbf{x} \geq 0} (\mathbf{c}^\top - \lambda^\top \mathbf{A})\mathbf{x}$$
$$= \lambda^\top \mathbf{b} + \begin{cases} 0, & \text{if } \mathbf{c}^\top - \lambda^\top \mathbf{A} \geq \mathbf{0}^\top \\ -\infty, & \text{otherwise} \end{cases}$$

The Lagrangian function $\mathcal{L}(\lambda)$ (with deep meaning in classical mechanics and the least energy principle) defines a lower bound on our optimization problem, so we maximize it $\max_{\lambda} \mathcal{L}(\lambda) = \max_{\lambda} \lambda^\top \mathbf{b}$

$$s.t.\ \lambda^\top \mathbf{A} \leq \mathbf{c}^\top$$

For LPs (and in general convex problems) the following holds:

$$\text{Strong duality: } {\lambda^*}^\top \mathbf{b} = \mathbf{c}^\top \mathbf{x}^*$$

[1] Adapted from Integer Programming (1st ed. 2014) by Michele Conforti, Gérard Cornuéjols, and Giacomo Zambelli

LP - State of the art

Simplex LP solvers

27 Aug 2020

Benchmark of Simplex LP solvers

H. Mittelmann (mittelmann@asu.edu)

Logfiles of these runs at: plato.asu.edu/ftp/lp_logs/

This benchmark was run on a Linux-PC (i7-7700K, 4.2GHz, 32GB).
The MPS-datafiles for all testcases are in one of (see column "s")

miplib2010.zib.de/ [1]
plato.asu.edu/ftp/lptestset/ [2]
www.netlib.org/lp/data/ [3,7]
www.sztaki.hu/~meszaros/public_ftp/lptestset/
(MISC[4], PROBLEMATIC[5], STOCHLP[6], INFEAS[8])

NOTE: some files in [2-8] need to be expanded with emps in same directory!

The simplex methods were tested of the codes:

MOSEK-9.2.10	www.mosek.com
CLP-1.17.6	projects.coin-or.org/Clp
Google-GLOP	LP with Glop
SOPLEX-5.0.0	soplex.zib.de/
Gurobi-9.0.2	Gurobi
GLPK-4.65	www.gnu.org/software/glpk/glpk.html
MATLAB-R2020a	mathworks.com (dual-simplex)
COPT-1.4	COPT
MindOpt-0.9.0	MindOpt
HiGHS-1.0.0	HiGHS
SAS-OR-15.1	SAS (dual-simplex)

Scaled shifted (by 10 sec) geometric mean of runtimes

40 probs	MSK	CLP	GLOP	SPLX	Gurob	GLPK	MATL	COPT	MDOPT	HiGHS	SAS
	6.82	3.47	15.7	18.2	1.25	61.2	18.1	1	1.14	11.8	6.96
solved	38	40	35	39	40	31	33	40	40	37	37

Barrier LP solvers (Gurobi is ~3x faster than Mosek)

17 Aug 2020

Benchmark of Barrier LP solvers

H. Mittelmann (mittelmann@asu.edu)

Logfiles of these runs at: plato.asu.edu/ftp/lp_logs/

This benchmark was run on a Linux-PC (i7-7700K, 4.2GHz, 32GB).
The MPS-datafiles for all testcases are in one of (see column "s")

miplib2010.zib.de/ [1]
plato.asu.edu/ftp/lptestset/ [2]
www.netlib.org/lp/data/ [3,7]
www.sztaki.hu/~meszaros/public_ftp/lptestset/
(MISC[4], PROBLEMATIC[5], STOCHLP[6], INFEAS[8], NEW[9])

NOTE: Most files in [2-9] need to be expanded with emps in same directory!

The barrier methods were tested of:

MOSEK-9.2.10	www.mosek.com
MATLAB-R2020a	mathworks.com (interior-point, NO CROSSOVER!)
BPMPD-2.21:	NEOS-BPMPD run locally (NO CROSSOVER!)
CLP-1.17.6	projects.coin-or.org/Clp
SAS-OR-15.1:	SAS
Tulip-0.5.1:	Tulip (NO CROSSOVER!)
COPL_LP:	COPL_LP(binary), COPL_LP(Python)

Scaled shifted (by 10 sec) geometric mean of runtimes

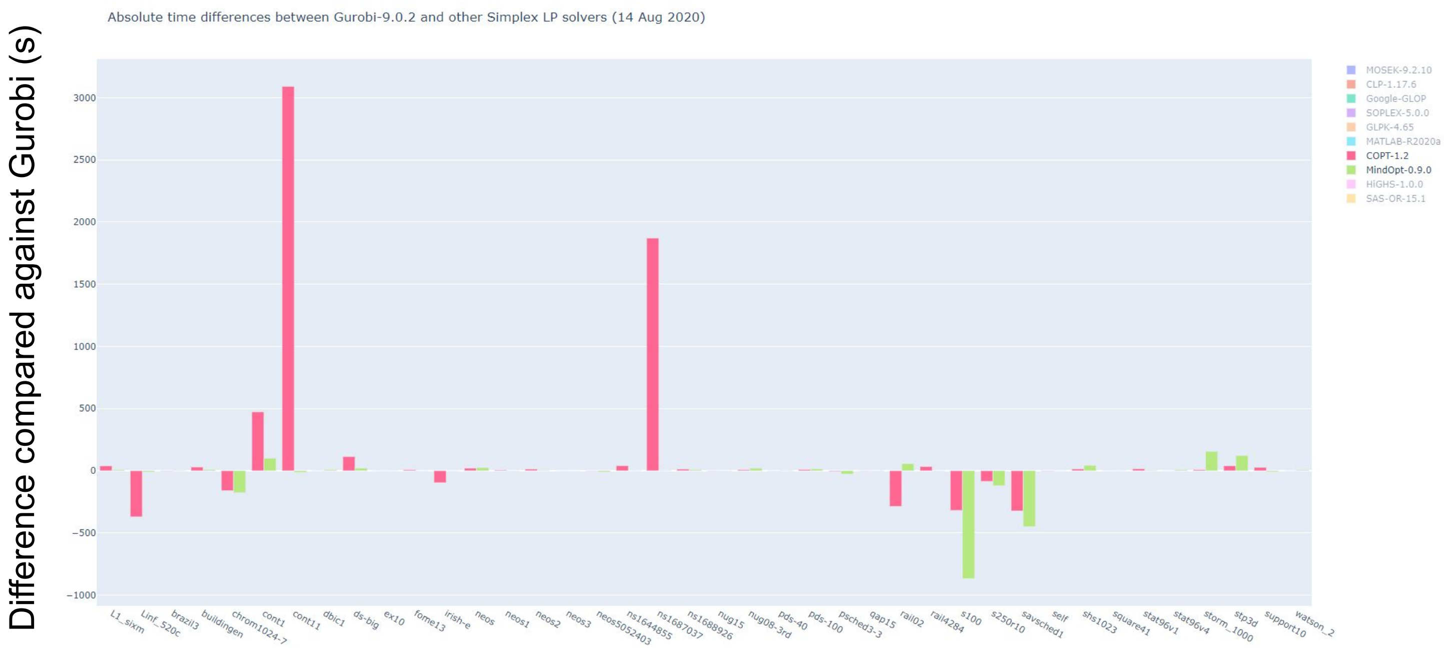
problem	MOSEK	MATLAB	BPMPD	CLP	SAS	TULIP	COPL_LP
45 probs	1	12.3	8.51	15.7	1.64	19.4	23.7
solved	44	35	32	39	44	34	37

[1] http://plato.asu.edu/bench.html
[2] http://plato.asu.edu/talks/informs2018.pdf

LP - State of the art

Previous results with 45 problems with ~205k constraints, ~225k variables and ~2.5M nonzeros.
The usual density of LP problems (in this case ~0.005% in average) is **very** low!
Best performing solvers used to be IBM CPLEX, FICO XPRESS and GUROBI

- They are not included in the benchmarks because of an incident in 2018
- but now players are in the arena! 🇨🇳



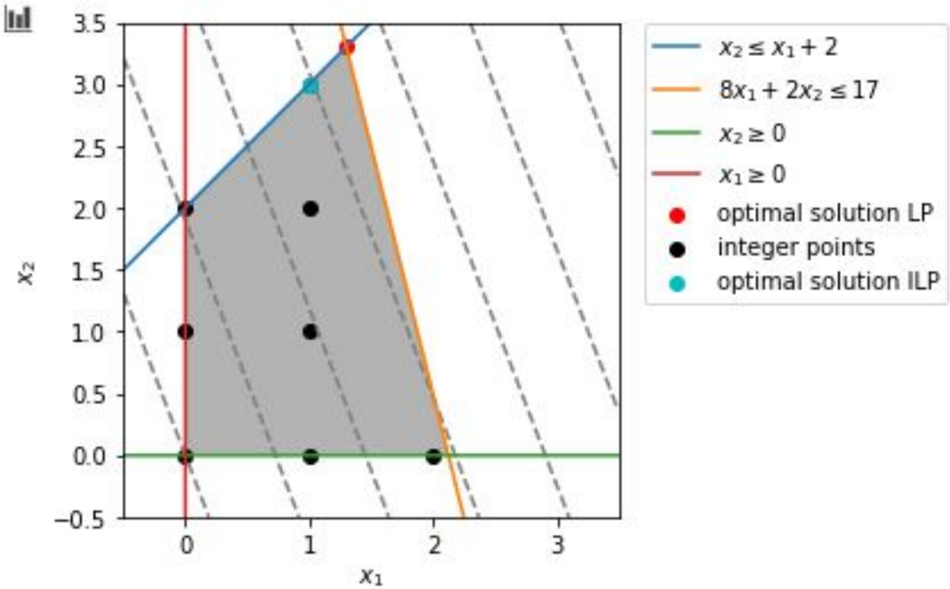
[1] http://plato.asu.edu/talks/euro2019.pdf
[2] https://mattmilten.github.io/mittelmann-plots/

Simple example - Continued

We found that the optimal solution was to produce 3.3 units of B, 1.3 units of A, and that would yield a profit of $14.08.

But what if we can only produce an integer number of products?
We modify our formulation to include this new information.

$$\begin{aligned} \max \quad & 5.5x_1 + 2.1x_2 \\ & -x_1 + \quad x_2 \leq 2 \\ & 8x_1 + \quad 2x_2 \leq 17 \\ & x_1, x_2 \geq 0 \\ & \boxed{x_1, x_2 \text{ integer}} \end{aligned}$$



The feasible region is no longer convex!

[1] Adapted from Integer Programming (1st ed. 2014) by Michele Conforti, Gérard Cornuéjols, and Giacomo Zambelli

Modeling non-convexities

The real-world is often abrupt, unexpected, sudden, discontinuous, non-smooth, …
This is the point where (Mixed-)Integer Programming comes into play!

$$\min_{x} f(x)$$
$$\text{s.t. } g(x) \leq 0$$
$$\text{some or all } x_i \text{ are integer}$$

Integer Programming can be understood as the universal tool for modeling non-convexities and discontinuities

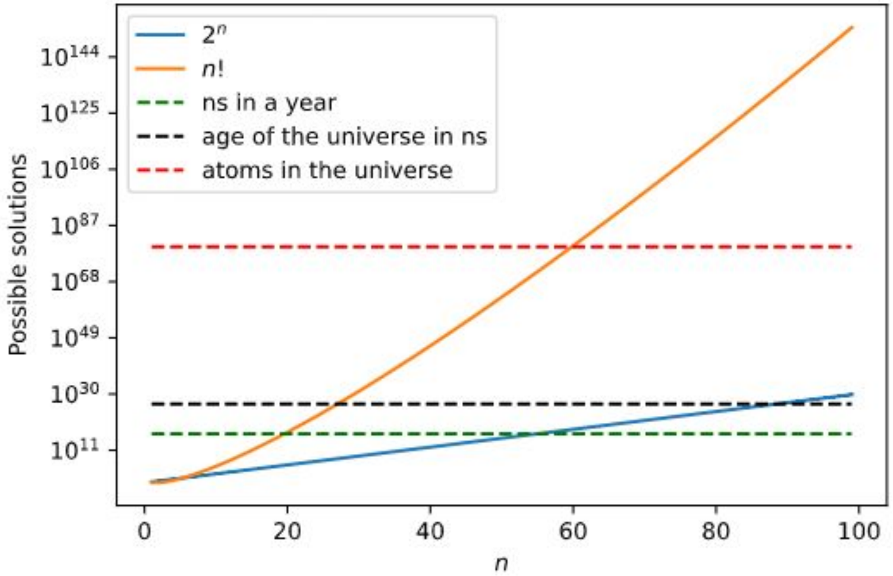
- Integrality condition may arise from indivisibility (people, objects)
- But it also can be used as a “trigger” or “switch”
  - Logical conditions such as disjunctions, implications, precedence can be modeled using this tool
- This is applicable to all areas of decision-making
  - ubiquitous, omnipresent [1]

[1] Adapted from Egon Balas’ lecture on Integer Programming

Enumerating

How hard can it be just to look at all the possible values, checking if they satisfy the constraints (being feasible) and comparing their objective function?

- Assuming only binary variables, the number of solutions grows as 2^n
- Many problems actually deal with permutations (assignments) therefore the number of solutions grow as $n!$



[1] Integer Programming (1st ed. 2014) by Michele Conforti, Gérard Cornuéjols, and Giacomo Zambelli

(Mixed-)Integer Programming

Solution and Enumeration

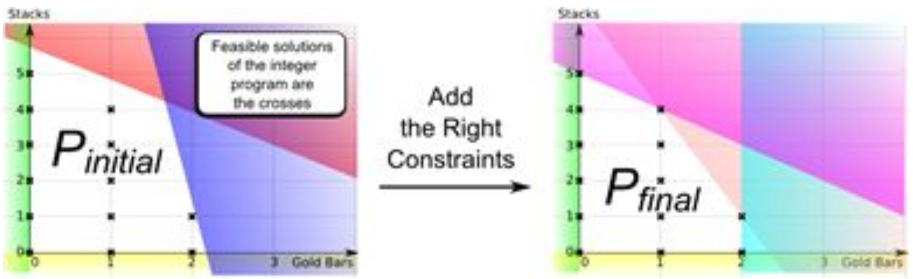
Back to the code!
https://colab.research.google.com/github/bernalde/QuIP/blob/master/notebooks/Notebook%201%20-%20LP%20and%20IP.ipynb

(Mixed-)Integer Programming

A Mixed-Integer Program (MIP) is an optimization problem of the form

$$\min_{x} f(x)$$
$$\text{s.t. } g(x) \leq 0$$
$$\text{some or all } x_i \text{ are integer}$$

Main concern is that is a strongly NP-complete problem



Branch-and-bound

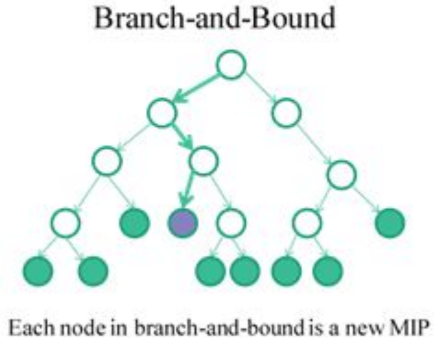
- Solution of each search node using linear programming

Cutting plane methods

- Polyhedral theory

Enhanced with constraint programming methods

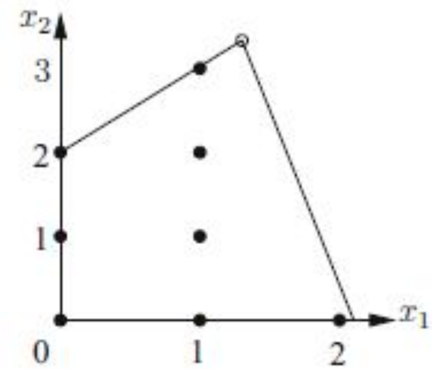
- Logic inference
- Domain reduction



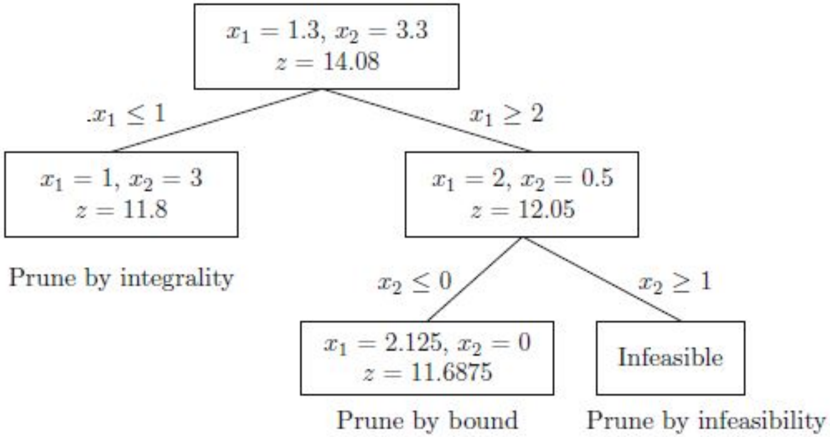
[1] https://www.ferc.gov/CalendarFiles/20100609110044-Bixby,%20Gurobi%20Optimization.pdf
[2] R. Kannan and C. L. Monma, On the computational complexity of integer programming problems, Lecture Notes in Economics and Mathematical Systems (Rudolf Henn, Bernhard Korte, and Werner Oettli, eds.), vol. 157, Springer-Verlag, 1978, pp. 161–172.
[3] https://www.slideshare.net/IBMOptimization/2013-11-informs12yearsofprogress

(Mixed-)Integer Programming

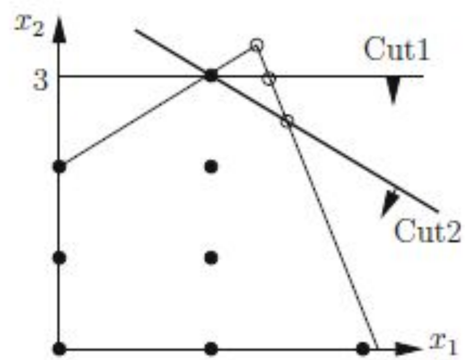
$$
\begin{aligned}
\max \quad & 5.5x_1 + 2.1x_2 \\
& -x_1 + \quad x_2 \le 2 \\
& 8x_1 + \ 2x_2 \le 17 \\
& x_1,\ x_2 \ge 0 \\
& x_1,\ x_2 \text{ integer}
\end{aligned}
$$



Branch-and-bound



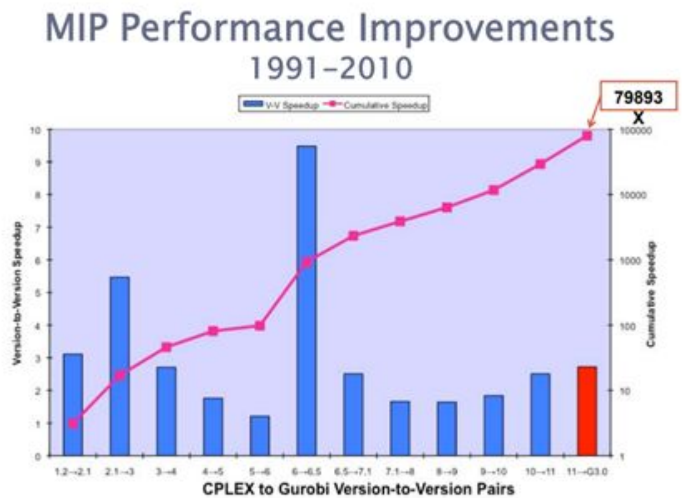
Cutting-plane methods



[1] Integer Programming (1st ed. 2014) by Michele Conforti, Gérard Cornuéjols, and Giacomo Zambelli

(Mixed-)Integer Programming

- Speedup between CPLEX 1.2 (1991) and CPLEX 11 (2007): 29,000 times
- Gurobi 1.0 (2009) comparable to CPLEX 11
- Speedup between Gurobi 1.0 and Gurobi 8.0 (2018): 91 times
- Total speedup 1991-2018: 2’600,000 times



- A MIP that would have taken 30 days to solve 27 years ago can now be solved in the same 25-year old computer in less than one second
- Hardware speed: 122.3 Pflops/s in 2018 vs. 59.7 Gflops/s in 1993 2’000,000 times
- Total speedup: **5.4 trillion times!**
- A MIP that would have taken 171,000 years to solve 27 years ago can now be solved in a modern computer in less than one second

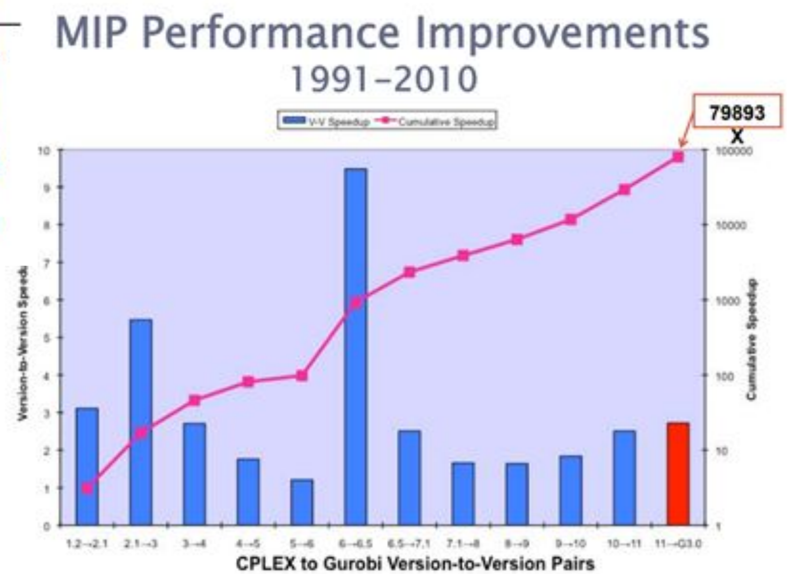
[1] https://www.ferc.gov/CalendarFiles/20100609110044-Bixby,%20Gurobi%20Optimization.pdf
[2] https://www.slideshare.net/IBMOptimization/2013-11-informs12yearsofprogress

(Mixed-)Integer Programming

Where is the improvement coming from

Feature	Speedup factor
Cuts	54
Preprocessing	11
Branching variable selection	3
Heuristics	1.5

Cut type	Speedup factor
Gomory mixed integer	2.5
Mixed integer rounding	1.8
Knapsack cover	1.4
Flow cover	1.2
Implied bounds	1.2
Path	1.04
Clique	1.02
GUB cover	1.02



[1] Integer Programming (1st ed. 2014) by Michele Conforti, Gérard Cornuéjols, and Giacomo Zambelli

(M)IP State of the art

Benchmark coming from MIPlib 2017, a 1065 collection of challenging MIPs that ranging from 1 to 19M of constraints and from 3 to 38M of variables.

Best commercial solvers are currently IBM CPLEX, FICO XPRESS and GUROBI.

- Focus on parallelization has been a center of research with many open questions there (usage of GPUs is not trivial for these algorithms)

```
19 May 2020          ==================================
                     The MIPLIB2017 Benchmark Instances
                     ==================================
                     H. Mittelmann (mittelmann@asu.edu)
```

The benchmark instances of MIPLIB2017 has been run by a number of codes. For pre-INFORMS2018 results see here

The following codes were run with a limit of 2 hours on the MIPLIB2017 benchmark set on two platforms.
1 thread: Intel i7-4790K, 4 cores, 32GB, 4GHz; 8 threads: Intel i7-5960X, 8 cores, 48GB, 3Ghz;

CBC-2.10.5: CBC
GLPK-4.65: www.gnu.org/software/glpk/glpk.html
LP_SOLVE-5.5.2: lpsolve.sourceforge.net/
MATLAB-2020a: MATLAB (intlinprog)
SAS-OR-15.1: SAS
(F)SCIP/spx-7.0.0: (Fiber)SCIP

Table for single thread, Result files per solver, Log files per solver

Table for 8 threads, Result files per solver, Log files per solver

++

Unscaled and scaled shifted geometric means of run times

```
All non-successes are counted as max-time.
The third line lists the number of problems (240 total) solved.
```

1 thr	CBC	GLPK	LP_SOL	MATLAB	SAS	SCIP
unscal	2107	5032	5335	3301	743	1100
scaled	2.84	6.77	7.18	4.44	1	1.48
solved	89	23	20	63	147	125

```
the best commercial solvers would have a geomean of about .2 to .3
```

8 thr	CBC	SAS	FSCIP
unscal	1723	580	1065
scaled	2.97	1	1.84
solved	98	157	138

```
the best commercial codes would have a geomean of about .2
```

[1] http://plato.asu.edu/bench.html
[2] http://miplib.zib.de/

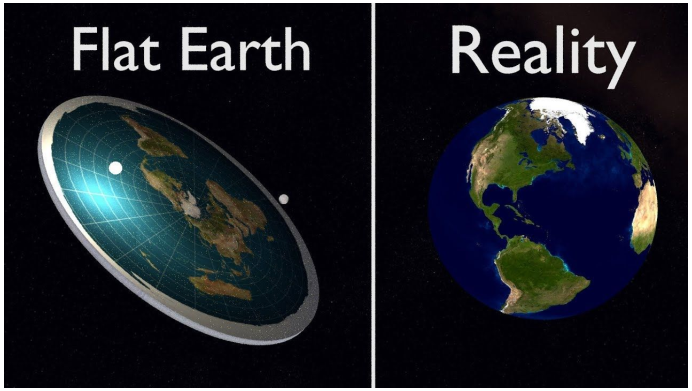
Modeling the real-world

The real-world is apart of abrupt, **nonlinear!**

$$\min_{x} f(x)$$
$$\text{s.t. } g(x) \le 0$$
$$\text{some or all } x_i \text{ are integer}$$

Although we can model discontinuities with integer variables, we can summarize more information using nonlinear constraints and objectives

- Assumption that $f(\mathbf{x}) = \mathbf{c}^\top \mathbf{x}, g(\mathbf{x}) = \mathbf{A}\mathbf{x} - \mathbf{b}$ does not hold in general
- This only makes our problem harder



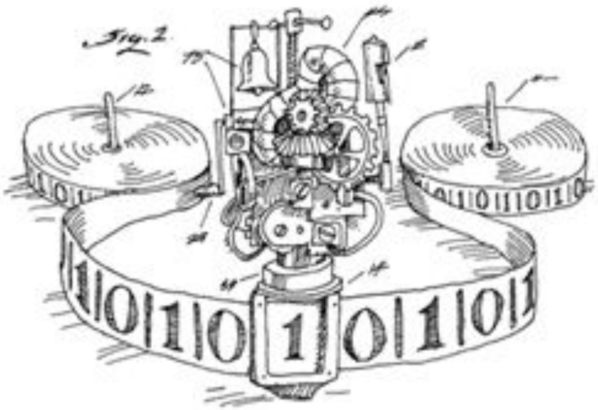
[1] https://www.vox.com/2018/4/28/17292244/flat-earthers-explain-philosophy

(Mixed-)Integer Nonlinear Programming

So far we have only discussed linear constraints and objective(s), but nonlinearity is key to modeling.

"***Most industrial processes*** *can be formulated as MINLP. Its expressive power is remarkable: it can encode any Turing Machine, including universal ones, such as Minsky's Recording Machine, which means that* ***every problem*** *can be formulated as a MINLP*.*"

- L. Liberti, Mathematical Programming. 2017



Representation of Turing Machine[1]

MINLP is **NP-Hard** since:

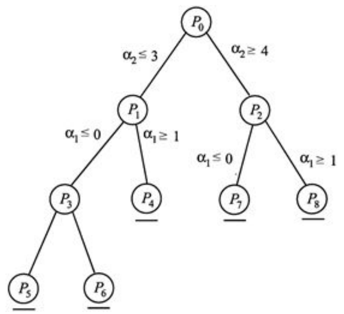
$$\text{SAT} \rightarrow \text{BIP} \subset \text{ILP} \subset \text{MILP} \subset \text{MINLP}$$

- Theorem 3 in Liberti, L. and Martinelli, F. "Mathematical programming: Turing completeness and applications to software analysis". 2014
- Problem 2, main Theorem in Karp, R.M. "Reducibility Among Combinatorial Problems". 1972

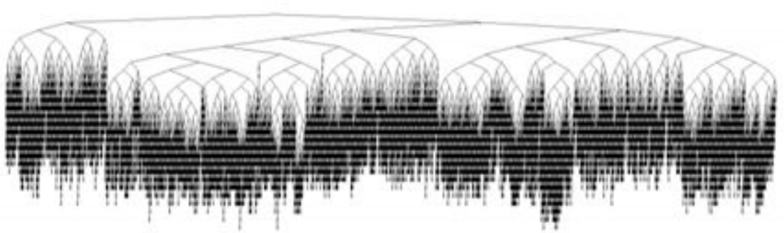
[1] COSC 545 - Theory of Computation, Georgetown University. Retrieved from http://people.cs.georgetown.edu/~cnewport/teaching/cosc545-spring14/ on 02/17/2019.

(Mixed-)Integer Nonlinear Programming

Branch-and-bound methods

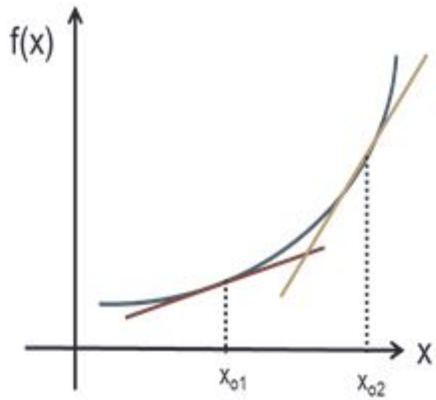


Representation of BB Tree

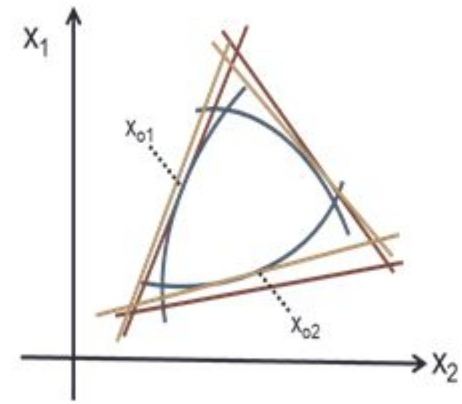


Actual BB Tree after 360s w/o preprocessing (~100k nodes)

(M)I Linear Programming based methods



Underestimate of objective function



Overestimate of the feasible region

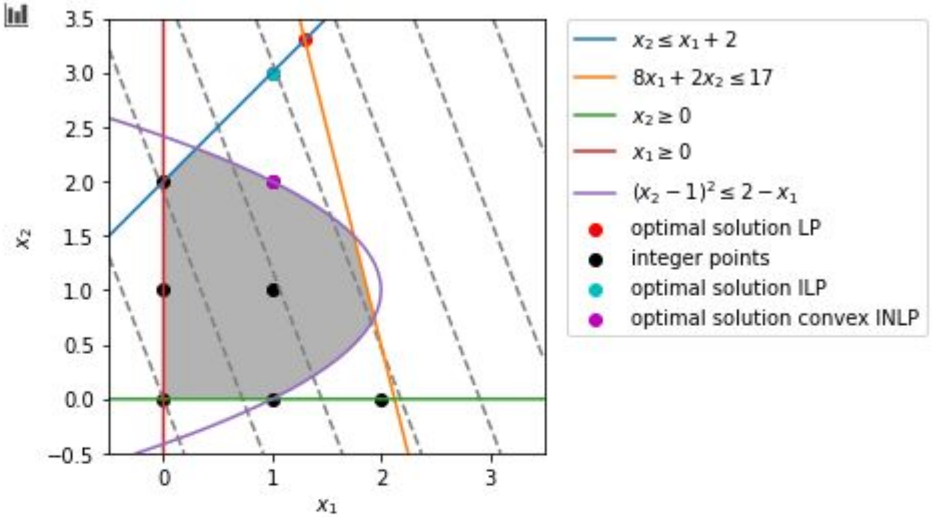
[1] Belotti, P., Kirches, C., Leyffer, S., S., Linderoth, J., Luedtke, J., and Mahajan, A. "Mixed-integer nonlinear optimization" 2012

Simple example - Continued

We found that the optimal solution of the IP was to produce 1 unit of A, and 3 units of B, and that would yield a profit of $11.8.

But what if we include an extra constraint, where the production of B minus 1 squared can only be smaller than 2 minus the production of A

$$\max_{x_1,x_2} 5.5x_1 + 2.1x_2$$
$$s.t.\ x_2 \le x_1 + 2$$
$$8x_1 + 2x_2 \le 17$$
$$\boxed{(x_2-1)^2 \le 2 - x_1}$$
$$x_1, x_2 \ge 0$$
$$x_1, x_2 \in \mathbb{Z}$$



The feasible region of the continuous relaxation is convex

[1] Adapted from Integer Programming (1st ed. 2014) by Michele Conforti, Gérard Cornuéjols, and Giacomo Zambelli

Convex (Mixed-)Integer Nonlinear Programming

Solution

Back to the code!
https://colab.research.google.com/github/bernalde/QuIP/blob/master/notebooks/Notebook%201%20-%20LP%20and%20IP.ipynb

Convex (M)INLP State of the art

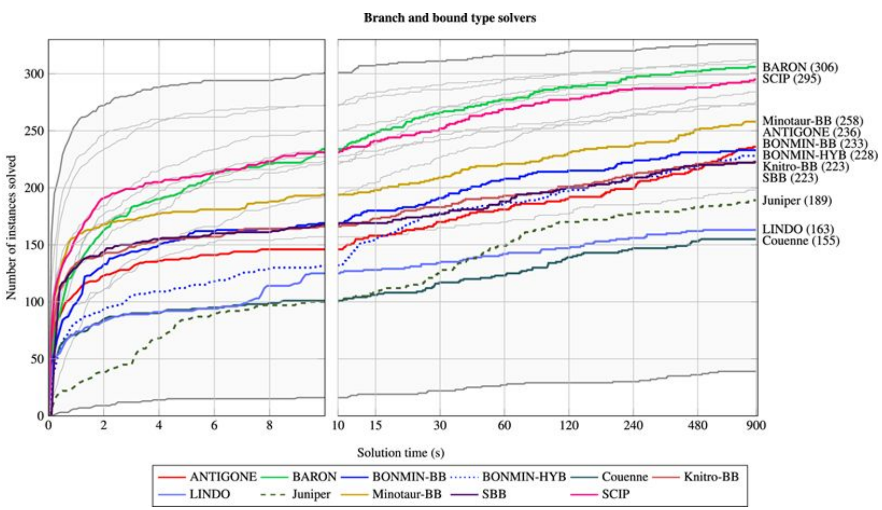
Complexity boundary does not lie between linearity and nonlinearity

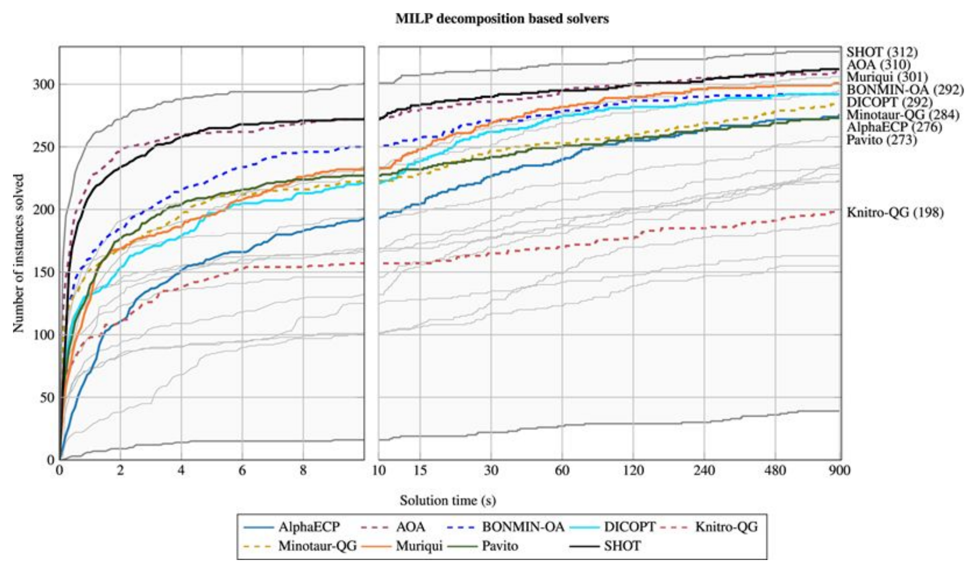
But between convexity and non-convexity.

Convex (M)INLP problems are more challenging but manageable

Benchmark from 335 problems with ~1000 variables and constraints in average

Branch-and-bound methods **(M)ILP based methods**





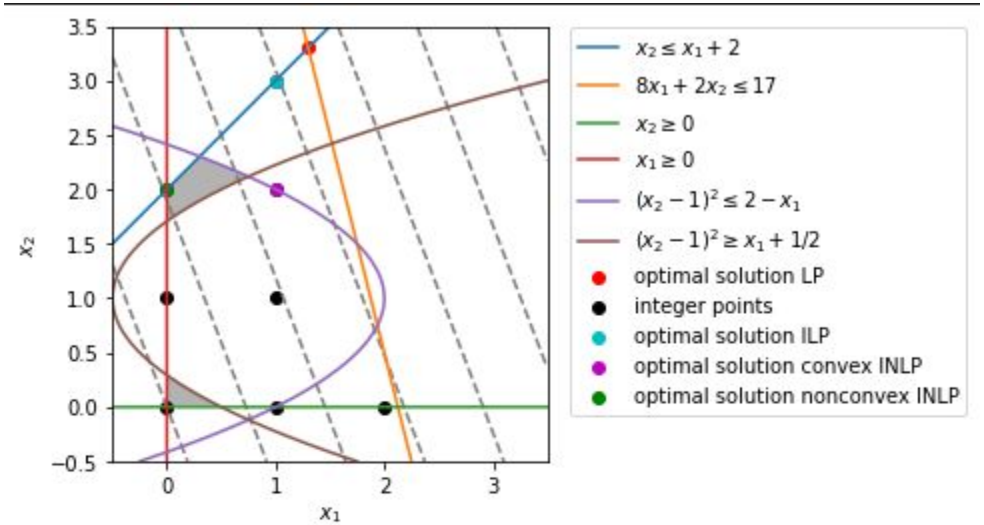
[1] Kronqvist, J., Bernal, D. E., Lundell, A. and Grossmann, I. E. [2018], `A review and comparison of solvers for convex MINLP', Optimization and Engineering pp. 1-59.

Simple example - Final version

We found that the optimal solution of the convex INLP was to produce 1 unit of A, and 2 units of B, and that would yield a profit of \$9.7.

Finally we include an extra constraint, where the production of B minus 1 squared can only be greater than 1/2 plus the production of A

$$
\begin{aligned}
\max_{x_1,x_2} \; & 5.5x_1 + 2.1x_2 \\
s.t.\; & x_2 \le x_1 + 2 \\
& 8x_1 + 2x_2 \le 17 \\
& (x_2-1)^2 \le 2 - x_1 \\
& \boxed{(x_2-1)^2 \ge 1/2 + x_1} \\
& x_1, x_2 \ge 0 \\
& x_1, x_2 \in \mathbb{Z}
\end{aligned}
$$



The feasible region of the continuous relaxation is non-convex!!!

[1] Adapted from Integer Programming (1st ed. 2014) by Michele Conforti, Gérard Cornuéjols, and Giacomo Zambelli

Non-Convex (Mixed-)Integer Nonlinear Programming

Solution

Back to the code!
https://colab.research.google.com/github/bernalde/QuIP/blob/master/notebooks/Notebook%201%20-%20LP%20and%20IP.ipynb

Nonconvex (M)NLIP State of the art

Benchmark coming from MINLPLib with instances with ~2500 variables and constraints.

BARON is the most performant solver with a geometric mean time to solve these instances of ~300 seconds.

Classical (M)IP solvers are increasing their capabilities to deal with nonlinearities

8 May 2020 ===
Mixed Integer Nonlinear Programming Benchmark (MINLPLIB)
===
H. Mittelmann (mittelmann@asu.edu)

The following codes were run through GAMS-31.0 with a limit of 2 hours on these instances from MINLPLIB and with one thread on an Intel i7-4790K, 32GB, 4GHz, available memory 20GB.

Description of selection process of benchmark instances. Statistics of the instances.

ANTIGONE, BARON, COUENNE, LINDO, SCIP

Table for all solvers, Result files per solver, Log files per solver, Trace files per solver, Error files per solver

++
Scaled and shifted geometric means of run times

Feasibility tolerance set to 1e-6. All non-successes are counted as max-time.
The second line lists the number of problems (87 total) solved.

The shifted geometric mean is computed on the 68 instances for which no solver failed.

	ANTIGONE	BARON	COUENNE	LINDO	SCIP
geom mean	4.42	1	9.33	6.49	2.89
solved	51	63	25	36	55

- IBM CPLEX can solve problems with convex quadratic constraints
- FICO XPRESS can solve nonlinear programs through linearization
- GUROBI can solve non-convex quadratic and bilinear programs

[1] http://plato.asu.edu/bench.html
[2] http://minlplib.org

Complexity introduction

When analyzing an algorithm we are interested in the amount of time and memory that it will take to solve it.

- To be safe this is usually answered in the **worst case scenario**
- The discussion here will be mainly the time complexity of algorithms.
- We will use the "big-O" notation where given two functions $f : S \to \mathbb{R}_+$ $g : S \to \mathbb{R}_+$, where S is an unbounded subset of $\mathbb{R}_+$we write that if there exists a positive real number M and $x_0 \in S$ such that $f(x) \leq Mg(x)$ for every $x > x_0$ then $f(x) = O(g(x))$
- For us S is going to be the set of instances or problems, and an *algorithm* is a procedure that will give a correct answer is a finite amount of time.
- An algorithm solves a problem in *polynomial time* if the function that measures its arithmetic operations $f : S \to \mathbb{R}_+$ is polynomially bounded by the function encoding the size of the problem $g : S \to \mathbb{R}_+$

[1] Adapted from Integer Programming (1st ed. 2014) by Michele Conforti, Gérard Cornuéjols, and Giacomo Zambelli

Complexity introduction

- If there exists an algorithm that solves a problem in polynomial time, the that problem becomes to the complexity class P
  - For example LP belongs to P because the interior point algorithm solves it in poly-time

- A decision problem is one with answer “yes” or “no”
- The complexity class NP, non-deterministic polynomial, is the class of all decision problems where the “yes”-answer can be verified in poly-time.

- If all the decision problems in NP can be reduced in poly-time to a problem *Q*, then *Q* is said to be NP-complete
  - “Is a (mixed-)integer linear set empty?” belongs to NP and is actually NP-complete

[1] Adapted from Integer Programming (1st ed. 2014) by Michele Conforti, Gérard Cornuéjols, and Giacomo Zambelli
[2] Cook, Stephen A. The Complexity of Theorem-Proving Procedures (1971)

Complexity introduction

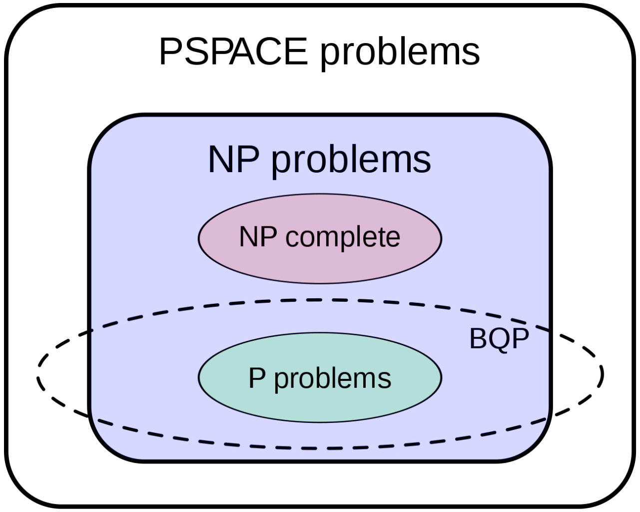
- A problem *Q* can be called NP-hard if all problems in NP can be reduced to *Q* in poly-time
  - Integer Programming is NP-hard
    - Since we can transform any NP problem into an integer program in poly-time, if there existed an algorithm to solve IP in poly-time, then we could solve any NP problem in poly-time: P=NP

- Integer programs with quadratic constraints are proved to be undecidable
  - Even after a long time without finding a solution, we cannot conclude none exists…
  - MINLP are **tough**!

[1] Adapted from Integer Programming (1st ed. 2014) by Michele Conforti, Gérard Cornuéjols, and Giacomo Zambelli
[2] Cook, Stephen A, The Complexity of Theorem-Proving Procedures (1971)

Complexity introduction

- A problem is said to belong to the complexity class BPP, bounded-error probabilistic polynomial time, if there is an algorithm that solves it such that
  - It is allowed to flip coins and make random decisions
  - It is guaranteed to run in polynomial time
  - On any given run of the algorithm, it has a probability of at most 1/3 of giving the wrong answer, whether the answer is YES or NO.
- There exists another complexity class called BQP, bounded-error quantum polynomial time, which is the quantum analogue of BPP
  - We hope that some problems belong to BQP and not to BPP to observe Quantum Advantage
    - E.g. Integer factorization



The suspected relationship of BQP to other problem spaces

[1] Michael Nielsen and Isaac Chuang (2000). Quantum Computation and Quantum Information. Cambridge: Cambridge University Press. ISBN 0-521-63503-9.

Take home message

Integer Programs lie at a very special intersection since they are:

- Interesting from an academic point of view
- Useful from a practical point of view
- Challenging from a computational point of view

We do not expect to observe Quantum Advantage by solving Integer Programs using Quantum Computers (but who knows right? Maybe P=BPP=BQP=NP)

We are still dealing with complicated problems that require answers, so we are going to try our best to solve them.

Welcome to Quantum Integer Programming!

More thoughts

Slide taken from Ed Rothberg (key developer of CPLEX and the RO in guRObi) on a talk of parallelization for (M)IP