Mixed-Integer Nonlinear Optimization

Martin Schmidt

@schmaidt

June 2021

JPOC Spring School on MINLPs and Bilevel Problems “in Paris”

At the end of this day you . . .

- know what an MINLP is
- can distinguish between convex and nonconvex MINLPs
- can apply standard MINLP modeling techniques
- know about and understand the classic algorithms for MINLP
- know the standard software tools for modeling MINLPs
- know the standard solvers that can be used to solve MINLPs

At the end of this day you . . .

- know what an MINLP is
- can distinguish between convex and nonconvex MINLPs
- can apply standard MINLP modeling techniques
- know about and understand the classic algorithms for MINLP
- know the standard software tools for modeling MINLPs
- know the standard solvers that can be used to solve MINLPs

I will teach principles, not formulas!

At the end of this day you . . .

- know what an MINLP is
- can distinguish between convex and nonconvex MINLPs
- can apply standard MINLP modeling techniques
- know about and understand the classic algorithms for MINLP
- know the standard software tools for modeling MINLPs
- know the standard solvers that can be used to solve MINLPs

I will teach principles, not formulas!

You will not remember the last ε,
but I hope you remember the core ideas!

There should be no crying in this compact course!

Overview

1. Introduction
2. Algorithms for Convex MINLP
3. MILP-Based Reformulations
4. Nonconvex MINLP
5. Modeling Languages
6. Solvers
7. What Else?
8. Literature

1. Introduction: Overview

What is Optimization Anyway?

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & f(x) \\ \text{s.t.} \quad & g_i(x) \geq 0, \quad i \in I = \{1, \dots, m\} \\ & h_j(x) = 0, \quad j \in J = \{1, \dots, p\} \end{aligned}$$

- x: vector of variables/decisions
- $f : \mathbb{R}^n \to \mathbb{R}$: objective function
- $g_i : \mathbb{R}^n \to \mathbb{R}$: inequality constraints
- $h_j : \mathbb{R}^n \to \mathbb{R}$: equality constraints

What is Optimization Anyway?

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & f(x) \\ \text{s.t.} \quad & g_i(x) \geq 0, \quad i \in I = \{1, \ldots, m\} \\ & h_j(x) = 0, \quad j \in J = \{1, \ldots, p\} \end{aligned}$$

- x: vector of variables/decisions
- $f : \mathbb{R}^n \to \mathbb{R}$: objective function
- $g_i : \mathbb{R}^n \to \mathbb{R}$: inequality constraints
- $h_j : \mathbb{R}^n \to \mathbb{R}$: equality constraints

Feasible Set

$$\Omega = \{x \in \mathbb{R}^n \colon g_i(x) \geq 0,\ i \in I,\ h_j(x) = 0,\ j \in J\}$$

Is it hard?

. . . it depends!

Is it hard?

. . . it depends!

- Are all functions linear?
- Are some of them nonlinear?
- Is the objective function convex?
- Is the feasible set convex?
- Are the variables continuous-valued?
- Do we have integer variables?
- Are the functions differentiable?

Convexity is Crucial

"The great watershed in optimization isn't between linearity and nonlinearity, but convexity and nonconvexity."

— R. Tyrrell Rockafellar

Mixed-Integer Nonlinear Optimization

We consider MINLPs of the form

$$\begin{aligned}\min_{x\in\mathbb{R}^n}\quad & f(x)\\ \text{s.t.}\quad & c(x)\le 0\\ & x\in X\\ & x_i\in\mathbb{Z},\quad i\in I\end{aligned}$$

- $f : \mathbb{R}^n \to \mathbb{R}$ and $c : \mathbb{R}^n \to \mathbb{R}^m$ are twice continuously differentiable
- $X \subset \mathbb{R}^n$ is a bounded polyhedral set, i.e.,

  $$X = \{x \in \mathbb{R}^n \colon l \le Ax \le u\}$$

  for some matrix A and some vectors l, u
- $I \subseteq \{1, \ldots, n\}$ is the index set of integer variables

Mixed-Integer Nonlinear Optimization

We consider MINLPs of the form

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & f(x) \\ \text{s.t.} \quad & c(x) \leq 0 \\ & x \in X \\ & x_i \in \mathbb{Z}, \quad i \in I \end{aligned}$$

- $f : \mathbb{R}^n \to \mathbb{R}$ and $c : \mathbb{R}^n \to \mathbb{R}^m$ are twice continuously differentiable
- $X \subset \mathbb{R}^n$ is a bounded polyhedral set, i.e.,

  $$X = \{x \in \mathbb{R}^n \colon l \leq Ax \leq u\}$$

  for some matrix A and some vectors l, u
- $I \subseteq \{1, \dots, n\}$ is the index set of integer variables

This also contains maximization problems, equality constraints, simple variable bounds, and more general discrete sets (later more . . .)

Is this important?

Yes!

Is this important?

Yes!

MINLPs are everywhere! We will see some examples soon.

Is this important?

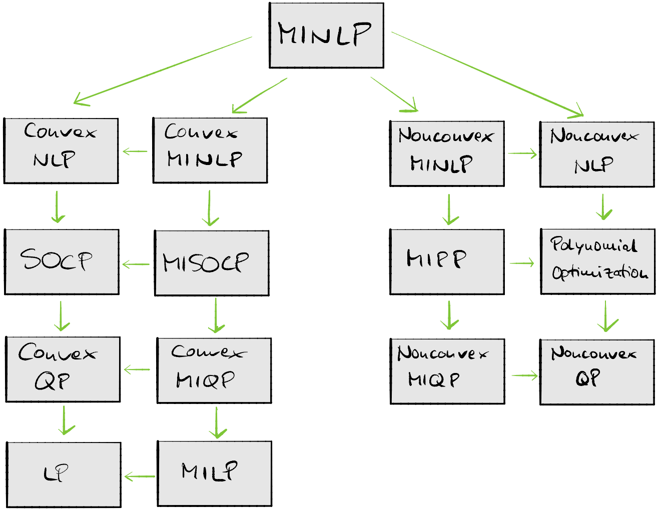
Yes!

MINLPs are everywhere! We will see some examples soon.

The problem class of MINLPs includes . . .

- nonlinear problems (NLPs),
- quadratic problems (QPs),
- linear problems (LPs),
- mixed-integer linear problems (MILPs),
- . . .

The MINLP Tree



Convex and Nonconvex MINLPs

“*The great watershed in optimization isn’t between linearity and nonlinearity, but convexity and nonconvexity.*”

— R. Tyrrell Rockafellar

Definition

An optimization problem is *convex* if

- the feasible set is convex
- and if the objective function is convex on the feasible set.

Convex and Nonconvex MINLPs

“*The great watershed in optimization isn’t between linearity and nonlinearity, but convexity and nonconvexity.*”

— R. Tyrrell Rockafellar

Definition

An optimization problem is *convex* if

- the feasible set is convex
- and if the objective function is convex on the feasible set.

Well ...

- But mixed-integer feasible sets are always nonconvex!
- So there are no “convex MINLPs”?

Definition

The MINLP

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & f(x) \\ \text{s.t.} \quad & c(x) \leq 0 \\ & x \in X \\ & x_i \in \mathbb{Z}, \quad i \in I \end{aligned}$$

is called *convex* if the objective function f and the constraint function c are convex functions. If the objective function or at least one of the constraints are nonconvex, the problem is called a *nonconvex MINLP*.

Convex and Nonconvex MINLPs

This means that the MINLP is called convex if the *NLP relaxation*

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & f(x) \\ \text{s.t.} \quad & c(x) \le 0 \\ & x \in X \end{aligned}$$

is a convex optimization problem.

- To obtain a convex feasible set using "$c(x) \le 0$", c needs to be convex.
- For inequalities "$c(x) \ge 0$", c needs to be concave to lead to a convex feasible set.

Hardness of MINLP

MINLP combines challenges of handling nonlinearities with the combinatorial explosion due to integer variables!

Hardness of MINLP

MINLP combines challenges of handling nonlinearities with the combinatorial explosion due to integer variables!

- It is an NP-hard combinatorial problem
  - ... because it includes MILP (Kannan and Monma 1978)
- Even worse, nonconvex integer optimization problems are in general undecidable (Jeroslow 1973)
  - Jeroslow: example of a quadratically constrained integer program
  - Theorem: no computing device exists that can compute the optimum for all problems in this class

Making it "practable"

Assumption

- X is compact, i.e., a polytope

It's still NP-hard, but . . .

Making it "practable"

Assumption

- X is compact, i.e., a polytope

It's still NP-hard, but . . .

Theorem

Suppose that the set Ω is non-empty and compact and that the function $f : \Omega \to \mathbb{R}$ is continuous. Then, f has at least one global minimizer and at least one global maximizer.

Is it harder than MILP?

- MINLP is NP-hard since it includes mixed-integer linear programming (MILP).
- Question: Is it harder?

Is it harder than MILP?

- MINLP is NP-hard since it includes mixed-integer linear programming (MILP).
- Question: Is it harder? Somehow, yes!

Definition

Let $S \subset \mathbb{R}^n$ be any set. The *convex hull* of S is the set

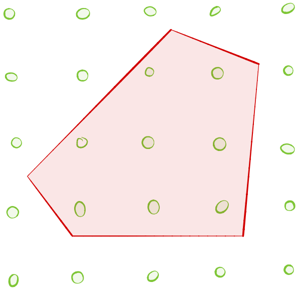
$$\text{conv}(S) := \{z \in \mathbb{R}^n \colon z = \lambda x + (1-\lambda)y \text{ for all } \lambda \in [0,1],\ x, y \in S\}.$$

Is it harder than MILP?

- The convex hull is crucial in mixed-integer linear programming.
- Linear Optimization 101: A linear problem obtains a solution at a vertex of the feasible set.
- Thus: We can solve the MILP by solving the LP over the convex hull of the MILP's integer-feasible points.

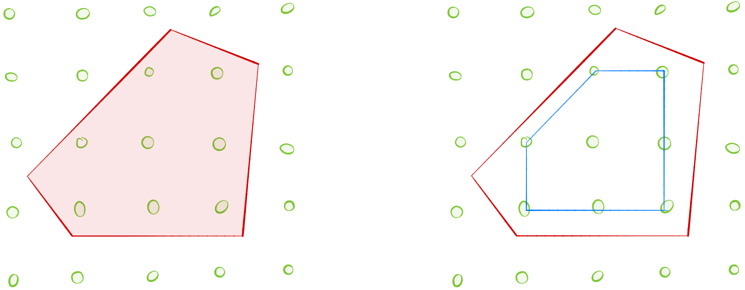
Is it harder than MILP?

- The convex hull is crucial in mixed-integer linear programming.
- Linear Optimization 101: A linear problem obtains a solution at a vertex of the feasible set.
- Thus: We can solve the MILP by solving the LP over the convex hull of the MILP's integer-feasible points.

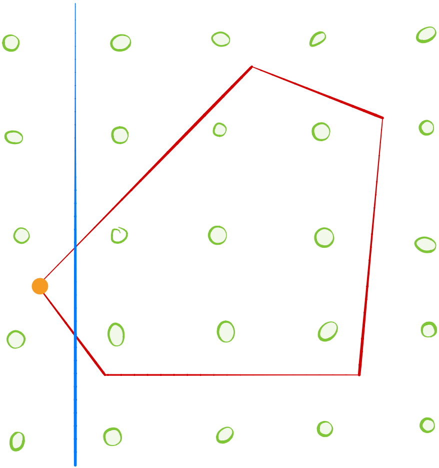


Is it harder than MILP?

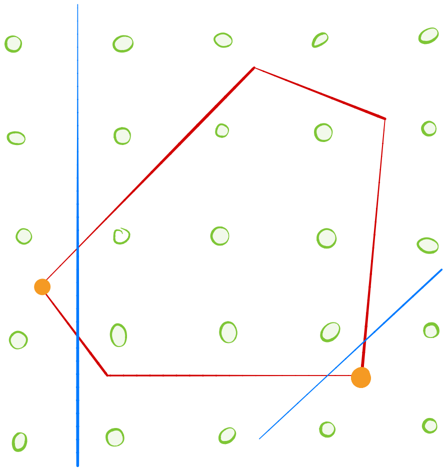
- The convex hull is crucial in mixed-integer linear programming.
- Linear Optimization 101: A linear problem obtains a solution at a vertex of the feasible set.
- Thus: We can solve the MILP by solving the LP over the convex hull of the MILP's integer-feasible points.



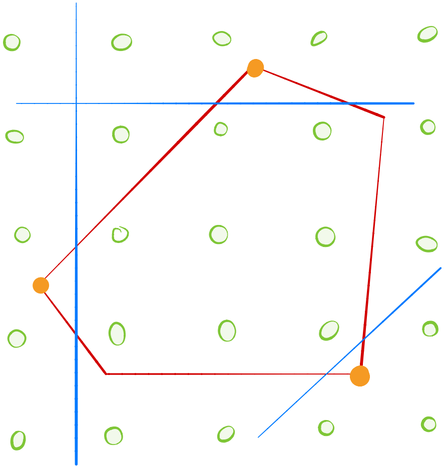
MILP: Separation = Optimization!



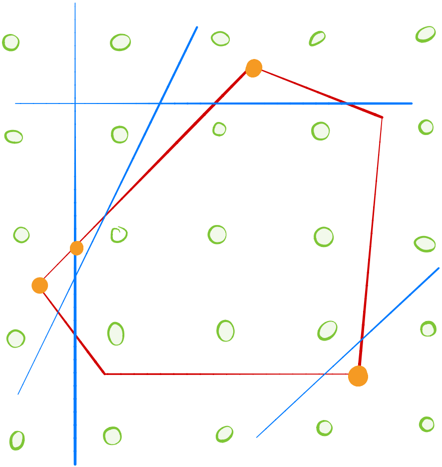
MILP: Separation = Optimization!



MILP: Separation = Optimization!



MILP: Separation = Optimization!

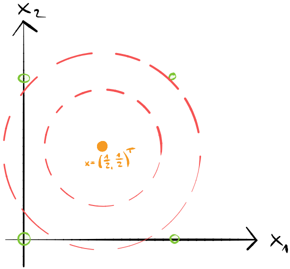


MINLP: Separation = Optimization? No!

Consider the MINLP

$$\min_{x \in \mathbb{R}^n} \quad \sum_{i=1}^{n} \left(x_i - \frac{1}{2} \right)^2$$

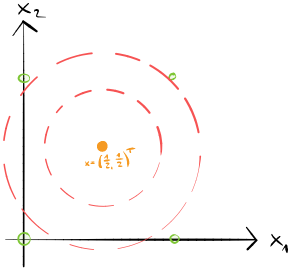
$$\text{s.t.} \quad x_i \in \{0, 1\}, \quad i = 1, \dots, n$$



MINLP: Separation = Optimization? No!

Consider the MINLP

$$\min_{x \in \mathbb{R}^n} \quad \sum_{i=1}^{n} \left(x_i - \frac{1}{2} \right)^2$$
$$\text{s.t.} \quad x_i \in \{0, 1\}, \quad i = 1, \ldots, n$$



- The solution of the continuous relaxation is

$$x = \left(\frac{1}{2}, \ldots, \frac{1}{2} \right)^\top$$

- This is not an extreme point of the feasible set of the continuous relaxation
- Even worse: it lies in the strict interior of the convex hull of the feasible set of the MINLP
- Thus: It cannot be separated!

A Potential Remedy: The Epigraph Formulation

The original MINLP

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & f(x) \\ \text{s.t.} \quad & c(x) \le 0 \\ & x \in X \\ & x_i \in \mathbb{Z}, \quad i \in I \end{aligned}$$

and its epigraph reformulation

$$\begin{aligned} \min_{x \in \mathbb{R}^n, \eta \in \mathbb{R}} \quad & \eta \\ \text{s.t.} \quad & f(x) \le \eta \\ & c(x) \le 0 \\ & x \in X \\ & x_i \in \mathbb{Z}, \quad i \in I \end{aligned}$$

are equivalent and the optimal solutions of the latter always lie on the boundary of the convex hull of the feasible set.

Subset Selection in Linear Regression: An MINLP?

- Given: m data points $x_i \in \mathbb{R}^d$ and $y_i \in \mathbb{R}$, $i = 1, \ldots, m$
- Task: Find $\beta \in \mathbb{R}^d$ such that

$$\sum_{i=1}^{m} \left(y_i - x_i^\top \beta \right)^2$$

is minimized while limiting the cardinality of β to K.

Subset Selection in Linear Regression: An MINLP?

- Given: m data points $x_i \in \mathbb{R}^d$ and $y_i \in \mathbb{R}$, $i = 1, \ldots, m$
- Task: Find $\beta \in \mathbb{R}^d$ such that

$$\sum_{i=1}^{m} \left(y_i - x_i^\top \beta \right)^2$$

is minimized while limiting the cardinality of β to K.

Model (Bertsimas, R. Shioda 2009)

$$\begin{aligned} \min_{\beta \in \mathbb{R}^d} \quad & \sum_{i=1}^{m} \left(y_i - \sum_{j=1}^{d} x_{ij} \beta_j \right)^2 \\ \text{s.t.} \quad & |\text{supp}(\beta)| \leq K \end{aligned}$$

Is this already an MINLP?

Subset Selection in Linear Regression: An MINLP?

- Given: m data points $x_i \in \mathbb{R}^d$ and $y_i \in \mathbb{R}$, $i = 1, \ldots, m$
- Task: Find $\beta \in \mathbb{R}^d$ such that

$$\sum_{i=1}^{m} \left(y_i - x_i^\top \beta \right)^2$$

is minimized while limiting the cardinality of β to K.

Model (Bertsimas, R. Shioda 2009)

$$\begin{aligned} \min_{\beta \in \mathbb{R}^d} \quad & \sum_{i=1}^{m} \left(y_i - \sum_{j=1}^{d} x_{ij} \beta_j \right)^2 \\ \text{s.t.} \quad & |\text{supp}(\beta)| \le K \end{aligned}$$

Is this already an MINLP? No! But ...

Subset Selection in Linear Regression: An MINLP?

- Introduce a binary variable $z_j \in \{0, 1\}$ for every entry β_j, $j = 1, \ldots, d$, in the β vector:

$$z_j = \begin{cases} 1, & \beta_j \text{ can be used} \\ 0, & \beta_j = 0 \end{cases}$$

- Count the used β_j's by counting the binary variables

$$\begin{aligned} \min_{\beta \in \mathbb{R}^d} \quad & \sum_{i=1}^{m} \left(y_i - \sum_{j=1}^{d} x_{ij} \beta_j \right)^2 \\ \text{s.t.} \quad & \sum_{j=1}^{d} z_j \le K \\ & M_j^l z_j \le \beta_j \le M_j^u z_j, \quad j = 1, \ldots, d \\ & z_j \in \{0, 1\}, \quad j = 1, \ldots, d \end{aligned}$$

$$\begin{aligned}
\min_{\beta \in \mathbb{R}^d} \quad & \sum_{i=1}^{m} \left(y_i - \sum_{j=1}^{d} x_{ij}\beta_j \right)^2 \\
\text{s.t.} \quad & \sum_{j=1}^{d} z_j \leq K \\
& M_j^l z_j \leq \beta_j \leq M_j^u z_j, \quad j = 1, \ldots, d \\
& z_j \in \{0, 1\}, \quad j = 1, \ldots, d
\end{aligned}$$

MINLP 101

Know the convexity properties of your instance!

Subset Selection in Linear Regression

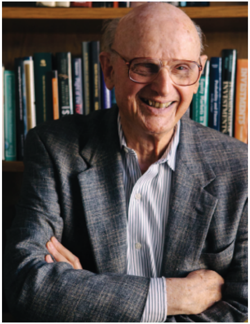
$$\begin{aligned}
\min_{\beta \in \mathbb{R}^d} \quad & \sum_{i=1}^{m} \left(y_i - \sum_{j=1}^{d} x_{ij}\beta_j \right)^2 \\
\text{s.t.} \quad & \sum_{j=1}^{d} z_j \leq K \\
& M_j^l z_j \leq \beta_j \leq M_j^u z_j, \quad j = 1, \ldots, d \\
& z_j \in \{0, 1\}, \quad j = 1, \ldots, d
\end{aligned}$$

MINLP 101

Know the convexity properties of your instance!

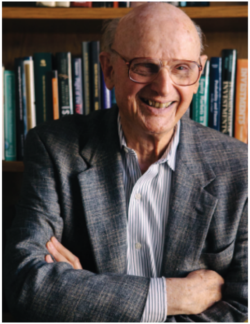
- This is a convex MINLP
- All constraints are linear, i.e., the feasible set is polyhedral and thus convex
- Objective function is convex in β

Portfolio Optimization



- Markowitz (1952)
- n possibly risky assets
- mean return vector $\mu \in \mathbb{R}^n$
- covariance return matrix $\Sigma \in \mathbb{R}^{n \times n}$
- minimum portfolio return $R > 0$
- vector of ones $e \in \mathbb{R}^n$

Portfolio Optimization



- Markowitz (1952)
- n possibly risky assets
- mean return vector $\mu \in \mathbb{R}^n$
- covariance return matrix $\Sigma \in \mathbb{R}^{n \times n}$
- minimum portfolio return $R > 0$
- vector of ones $e \in \mathbb{R}^n$

$$\min_{x \in \mathbb{R}^n} \quad x^\top \Sigma x$$
$$\text{s.t.} \quad \mu^\top x \geq R, \; e^\top x = 1, \; x \geq 0$$

Cardinality-Constrained Portfolio Optimization

Let K be the maximum number of assets that can be included in the portfolio

$$\begin{aligned}
\min_{x \in \mathbb{R}^n, z \in \mathbb{R}^n} \quad & x^\top \Sigma x \\
\text{s.t.} \quad & \mu^\top x \geq R,\ e^\top x = 1,\ x \geq 0 \\
& 0 \leq x_i \leq M_i^u z_i, \quad i = 1, \dots, n \\
& \sum_{i=1}^{n} z_i \leq K
\end{aligned}$$

Cardinality-Constrained Portfolio Optimization

Let K be the maximum number of assets that can be included in the portfolio

$$\begin{aligned}
\min_{x\in\mathbb{R}^n, z\in\mathbb{R}^n} \quad & x^\top \Sigma x \\
\text{s.t.} \quad & \mu^\top x \geq R,\ e^\top x = 1,\ x \geq 0 \\
& 0 \leq x_i \leq M_i^u z_i, \quad i = 1, \ldots, n \\
& \sum_{i=1}^{n} z_i \leq K
\end{aligned}$$

We already used a standard modeling trick twice: big-M constraints

k-Means Clustering: Setting

- Let $X \in \mathbb{R}^{p \times n}$ be the matrix containing the data set
- Thus, we have n data points in $\mathbb{R}^p$.
- Data point $x^i \in \mathbb{R}^p$ corresponds to the ith column of X
- Goal: find k mean vectors $\mu^j \in \mathbb{R}^p$, $j = 1, \ldots, k$, that satisfy

$$\mu^* = \arg\min_{\mu} \; h(X, \mu), \quad \mu = (\mu^j)_{j=1,\ldots,k}$$

- h is a sum of distances such as the squared Euclidean distance

$$h(X, \mu) = \sum_{j=1}^{k} \sum_{x^i \in S_j} \|x^i - \mu^j\|_2^2,$$

- $S_j \subset \mathbb{R}^p$ is the set of data points that are assigned to cluster $j = 1, \ldots, k$
- μ^j is the corresponding mean vector of the cluster

- Introduce binary variables $b_{i,j} \in \{0,1\}$ for $i = 1, \dots, n$ and $j = 1, \dots, k$
- Binary variables have the meaning

$$b_{i,j} = \begin{cases} 1, & \text{if point } x^i \text{ is assigned to cluster } S_j \\ 0, & \text{otherwise} \end{cases}$$

- Reformulate the function h as

$$h(X, b, \mu) = \sum_{j=1}^{k} \sum_{i=1}^{n} b_{i,j} \|x^i - \mu^j\|_2^2, \quad b = (b_{i,j})_{i=1,\dots,n}^{j=1,\dots,k}$$

- $x^i \in \mathbb{R}^p$ should belong to exactly one cluster
- Can be modeled using the SOS-1-type constraints

$$\sum_{j=1}^{k} b_{i,j} = 1 \quad \text{for all} \quad i = 1, \dots, n$$

$$\begin{aligned}
\min_{\mu, b} \quad & \sum_{j=1}^{k} \sum_{i=1}^{n} b_{i,j} \|x^i - \mu^j\|_2^2 \\
\text{s.t.} \quad & \sum_{j=1}^{k} b_{i,j} = 1 \quad \text{for all} \quad i = 1, \dots, n \\
& b_{i,j} \in \{0, 1\} \quad \text{for all} \quad i = 1, \dots, n, \ j = 1, \dots, k \\
& \mu \in \mathbb{R}^{p \times k}
\end{aligned}$$

k-means Clustering: MINLP Modeling

$$
\begin{aligned}
\min_{\mu, b} \quad & \sum_{j=1}^{k} \sum_{i=1}^{n} b_{i,j} \|x^i - \mu^j\|_2^2 \\
\text{s.t.} \quad & \sum_{j=1}^{k} b_{i,j} = 1 \quad \text{for all} \quad i = 1, \ldots, n \\
& b_{i,j} \in \{0, 1\} \quad \text{for all} \quad i = 1, \ldots, n, \ j = 1, \ldots, k \\
& \mu \in \mathbb{R}^{p \times k}
\end{aligned}
$$

Convex or Nonconvex MINLP?

$$\begin{aligned}
\min_{\mu, b} \quad & \sum_{j=1}^{k} \sum_{i=1}^{n} b_{i,j} \|x^i - \mu^j\|_2^2 \\
\text{s.t.} \quad & \sum_{j=1}^{k} b_{i,j} = 1 \quad \text{for all} \quad i = 1, \dots, n \\
& b_{i,j} \in \{0, 1\} \quad \text{for all} \quad i = 1, \dots, n, \; j = 1, \dots, k \\
& \mu \in \mathbb{R}^{p \times k}
\end{aligned}$$

Convex or Nonconvex MINLP?

- We have products of binary variables $b_{i,j}$ and continuous variables μ^j in the objective function.
- Thus, it's a nonconvex MINLP.

Bilinearities

Consider the bilinear function

$$f : \mathbb{R}^2 \to \mathbb{R}, \quad f(x, y) = xy.$$

Bilinearities

Consider the bilinear function

$$f : \mathbb{R}^2 \to \mathbb{R}, \quad f(x, y) = xy.$$

The gradient is given by

$$\nabla f(x) = \begin{pmatrix} y \\ x \end{pmatrix}$$

Bilinearities

Consider the bilinear function

$$f : \mathbb{R}^2 \to \mathbb{R}, \quad f(x, y) = xy.$$

The gradient is given by

$$\nabla f(x) = \begin{pmatrix} y \\ x \end{pmatrix}$$

and the Hessian reads

$$\nabla^2 f(x) = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix},$$

which is an indefinite matrix.

Bilinearities

Consider the bilinear function

$$f : \mathbb{R}^2 \to \mathbb{R}, \quad f(x, y) = xy.$$

The gradient is given by

$$\nabla f(x) = \begin{pmatrix} y \\ x \end{pmatrix}$$

and the Hessian reads

$$\nabla^2 f(x) = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix},$$

which is an indefinite matrix.

Thus, it's nonconvex!

2. Algorithms for Convex MINLP: Overview

ECONOMETRICA

VOLUME 28 July, 1960 NUMBER 3

AN AUTOMATIC METHOD OF SOLVING DISCRETE PROGRAMMING PROBLEMS

BY A. H. LAND AND A. G. DOIG

In the classical linear programming problem the behaviour of continuous, nonnegative variables subject to a system of linear inequalities is investigated. One possible generalization of this problem is to relax the continuity condition on the variables. This paper presents a simple numerical algorithm for the solution of programming problems in which some or all of the variables can take only discrete values. The algorithm requires no special techniques beyond those used in ordinary linear programming, and lends itself to automatic computing. Its use is illustrated on two numerical examples.

Branch-and-bound was proposed by Ailsa Land and Alison Doig in 1960

Branch-and-bound for convex MINLP is almost the same as for mixed-integer linear programming.

For the ease of presentation: all integer variables are binary

For the ease of presentation: all integer variables are binary

The original (M)ILP ...

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & c^\top x \\ \text{s.t.} \quad & Ax \geq b \\ & x \in \{0,1\}^n \end{aligned} \tag{1}$$

... and after some fixations

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & c^\top x \\ \text{s.t.} \quad & Ax \geq b \\ & x \in \{0,1\}^n \\ & x_i = 0 \quad \text{for all } i \in Z \\ & x_i = 1 \quad \text{for all } i \in O \end{aligned} \tag{2}$$

with $Z, O \subseteq \{1, \ldots, n\}$

Relaxations

Definition

Consider the optimization problem $\min_x\{f(x)\colon x \in \Omega\}$ with objective function f and feasible set Ω. Another optimization problem $\min_x\{g(x)\colon x \in \Omega'\}$ is called *relaxation* of the original problem if $\Omega \subseteq \Omega'$ and if $g(x) \leq f(x)$ holds for all $x \in \Omega$.

Definition

Consider the optimization problem $\min_x\{f(x) \colon x \in \Omega\}$ with objective function f and feasible set Ω. Another optimization problem $\min_x\{g(x) \colon x \in \Omega'\}$ is called *relaxation* of the original problem if $\Omega \subseteq \Omega'$ and if $g(x) \leq f(x)$ holds for all $x \in \Omega$.

Continuous Relaxations

- Convex NLP relaxation for convex MINLP
- LP relaxation for (M)ILP

Relaxations

Definition

Consider the optimization problem $\min_x \{f(x) \colon x \in \Omega\}$ with objective function f and feasible set Ω. Another optimization problem $\min_x \{g(x) \colon x \in \Omega'\}$ is called *relaxation* of the original problem if $\Omega \subseteq \Omega'$ and if $g(x) \leq f(x)$ holds for all $x \in \Omega$.

Continuous Relaxations

- Convex NLP relaxation for convex MINLP
- LP relaxation for (M)ILP

Goals

- Relaxations are used to compute lower bounds on the optimal objective function value
- A "good" relaxation should be tractable and "tight".

Simply ignore the integrality conditions ...

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & c^\top x \\ \text{s.t.} \quad & Ax \geq b \\ & x \in [0,1]^n \\ & x_i = 0 \quad \text{for all } i \in Z \\ & x_i = 1 \quad \text{for all } i \in O \end{aligned} \tag{3}$$

Lemma

Let $Z, O \subseteq \{1, \dots, n\}$. Moreover, let z_{LP} be the objective value of the solution of the LP relaxation (3) *and let z_{IP} be the optimal objective function value of Problem* (2) *(if they exist, otherwise we set them to ∞). Then,*

$$z_{IP} \geq z_{LP}$$

holds. Furthermore, infeasibility of the LP relaxation (3) *implies the infeasibility of* (2)*.*

Lemma

Let $Z, O \subseteq \{1, \ldots, n\}$. *Moreover, let* z_{LP} *be the objective value of the solution of the LP relaxation* (3) *and let* z_{IP} *be the optimal objective function value of Problem* (2) *(if they exist, otherwise we set them to* ∞*). Then,*

$$z_{IP} \geq z_{LP}$$

holds. Furthermore, infeasibility of the LP relaxation (3) *implies the infeasibility of* (2).

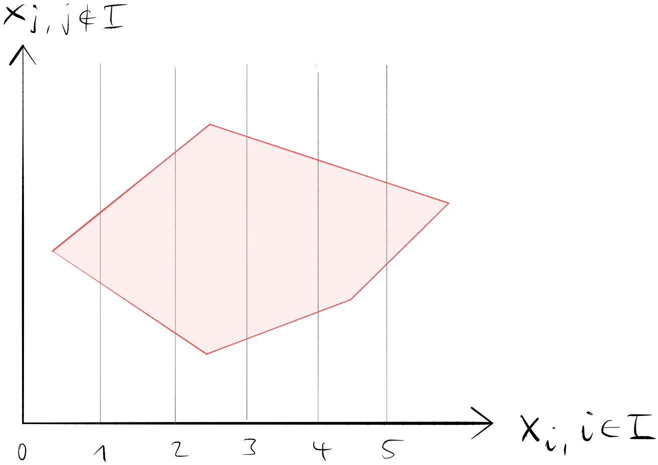
- Solving the LP relaxation gives us a lower bound on the optimal value
- If the LP relaxation is infeasible, then the original problem is infeasible

Lemma

Let $Z, O \subseteq \{1, \dots, n\}$*. Moreover, let* $x \in \{0,1\}^n$ *be feasible for* (2) *with the sets* Z, O *and* $i \in \{1, \dots, n\}$*. Then,* x *is either feasible for* (2) *with the sets* $(Z \cup \{i\}, O)$ *or feasible for* (2) *with the sets* $(Z, O \cup \{i\})$*.*

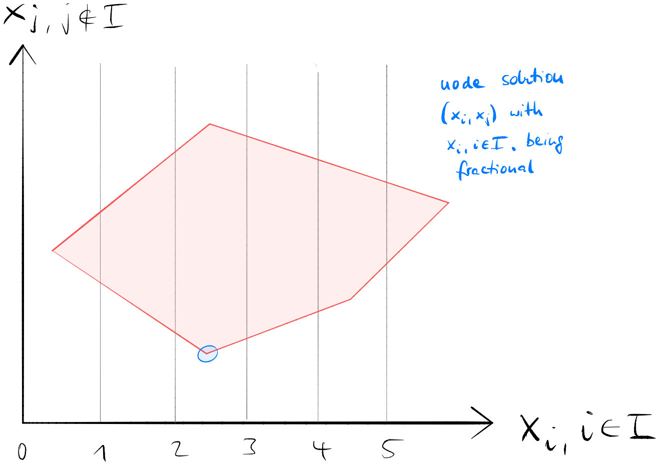
Branch-and-Bound for (Binary) MILPs

$u \leftarrow +\infty$ and $Q \leftarrow \{(\emptyset, \emptyset)\}$.
while $Q \neq \emptyset$ **do**
  Choose $(Z, O) \in Q$ and set $Q \leftarrow Q \setminus \{(Z, O)\}$.
  Solve the Problem (3) with Z and O.
  if (3) with Z and O is infeasible **then**
    Continue.
  end if
  Let $\bar{x}$ be the optimal solution of Problem (3).
  if $c^\top \bar{x} \geq u$ **then**
    Continue.
  end if
  if $\bar{x}$ is integer-feasible **then**
    Set $x^* \leftarrow \bar{x}$, $u \leftarrow c^\top x^*$, and continue.
  end if
  Choose i with $\bar{x}_i \notin \{0, 1\}$.
  Set $Q \leftarrow Q \cup \{(Z \cup \{i\}, O), (Z, O \cup \{i\})\}$.
end while
if $u < +\infty$ **then**
  return optimal solution x^*.
else
  return "The problem is infeasible."
end if

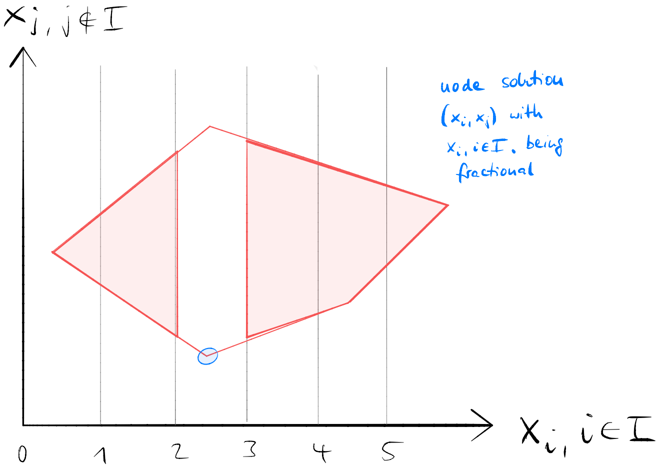


$x_j, j \notin I$
$x_i, i \in I$
0
1
2
3
4
5

Branch-and-Bound for General MILPs



Branch-and-Bound for General MILPs



Branch-and-Bound for (Binary) MILPs

$$\min \quad f(x) = c^\top x \quad \text{s.t.} \quad Ax = b, \quad x \geq 0, \quad x = (y, z), \quad y \in \mathbb{R}^m, \quad z \in \{0,1\}^k$$

Branch-and-Bound for (Binary) MILPs

$$\min \quad f(x) = c^\top x \quad \text{s.t.} \quad Ax = b, \quad x \geq 0, \quad x = (y, z), \quad y \in \mathbb{R}^m, \quad z \in \{0,1\}^k$$

best integer-feasible solution: $u = \infty$

$$\min \quad f(x) = c^\top x \quad \text{s.t.} \quad Ax = b, \quad x \geq 0, \quad x = (y, z), \quad y \in \mathbb{R}^m, \quad z \in \{0,1\}^k$$

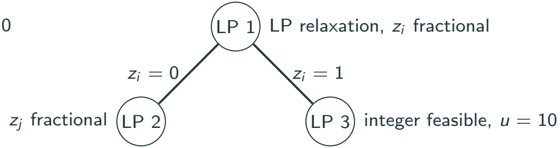
best integer-feasible solution: $u = \infty$

(LP 1) LP relaxation, z_i fractional

Branch-and-Bound for (Binary) MILPs

$$\min \quad f(x) = c^\top x \quad \text{s.t.} \quad Ax = b, \quad x \geq 0, \quad x = (y, z), \quad y \in \mathbb{R}^m, \quad z \in \{0,1\}^k$$

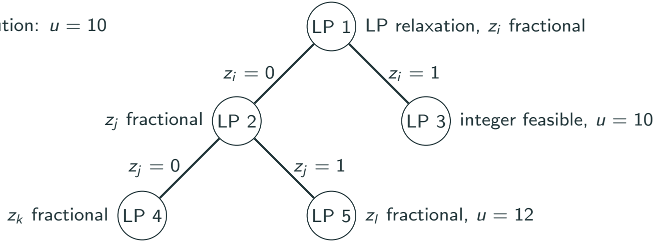
best integer-feasible solution: $u = 10$



Branch-and-Bound for (Binary) MILPs

$$\min \quad f(x) = c^\top x \quad \text{s.t.} \quad Ax = b, \quad x \geq 0, \quad x = (y, z), \quad y \in \mathbb{R}^m, \quad z \in \{0,1\}^k$$

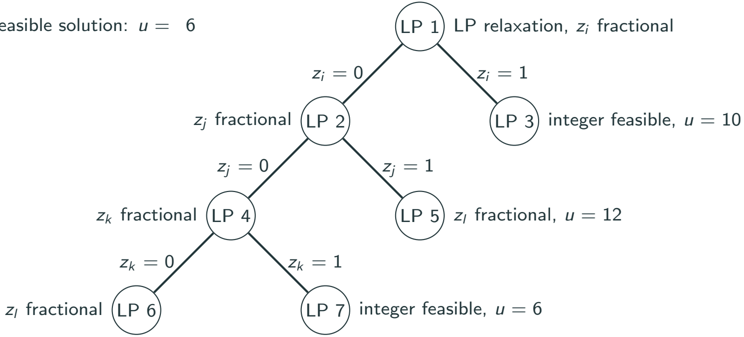
best integer-feasible solution: $u = 10$



Branch-and-Bound for (Binary) MILPs

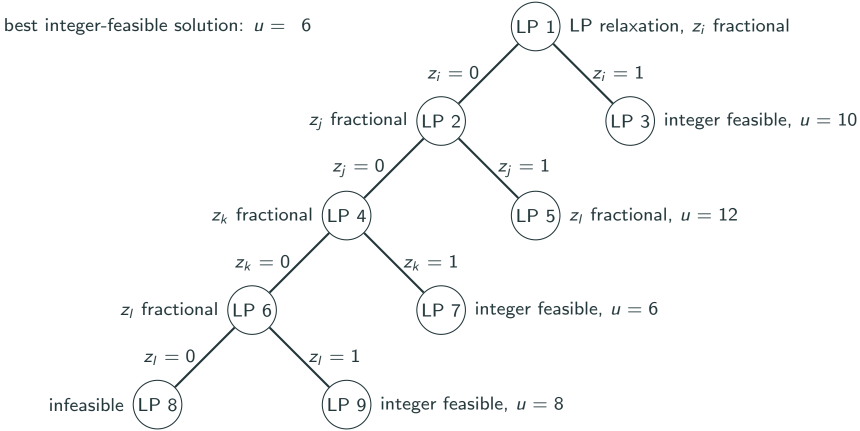
$$\min \quad f(x) = c^\top x \quad \text{s.t.} \quad Ax = b, \quad x \geq 0, \quad x = (y, z), \quad y \in \mathbb{R}^m, \quad z \in \{0,1\}^k$$

best integer-feasible solution: $u = 6$



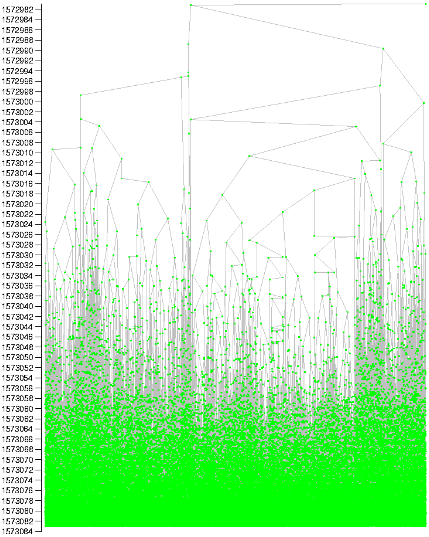
Branch-and-Bound for (Binary) MILPs

$$\min \quad f(x) = c^\top x \quad \text{s.t.} \quad Ax = b, \quad x \geq 0, \quad x = (y, z), \quad y \in \mathbb{R}^m, \quad z \in \{0,1\}^k$$



Branch-and-Bound for MILPs

Search trees get huge!



http://www.math.uwaterloo.ca/tsp/d15sol/computation

Branch-and-Bound for MILPs

- Every node of the branch-and-bound tree represents a sub-MILP
- In every node an LP is solved
  - LP relaxation + set of additional variables bounds (or fixations)
- If a node has an integer feasible point it becomes a leaf of the search tree
  - Integer feasible points yield upper bounds
  - Best (smallest) upper bound is called "incumbent" u
- Infeasible nodes also become leafs of the search tree
- Nodes with fractional solution and objective function value $f > u$ also become leafs
- Relaxation solutions yield lower bounds
- Best lower bound ("best bound") ℓ
- Optimality gap: $g = u - \ell$
  - $g = 0$ is a proof of optimality

Theorem (Correctness Theorem)

Suppose that the LP relaxation of the original MILP is bounded. Then, the algorithm terminates after a finite number of nodes with a global optimal solution or with the correct indication of infeasibility.

From MILP to Convex MINLP

- Replace LP relaxation with convex NLP relaxations in the nodes
- Technical details
  - All functions need to be continuously differentiable
  - All convex node NLPs need to satisfy Slater's condition
- All node NLPs need to be solved to global optimality
  - ... which is "easy" since they are convex!

What about nonconvex MINLPs?

Branch-and-bound is not correct for nonconvex MINLPs

But why?

What about nonconvex MINLPs?

Branch-and-bound is not correct for nonconvex MINLPs

But why?

- Locally optimal solutions might lead to pruned nodes that cannot be pruned!
- We might not find the global optimal solution.

Further Algorithmic Ingredients

- Node selection
- Branching rules
- Preprocessing of the entire MILP (root node)
- Node preprocessing (sub-MILPs)
- Cutting planes
- Heuristics
- Parallelization
- ...

Further Algorithmic Ingredients

- Node selection
- Branching rules
- Preprocessing of the entire MILP (root node)
- Node preprocessing (sub-MILPs)
- Cutting planes
- Heuristics
- Parallelization
- ...

Performance

- Worst-case complexity: exponential
- In practice it often performs much better!

Branch-and-Bound Solvers for MILPs

- CPLEX
  - http://www-01.ibm.com/software/commerce/optimization/cplex-optimizer/
  - Commercial (IBM), but free licenses available for academic purposes
- Gurobi
  - http://www.gurobi.com/
  - Commercial, but free licenses available for academic purposes
- Xpress
  - http://www.fico.com/en/products/fico-xpress-optimization-suite/
  - Commercial (FICO), but free licenses available for academic purposes
- SCIP
  - http://scip.zib.de
  - Academic code, free for non-commercial purposes, open source
- CBC
  - https://projects.coin-or.org/Cbc
  - Open source

Branch-and-Bound vs. Cutting-Plane Methods

The Main Idea of Branch-and-Bound

- By branching we get rid of the integer variables
- Subproblems that need to be solved are continuous relaxations
- Bounding and finiteness of the set of integer-feasible points leads to correctness of the method

Branch-and-Bound vs. Cutting-Plane Methods

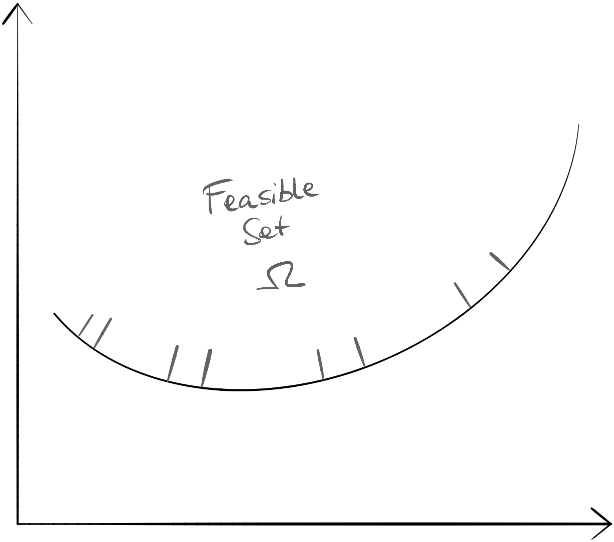
The Main Idea of Branch-and-Bound

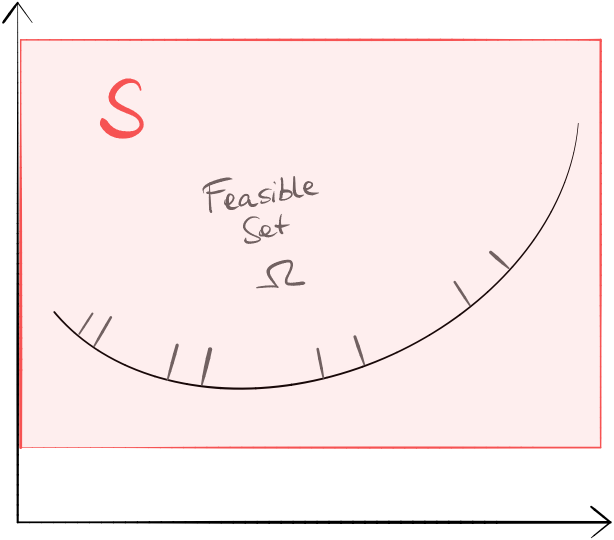
- By branching we get rid of the integer variables
- Subproblems that need to be solved are continuous relaxations
- Bounding and finiteness of the set of integer-feasible points leads to correctness of the method

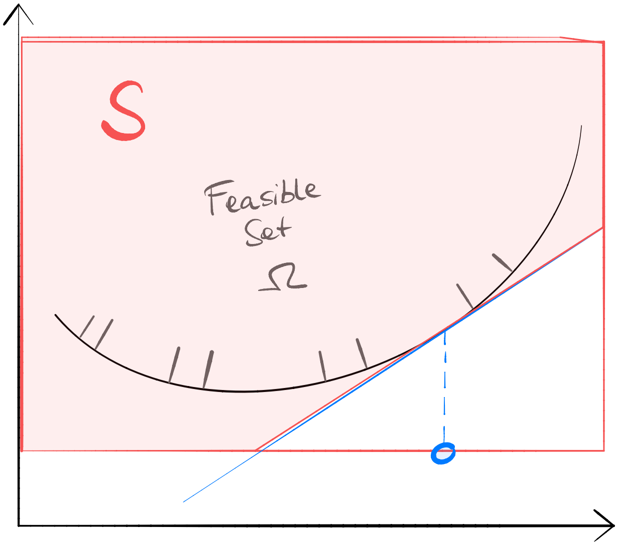
The Main Idea of Cutting-Plane Methods

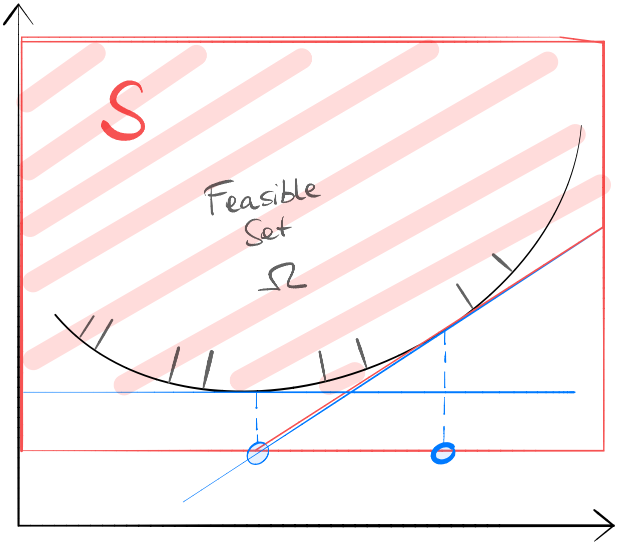
- Besides integrality constraints, the hardness of MINLPs comes from nonlinearities
- Assumption: We can solve mixed-integer linear problems
  - Tractability: NP-hardness vs. polytime solvable problems
  - Practability: Powerful solvers that solve NP-hard problems
- Idea: Get rid of nonlinearities by linear approximations
- Correctness will follow by convergence instead of finiteness arguments

Kelley in a Nutshell




S
Feasible
Set
Ω


S
Feasible
Set
Ω


S
Feasible
Set
Ω

The Mother of all Cutting-Plane Methods . . .

. . . at least for nonlinear problems!

J. Soc. Indust. Appl. Math.
Vol. 8, No. 4, December, 1960
Printed in U.S.A.

THE CUTTING-PLANE METHOD FOR SOLVING CONVEX PROGRAMS*

J. E. KELLEY, Jr.†

1. Introduction. Although generally quite difficult to solve, constrained minima problems are of perennial interest. There has been relatively little success in finding general computational techniques for handling them. However, useful techniques have been developed for certain small classes of these problems. One interesting class involves minimizing a continuous convex function on a closed convex set. It is known as the convex programming problem and has been the subject of numerous studies in recent years.[1] The main reason for success in this area appears to be that, with convex functions, all local minima are global minima.

Kelley's Cutting-Plane Method: Setting

Just for a moment: Forget about integrality constraints.

Convex Optimization

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & f(x) \\ \text{s.t.} \quad & x \in \Omega \end{aligned}$$

- Without loss of generality: linear objective function $f(x) = d^\top x$, $d \in \mathbb{R}^n$
- Nonempty convex feasible set

$$\Omega := \{x \in \mathbb{R}^n \colon c(x) \leq 0\},$$

i.e., $c : \mathbb{R}^n \to \mathbb{R}^m$ is convex

Kelley's Cutting-Plane Method: Setting

Assumptions

1. $\|d\| < \infty$
2. c is continuously differentiable
3. $\|\nabla c(x)\| \leq K$ for a finite constant K and all $x \in \Omega$
4. There exists a compact polyhedral set

$$S = \{x \in \mathbb{R}^n \colon Ax \geq b\} \supseteq \Omega, \quad A \in \mathbb{R}^{m \times n}, \quad b \in \mathbb{R}^m$$

Kelley's Cutting-Plane Method: Setting

Definition

An *extreme support* to the graph of c is an $(n+1)$-dimensional hyperplane that intersects the boundary of the convex set

$$P = \{(x, y)\colon x \in S,\ y \geq c(x)\}$$

and does not cut the interior of P.

Definition

An *extreme support* to the graph of c is an $(n+1)$-dimensional hyperplane that intersects the boundary of the convex set

$$P = \{(x, y) \colon x \in S,\ y \geq c(x)\}$$

and does not cut the interior of P.

For every $x \in S$ there exists an extreme support to the graph of c since it is convex.

Kelley's Cutting-Plane Method: Setting

Definition

An *extreme support* to the graph of c is an $(n+1)$-dimensional hyperplane that intersects the boundary of the convex set

$$P = \{(x, y) \colon x \in S,\ y \geq c(x)\}$$

and does not cut the interior of P.

For every $x \in S$ there exists an extreme support to the graph of c since it is convex.

Simply use Taylor's first-order approximation of c.

Kelley's Cutting-Plane Method: Setting

Definition

An *extreme support* to the graph of c is an $(n+1)$-dimensional hyperplane that intersects the boundary of the convex set

$$P = \{(x, y)\colon x \in S,\ y \geq c(x)\}$$

and does not cut the interior of P.

For every $x \in S$ there exists an extreme support to the graph of c since it is convex.

Simply use Taylor's first-order approximation of c.

For a point $x^0 \in S$, the extreme support $y = p(x; x^0)$ is given by

$$p(x; x^0) = c(x^0) + \nabla p(x; x^0)^\top (x - x^0), \quad \nabla p(x; x^0) = \nabla c(x^0).$$

Kelley's Cutting-Plane Method: Let's Start!

1. Solve the relaxation $\min_x \{f(x) \colon x \in S\}$.
    - If the problem is infeasible, the original problem is infeasible.
    - Otherwise, let x^0 be the optimal solution.
2. If $x^0 \in \Omega$, we are done. So let $x^0 \in S \setminus \Omega$.
3. Since c is convex, we have

$$p(x; x^0) \leq c(x) \quad \text{for all } x \in S.$$

Definition

Consider the optimization problem

$$\min_{x} \quad f(x) \quad \text{s.t.} \quad x \in \Omega.$$

An inequality $a^\top x \leq b$ is called a *valid inequality* (for Ω) if it is satisfied for all feasible points $x \in \Omega$.

It really cuts

Lemma

Let $x^0 \in S \setminus \Omega$. *The hyperplane defined by* $p(x; x^0) = 0$ *separates the point* x^0 *from the feasible set* Ω.

It really cuts

Lemma

Let $x^0 \in S \setminus \Omega$. *The hyperplane defined by* $p(x; x^0) = 0$ *separates the point* x^0 *from the feasible set* Ω.

Proof.

- It holds $p(x; x^0) \leq c(x)$ for all $x \in S$.
- Thus, if $x \in \Omega$, we have $p(x; x^0) \leq c(x) \leq 0$.
- Since $x^0 \notin \Omega$, $p(x^0; x^0) = c(x^0) > 0$. □

It really cuts

Lemma

Let $x^0 \in S \setminus \Omega$. *The hyperplane defined by* $p(x; x^0) = 0$ *separates the point* x^0 *from the feasible set* Ω.

Proof.

- It holds $p(x; x^0) \leq c(x)$ for all $x \in S$.
- Thus, if $x \in \Omega$, we have $p(x; x^0) \leq c(x) \leq 0$.
- Since $x^0 \notin \Omega$, $p(x^0; x^0) = c(x^0) > 0$. □

In this situation, the valid inequality of this hyperplane is called a *cut*.

Kelley's Cutting-Plane Method: The Iteration

- Consider a sequence $(S_k)_k$ of convex sets with $S_k \subset S_{k-1}$ and consider the sequence of convex optimization problems

$$\min_x \quad f(x) \quad \text{s.t.} \quad x \in S_k$$

  with optimal solutions x^k. Then $f(x^k) \geq f(x^{k-1})$ holds.
- Let $S_0 = S$ and

$$S_1 = S_0 \cap \{x \in \mathbb{R}^n \colon p(x; x^0) \leq 0\}.$$

- More general, we have the *tightenings*

$$S_k = S_{k-1} \cap \{x \in \mathbb{R}^n \colon p(x; x^{k-1}) \leq 0\}.$$

We obtain ...

- points x^k that minimize $f(x)$ over S_k
- sequences $(x^k)_k$ and $(f_k)_k$ with $f_k = d^\top x^k$

Kelley's Cutting-Plane Method: Goal

If $(x^k)_k$ has a convergent subsequence that converges to a point $x^* \in \Omega$, then the monotonically increasing sequence $(f_k)_k$ converges to $f(x^*)$ and x^* is the desired optimal solution.

Kelley's Cutting-Plane Method: Convergence

- x^k minimizes $f(x)$ over S_k, i.e.,

$$c(x^k) + \nabla p(x^k; x^i)^\top (x^k - x^i) \leq 0$$

  for all $i = 0, \dots, k-1$.
- Moreover, if $(x^k)_k$ has a subsequence converging to a point in Ω, then $(c(x^k))_k$ needs to have a subsequence converging to 0.
- If not, there exists an $r > 0$ (independent of k) such that

$$r \leq c(x^i) \leq \nabla p(x^k; x^i)^\top (x^i - x^k) \leq K \|x^i - x^k\|$$

  for all $i = 0, \dots, k-1$.
- Thus, for every subsequence $(x^{k_\ell})_\ell$ we obtain

$$\|x^{k_q} - x^{k_\ell}\| \geq \frac{r}{K} > 0$$

  for all $q < \ell$.
- Thus, $(x^k)_k$ is not a Cauchy sequence, which is a contradiction since S is compact.

Kelley's Cutting-Plane Method: Main Theorem

Theorem

Let c be a continuous and convex function defined on S so that for every point $t \in S$, there exists an extreme support, $y = p(x; t)$ to the graph of c with $\|\nabla p(x; t)\| \leq K$ for some finite constant K and for all $x \in S$.

Theorem

Let c be a continuous and convex function defined on S so that for every point $t \in S$, there exists an extreme support, $y = p(x;t)$ to the graph of c with $\|\nabla p(x;t)\| \leq K$ for some finite constant K and for all $x \in S$. Furthermore, let $\|d\| \leq \infty$ and let $\emptyset \neq \Omega = \{x \in \mathbb{R}^n : c(x) \leq 0\} \subset S$.

Theorem

Let c be a continuous and convex function defined on S so that for every point $t \in S$, there exists an extreme support, $y = p(x;t)$ to the graph of c with $\|\nabla p(x;t)\| \leq K$ for some finite constant K and for all $x \in S$. Furthermore, let $\|d\| \leq \infty$ and let $\emptyset \neq \Omega = \{x \in \mathbb{R}^n : c(x) \leq 0\} \subset S$. If $x^k \in S^k$ solves

$$\min_x \quad f(x) \quad s.t. \quad x \in S_k \quad \text{for } k = 0, 1, \ldots$$

with $S_0 = S$ and

$$S_k = S_{k-1} \cap \left\{ x \in \mathbb{R}^n : p(x; x^{k-1}) \leq 0 \right\},$$

Theorem

Let c be a continuous and convex function defined on S so that for every point $t \in S$, there exists an extreme support, $y = p(x;t)$ to the graph of c with $\|\nabla p(x;t)\| \le K$ for some finite constant K and for all $x \in S$. Furthermore, let $\|d\| \le \infty$ and let $\emptyset \neq \Omega = \{x \in \mathbb{R}^n \colon c(x) \le 0\} \subset S$. If $x^k \in S^k$ solves

$$\min_x \quad f(x) \quad s.t. \quad x \in S_k \quad \text{for } k = 0, 1, \ldots$$

with $S_0 = S$ and

$$S_k = S_{k-1} \cap \left\{ x \in \mathbb{R}^n \colon p(x; x^{k-1}) \le 0 \right\},$$

then the sequence $(x^k)_k$ contains subsequences that converges to a point $x^* \in \Omega$ with $f(x^*) \le f(x)$ for all $x \in \Omega$.

- In Kelley's method we solve LPs in every iteration, which are polyhedral outer approximations of the original feasible set.
- If we have a convex MINLP, simply solve MILPs instead of LPs.
- Every point x^k then is the solution of the MILP with feasible set S_k that also incorporates the integrality constraints.
- That's it?

- In Kelley's method we solve LPs in every iteration, which are polyhedral outer approximations of the original feasible set.
- If we have a convex MINLP, simply solve MILPs instead of LPs.
- Every point x^k then is the solution of the MILP with feasible set S_k that also incorporates the integrality constraints.
- That's it?

Yes; except for ugly technicalities

- We need that the vector c and all constraints defining $S_0, S_1, S_2, \ldots$ only have rational coefficients and constants.
- There are workarounds; see page 707 of the original Kelley paper.

Single-Tree vs. Multi-Tree Methods

Single-Tree

- Branch-and-bound
- Only a single search-tree is enumerated
- Every node of the search tree corresponds to a continuous optimization problem

Single-Tree vs. Multi-Tree Methods

Single-Tree

- Branch-and-bound
- Only a single search-tree is enumerated
- Every node of the search tree corresponds to a continuous optimization problem

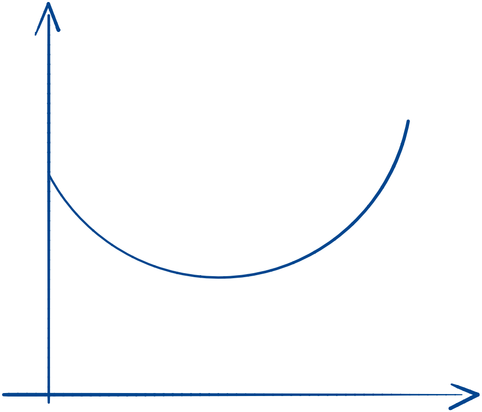
Multi-Tree

- Kelley's Method for convex MINLPs
- An MILP is solved in every iteration
- Thus, every iteration corresponds to one search-tree

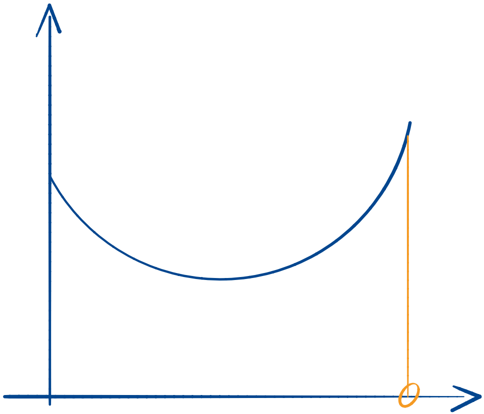
$$\min_{x \in \mathbb{R}^2} \quad x_1 - x_2 \quad \text{s.t.} \quad 3x_1^2 - 2x_1x_2 + x_2^2 - 1 \leq 0$$

The original numerical results from the Kelley paper

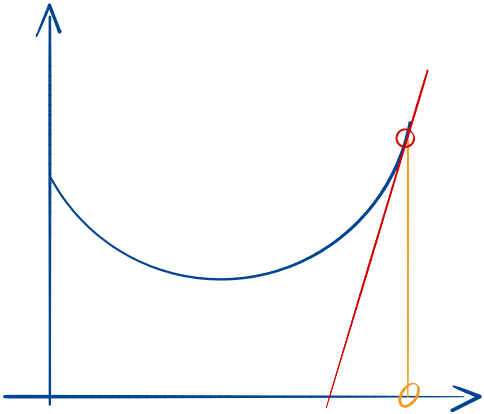
k	$p(\mathbf{x}; \mathbf{t}_k)$	$t_1^{(k)}$	$t_2^{(k)}$	f_k	$G(\mathbf{t}_k)$
0	$-16.00000x_1+8.00000x_2-25.00000$	-2.00000	2.00000	-4.00000	23.00000
1	$-7.37500x_1+5.12500x_2-8.19922$	-0.56250	2.00000	-2.56250	6.19922
2	$-2.33157x_1+3.44386x_2-4.11958$	0.27870	2.00000	-1.72193	2.11978
3	$-4.85341x_1+2.73459x_2-3.43067$	-0.52970	0.83759	-1.36730	1.43067
4	$-2.63930x_1+2.42675x_2-2.47792$	-0.05314	1.16024	-1.21338	0.47793
5	$-0.41071x_1+2.11690x_2-2.48420$	0.42655	1.48499	-1.05845	0.48419
6	$-1.38975x_1+2.07205x_2-2.13155$	0.17058	1.20660	-1.03603	0.13154
7	$-1.97223x_1+2.04538x_2-2.04657$	0.01829	1.04098	-1.02269	0.04656
8	$-2.67809x_1+2.01305x_2-2.06838$	-0.16626	0.84027	-1.00653	0.06838
9		-0.07348	0.92972	-1.00321	0.01723



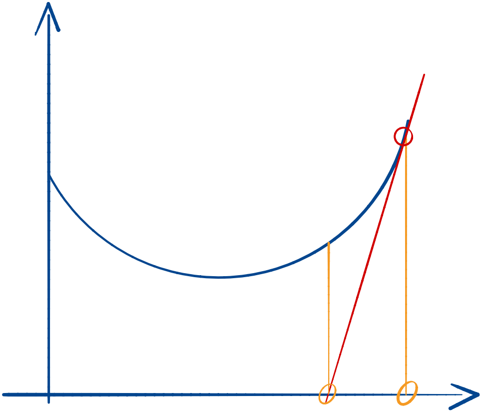
Kelley: A Good Idea That Is Not So Good?



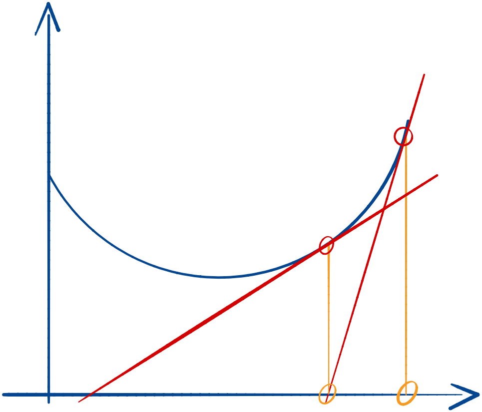
Kelley: A Good Idea That Is Not So Good?



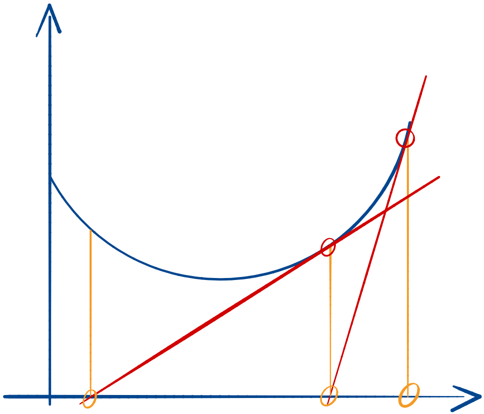
Kelley: A Good Idea That Is Not So Good?



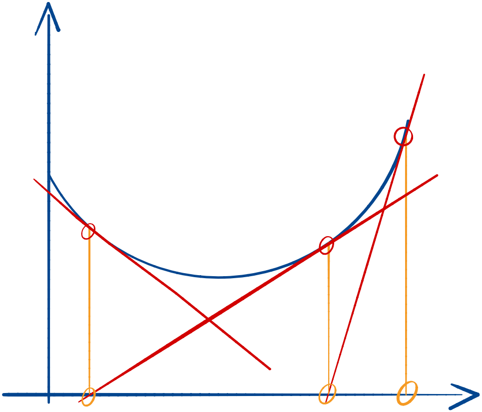
Kelley: A Good Idea That Is Not So Good?



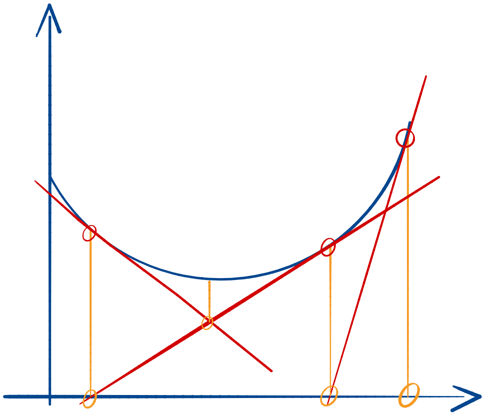
Kelley: A Good Idea That Is Not So Good?



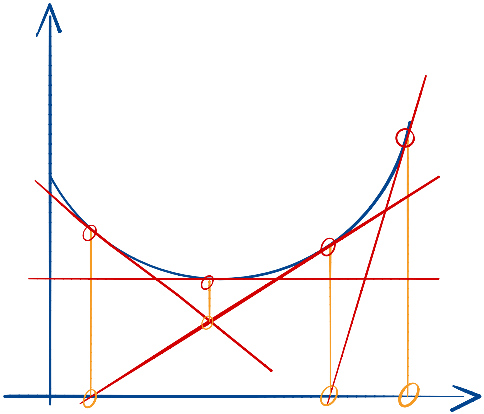
Kelley: A Good Idea That Is Not So Good?

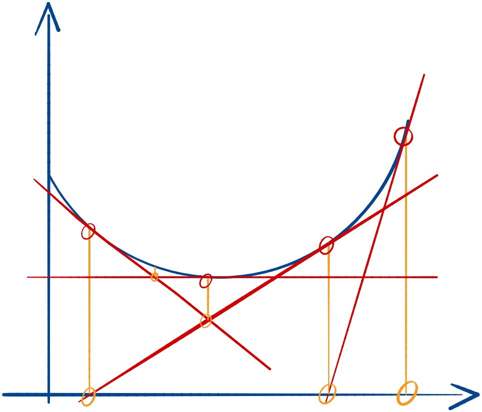


Kelley: A Good Idea That Is Not So Good?

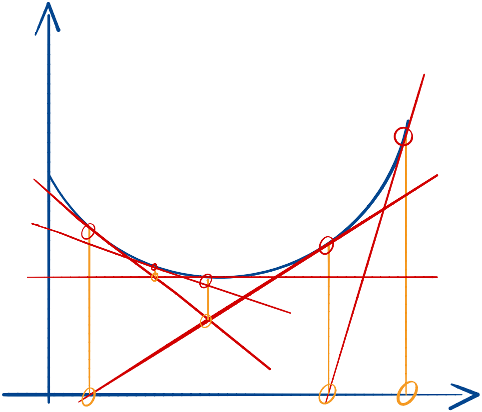


Kelley: A Good Idea That Is Not So Good?

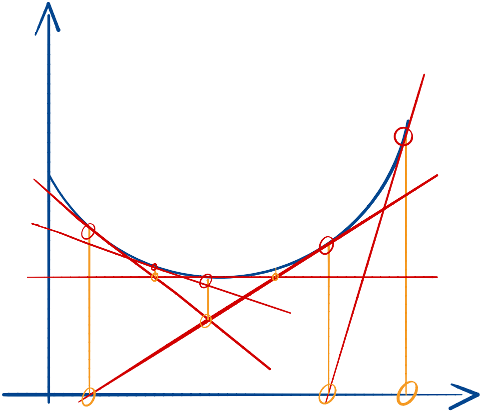




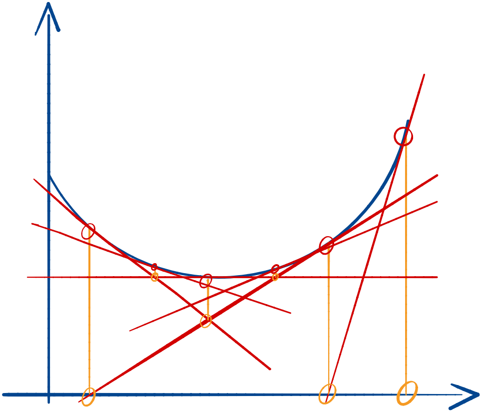
Kelley: A Good Idea That Is Not So Good?



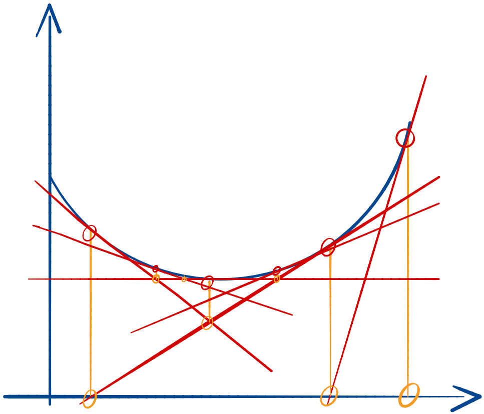
Kelley: A Good Idea That Is Not So Good?



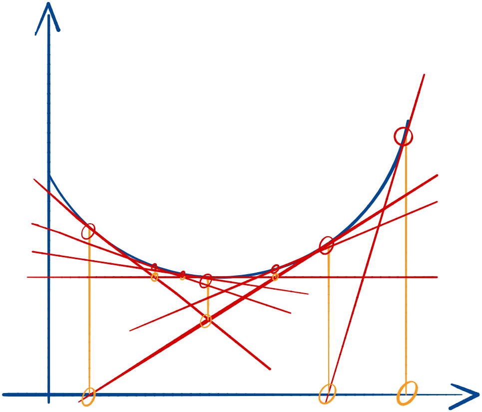
Kelley: A Good Idea That Is Not So Good?



Kelley: A Good Idea That Is Not So Good?



Kelley: A Good Idea That Is Not So Good?



Who Would Even Implement This?

Who Would Even Implement This?

Well, we did . . .

Well, we did . . .

23	28.721478	29.782867	1.03e-01	5.14e+00	0.0	28.721478	29.782862	3.5637%	11.10s	255.34s
24	28.919258	29.782867	9.23e-02	4.56e+00	0.0	28.919258	29.782862	2.8997%	11.15s	267.53s
25	29.039661	29.782867	7.99e-02	5.34e+00	0.0	29.039661	29.782862	2.4954%	11.17s	279.31s
26	29.134855	29.782867	6.32e-02	5.02e+00	0.0	29.134855	29.782862	2.1758%	11.20s	291.15s
27	29.231516	29.782867	4.44e-02	3.40e+00	0.0	29.231516	29.782862	1.8512%	11.24s	303.54s
28	29.346041	29.782867	4.36e-02	3.08e+00	0.0	29.346041	29.782862	1.4667%	11.29s	316.08s
29	29.382217	29.782867	2.63e-02	3.40e+00	0.0	29.382217	29.782862	1.3452%	11.32s	328.39s
30	29.508949	29.782867	2.19e-02	2.14e+00	0.0	29.508949	29.782862	9.2e-01%	11.36s	340.82s
31	29.541662	29.782867	1.61e-02	2.00e+00	0.0	29.541662	29.782862	8.1e-01%	11.39s	353.03s
32	29.582359	29.782867	1.46e-02	2.40e+00	0.0	29.582359	29.782862	6.7e-01%	11.43s	365.74s
33	29.613035	29.782867	7.94e-03	1.77e+00	0.0	29.613035	29.782862	5.7e-01%	11.46s	378.12s
34	29.693581	29.782867	6.27e-03	1.33e+00	0.0	29.693581	29.782862	3.0e-01%	11.49s	390.53s
35	29.706371	29.782867	3.29e-03	1.31e+00	0.0	29.706371	29.782862	2.6e-01%	11.51s	402.89s
36	29.758188	29.782867	6.22e-04	6.48e-01	0.0	29.758188	29.782862	8.3e-02%	11.54s	415.26s
37	29.769352	29.782867	1.88e-04	3.22e-01	0.0	29.769352	29.782862	4.5e-02%	11.57s	428.17s
38	29.775732	29.782867	1.92e-04	3.31e-01	0.0	29.775732	29.782862	2.4e-02%	11.60s	440.78s
39	29.776949	29.782867	5.80e-05	1.72e-01	0.0	29.775732	29.782862	2.4e-02%	11.40s	444.48s
40	29.781179	29.782867	8.63e-06	7.68e-02	0.0	29.781179	29.782862	5.6e-03%	11.41s	456.40s

Well, we did . . .

23	28.721478	29.782867	1.03e-01	5.14e+00	0.0	28.721478	29.782862	3.5637%	11.10s	255.34s
24	28.919258	29.782867	9.23e-02	4.56e+00	0.0	28.919258	29.782862	2.8997%	11.15s	267.53s
25	29.039661	29.782867	7.99e-02	5.34e+00	0.0	29.039661	29.782862	2.4954%	11.17s	279.31s
26	29.134855	29.782867	6.32e-02	5.02e+00	0.0	29.134855	29.782862	2.1758%	11.20s	291.15s
27	29.231516	29.782867	4.44e-02	3.40e+00	0.0	29.231516	29.782862	1.8512%	11.24s	303.54s
28	29.346041	29.782867	4.36e-02	3.08e+00	0.0	29.346041	29.782862	1.4667%	11.29s	316.08s
29	29.382217	29.782867	2.63e-02	3.40e+00	0.0	29.382217	29.782862	1.3452%	11.32s	328.39s
30	29.508949	29.782867	2.19e-02	2.14e+00	0.0	29.508949	29.782862	9.2e-01%	11.36s	340.82s
31	29.541662	29.782867	1.61e-02	2.00e+00	0.0	29.541662	29.782862	8.1e-01%	11.39s	353.03s
32	29.582359	29.782867	1.46e-02	2.40e+00	0.0	29.582359	29.782862	6.7e-01%	11.43s	365.74s
33	29.613035	29.782867	7.94e-03	1.77e+00	0.0	29.613035	29.782862	5.7e-01%	11.46s	378.12s
34	29.693581	29.782867	6.27e-03	1.33e+00	0.0	29.693581	29.782862	3.0e-01%	11.49s	390.53s
35	29.706371	29.782867	3.29e-03	1.31e+00	0.0	29.706371	29.782862	2.6e-01%	11.51s	402.89s
36	29.758188	29.782867	6.22e-04	6.48e-01	0.0	29.758188	29.782862	8.3e-02%	11.54s	415.26s
37	29.769352	29.782867	1.88e-04	3.22e-01	0.0	29.769352	29.782862	4.5e-02%	11.57s	428.17s
38	29.775732	29.782867	1.92e-04	3.31e-01	0.0	29.775732	29.782862	2.4e-02%	11.60s	440.78s
39	29.776949	29.782867	5.80e-05	1.72e-01	0.0	29.775732	29.782862	2.4e-02%	11.40s	444.48s
40	29.781179	29.782867	8.63e-06	7.68e-02	0.0	29.781179	29.782862	5.6e-03%	11.41s	456.40s

Who Would Even Implement This?

Well, we did . . .

23	28.721478	29.782867	1.03e-01	5.14e+00	0.0	28.721478	29.782862	3.5637%	11.10s	255.34s
24	28.919258	29.782867	9.23e-02	4.56e+00	0.0	28.919258	29.782862	2.8997%	11.15s	267.53s
25	29.039661	29.782867	7.99e-02	5.34e+00	0.0	29.039661	29.782862	2.4954%	11.17s	279.31s
26	29.134855	29.782867	6.32e-02	5.02e+00	0.0	29.134855	29.782862	2.1758%	11.20s	291.15s
27	29.231516	29.782867	4.44e-02	3.40e+00	0.0	29.231516	29.782862	1.8512%	11.24s	303.54s
28	29.346041	29.782867	4.36e-02	3.08e+00	0.0	29.346041	29.782862	1.4667%	11.29s	316.08s
29	29.382217	29.782867	2.63e-02	3.40e+00	0.0	29.382217	29.782862	1.3452%	11.32s	328.39s
30	29.508949	29.782867	2.19e-02	2.14e+00	0.0	29.508949	29.782862	9.2e-01%	11.36s	340.82s
31	29.541662	29.782867	1.61e-02	2.00e+00	0.0	29.541662	29.782862	8.1e-01%	11.39s	353.03s
32	29.582359	29.782867	1.46e-02	2.40e+00	0.0	29.582359	29.782862	6.7e-01%	11.43s	365.74s
33	29.613035	29.782867	7.94e-03	1.77e+00	0.0	29.613035	29.782862	5.7e-01%	11.46s	378.12s
34	29.693581	29.782867	6.27e-03	1.33e+00	0.0	29.693581	29.782862	3.0e-01%	11.49s	390.53s
35	29.706371	29.782867	3.29e-03	1.31e+00	0.0	29.706371	29.782862	2.6e-01%	11.51s	402.89s
36	29.758188	29.782867	6.22e-04	6.48e-01	0.0	29.758188	29.782862	8.3e-02%	11.54s	415.26s
37	29.769352	29.782867	1.88e-04	3.22e-01	0.0	29.769352	29.782862	4.5e-02%	11.57s	428.17s
38	29.775732	29.782867	1.92e-04	3.31e-01	0.0	29.775732	29.782862	2.4e-02%	11.60s	440.78s
39	29.776949	29.782867	5.80e-05	1.72e-01	0.0	29.775732	29.782862	2.4e-02%	11.40s	444.48s
40	29.781179	29.782867	8.63e-06	7.68e-02	0.0	29.781179	29.782862	5.6e-03%	11.41s	456.40s

Who Would Even Implement This?

Well, we did . . .

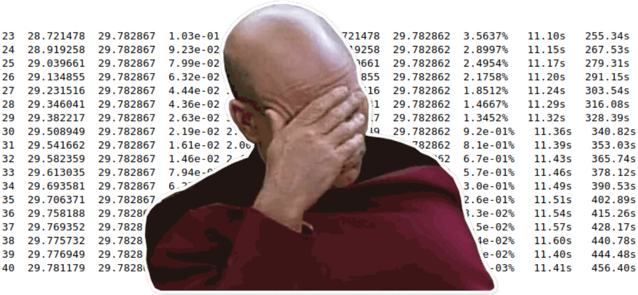
23	28.721478	29.782867	1.03e-01	5.14e+00	0.0	28.721478	29.782862	3.5637%	11.10s	255.34s
24	28.919258	29.782867	9.23e-02	4.56e+00	0.0	28.919258	29.782862	2.8997%	11.15s	267.53s
25	29.039661	29.782867	7.99e-02	5.34e+00	0.0	29.039661	29.782862	2.4954%	11.17s	279.31s
26	29.134855	29.782867	6.32e-02	5.02e+00	0.0	29.134855	29.782862	2.1758%	11.20s	291.15s
27	29.231516	29.782867	4.44e-02	3.40e+00	0.0	29.231516	29.782862	1.8512%	11.24s	303.54s
28	29.346041	29.782867	4.36e-02	3.08e+00	0.0	29.346041	29.782862	1.4667%	11.29s	316.08s
29	29.382217	29.782867	2.63e-02	3.40e+00	0.0	29.382217	29.782862	1.3452%	11.32s	328.39s
30	29.508949	29.782867	2.19e-02	2.14e+00	0.0	29.508949	29.782862	9.2e-01%	11.36s	340.82s
31	29.541662	29.782867	1.61e-02	2.00e+00	0.0	29.541662	29.782862	8.1e-01%	11.39s	353.03s
32	29.582359	29.782867	1.46e-02	2.40e+00	0.0	29.582359	29.782862	6.7e-01%	11.43s	365.74s
33	29.613035	29.782867	7.94e-03	1.77e+00	0.0	29.613035	29.782862	5.7e-01%	11.46s	378.12s
34	29.693581	29.782867	6.27e-03	1.33e+00	0.0	29.693581	29.782862	3.0e-01%	11.49s	390.53s
35	29.706371	29.782867	3.29e-03	1.31e+00	0.0	29.706371	29.782862	2.6e-01%	11.51s	402.89s
36	29.758188	29.782867	6.22e-04	6.48e-01	0.0	29.758188	29.782862	8.3e-02%	11.54s	415.26s
37	29.769352	29.782867	1.88e-04	3.22e-01	0.0	29.769352	29.782862	4.5e-02%	11.57s	428.17s
38	29.775732	29.782867	1.92e-04	3.31e-01	0.0	29.775732	29.782862	2.4e-02%	11.60s	440.78s
39	29.776949	29.782867	5.80e-05	1.72e-01	0.0	29.775732	29.782862	2.4e-02%	11.40s	444.48s
40	29.781179	29.782867	8.63e-06	7.68e-02	0.0	29.781179	29.782862	5.6e-03%	11.41s	456.40s

Who Would Even Implement This?

Well, we did . . .

23	28.721478	29.782867	1.03e-01	5.14e+00	0.0	28.721478	29.782862	3.5637%	11.10s	255.34s
24	28.919258	29.782867	9.23e-02	4.56e+00	0.0	28.919258	29.782862	2.8997%	11.15s	267.53s
25	29.039661	29.782867	7.99e-02	5.34e+00	0.0	29.039661	29.782862	2.4954%	11.17s	279.31s
26	29.134855	29.782867	6.32e-02	5.02e+00	0.0	29.134855	29.782862	2.1758%	11.20s	291.15s
27	29.231516	29.782867	4.44e-02	3.40e+00	0.0	29.231516	29.782862	1.8512%	11.24s	303.54s
28	29.346041	29.782867	4.36e-02	3.08e+00	0.0	29.346041	29.782862	1.4667%	11.29s	316.08s
29	29.382217	29.782867	2.63e-02	3.40e+00	0.0	29.382217	29.782862	1.3452%	11.32s	328.39s
30	29.508949	29.782867	2.19e-02	2.14e+00	0.0	29.508949	29.782862	9.2e-01%	11.36s	340.82s
31	29.541662	29.782867	1.61e-02	2.00e+00	0.0	29.541662	29.782862	8.1e-01%	11.39s	353.03s
32	29.582359	29.782867	1.46e-02	2.40e+00	0.0	29.582359	29.782862	6.7e-01%	11.43s	365.74s
33	29.613035	29.782867	7.94e-03	1.77e+00	0.0	29.613035	29.782862	5.7e-01%	11.46s	378.12s
34	29.693581	29.782867	6.27e-03	1.33e+00	0.0	29.693581	29.782862	3.0e-01%	11.49s	390.53s
35	29.706371	29.782867	3.29e-03	1.31e+00	0.0	29.706371	29.782862	2.6e-01%	11.51s	402.89s
36	29.758188	29.782867	6.22e-04	6.48e-01	0.0	29.758188	29.782862	8.3e-02%	11.54s	415.26s
37	29.769352	29.782867	1.88e-04	3.22e-01	0.0	29.769352	29.782862	4.5e-02%	11.57s	428.17s
38	29.775732	29.782867	1.92e-04	3.31e-01	0.0	29.775732	29.782862	2.4e-02%	11.60s	440.78s
39	29.776949	29.782867	5.80e-05	1.72e-01	0.0	29.775732	29.782862	2.4e-02%	11.40s	444.48s
40	29.781179	29.782867	8.63e-06	7.68e-02	0.0	29.781179	29.782862	5.6e-03%	11.41s	456.40s

Well, we did . . .


23 28.721478 29.782867 1.03e-01 721478 29.782862 3.5637% 11.10s 255.34s
24 28.919258 29.782867 9.23e-02 19258 29.782862 2.8997% 11.15s 267.53s
25 29.039661 29.782867 7.99e-02 661 29.782862 2.4954% 11.17s 279.31s
26 29.134855 29.782867 6.32e-02 855 29.782862 2.1758% 11.20s 291.15s
27 29.231516 29.782867 4.44e-02 16 29.782862 1.8512% 11.24s 303.54s
28 29.346041 29.782867 4.36e-02 1 29.782862 1.4667% 11.29s 316.08s
29 29.382217 29.782867 2.63e-02 7 29.782862 1.3452% 11.32s 328.39s
30 29.508949 29.782867 2.19e-02 29.782862 9.2e-01% 11.36s 340.82s
31 29.541662 29.782867 1.61e-02 782862 8.1e-01% 11.39s 353.03s
32 29.582359 29.782867 1.46e-02 6.7e-01% 11.43s 365.74s
33 29.613035 29.782867 5.7e-01% 11.46s 378.12s
34 29.693581 29.782867 3.0e-01% 11.49s 390.53s
35 29.706371 29.782867 2.6e-01% 11.51s 402.89s
36 29.758188 3e-02% 11.54s 415.26s
37 29.769352 5e-02% 11.57s 428.17s
38 29.775732 e-02% 11.60s 440.78s
39 29.776949 e-02% 11.40s 444.48s
40 29.781179 -03% 11.41s 456.40s

Kelley 2.0: Outer Approximation

Drawbacks of Kelley's method

- Linear convergence, many iterations (runtime!)
- Almost linear dependent inequalities (numerics!)

Resolved by **Outer Approximation**

- Duran and Grossmann (1986)
- Fletcher and Leyffer (1994)

Kelley 2.0: Outer Approximation

Kelley 2.0: Outer Approximation

Same instance as before
but with some **outer-approximation magic** applied . . .

Kelley 2.0: Outer Approximation

Same instance as before

but with some **outer-approximation magic** applied . . .

```
1 -76.197311  29.782862  7.19e+02 0.00e+00  0.0 -76.197311  29.782862  355.84%    3.82s     3.82s
2  29.782867  29.782862  1.96e-11 6.65e+02  0.0  29.782867  29.782862 -1.5e-05%    8.31s    16.63s
```

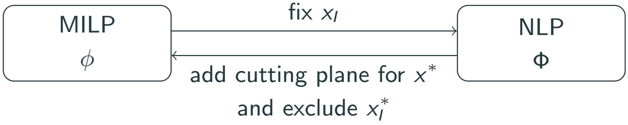
From Kelley . . .

1. Solve an MILP relaxation to obtain x^*
2. Add linear approximation around x^* to the MILP relaxation
3. Repeat until ε-tolerance is fulfilled

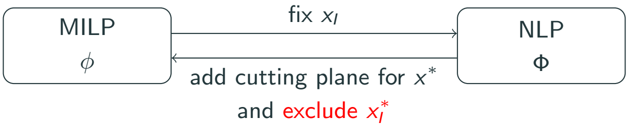
... to Outer Approximation

1. Solve an MILP relaxation to obtain x and lower bound ϕ
2. Solve an NLP with fixed integers x_I to obtain x^* and upper bound $\Phi = \min\{\Phi, f(x^*)\}$
3. Update the MILP relaxation
    - Add Kelley cutting plane for x^*
    - Exclude integer-feasible solution x_I^*
4. Repeat until $\phi \geq \Phi$

1. Solve an MILP relaxation to obtain x and lower bound ϕ
2. Solve an NLP with fixed integers x_I to obtain x^* and upper bound $\Phi = \min\{\Phi, f(x^*)\}$
3. Update the MILP relaxation
   - Add Kelley cutting plane for x^*
   - Exclude integer-feasible solution x_I^*
4. Repeat until $\phi \geq \Phi$



1. Solve an MILP relaxation to obtain x and lower bound ϕ
2. Solve an NLP with fixed integers x_I to obtain x^* and upper bound $\Phi = \min\{\Phi, f(x^*)\}$
3. Update the MILP relaxation
   - Add Kelley cutting plane for x^*
   - Exclude integer-feasible solution x_I^*
4. Repeat until $\phi \geq \Phi$



How to Exclude Integer Solutions?

How to Exclude Integer Solutions?

Simple idea: No-good-cuts

$$\sum_{i \in I : x_i = 0} x_i + \sum_{i \in I : x_i = 1} (1 - x_i) \geq 1$$

How to Exclude Integer Solutions?

Simple idea: No-good-cuts

$$\sum_{i \in I : x_i = 0} x_i + \sum_{i \in I : x_i = 1} (1 - x_i) \geq 1$$

No good!

How to Exclude Integer Solutions?

Simple idea: No-good-cuts

$$\sum_{i \in I : x_i = 0} x_i + \sum_{i \in I : x_i = 1} (1 - x_i) \geq 1$$

No good!

But: Duran and Grossmann had a simple **and** good idea

Mathematical Programming 36 (1986) 307–339
North-Holland

AN OUTER-APPROXIMATION ALGORITHM FOR A CLASS OF MIXED-INTEGER NONLINEAR PROGRAMS

Marco A. DURAN* and Ignacio E. GROSSMANN
Department of Chemical Engineering, Carnegie-Mellon University, Pittsburgh, PA 15213, USA

Received 28 May 1984

How to Exclude Integer Solutions?

Assume you have an integer point x_I^j and assume that the **subproblem**

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & f(x) \\ \text{s.t.} \quad & c(x) \le 0 \\ & x \in X \\ & x_I = x_I^j \end{aligned} \qquad (\mathrm{S}(x_I^j))$$

has a solution x^j

How to Exclude Integer Solutions?

Assume you have an integer point x_I^j and assume that the **subproblem**

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & f(x) \\ \text{s.t.} \quad & c(x) \leq 0 \\ & x \in X \\ & x_I = x_I^j \end{aligned} \qquad (\mathrm{S}(x_I^j))$$

has a solution x^j

Important technicality: a constraint qualification needs to hold at x^j

Recap: Kelley's Cuts

Taylor's first-order approximation

$$p(x; x^0) = c(x^0) + \nabla c(x^0)^\top (x - x^0)$$

1. Since c is convex, $p(x; x^0) \leq 0$ is a valid inequality for all $x \in \Omega$
2. For $x^0 \in S \setminus \Omega$, $p(x; x^0) \leq 0$ cuts off x^0

The Simple and Good Idea

Add first-order approximations for solutions x^j of the subproblem $(\mathrm{S}(x_I^j))$

$$p(x; x^j) = c(x^j) + \nabla c(x^j)^\top (x - x^j)$$

1. Since c is convex, $p(x; x^j) \leq 0$ is a valid inequality for all $x \in \Omega$
2. For x^j, $p(x; x^j) \leq 0$ does not cut off x^j!

Tangential cone of Ω in x

$$T_\Omega(x) := \left\{ d \in \mathbb{R}^n : \exists (x^k)_k \in \Omega, (t_k)_k \in \mathbb{R}_{\geq 0} \text{ such that } \lim_{k\to\infty} x^k = x,\ \lim_{k\to\infty} t_k = 0,\ \text{and } \lim_{k\to\infty} \frac{x^k - x}{t_k} = d \right\}$$

Tangential cone of Ω in x

$$T_\Omega(x) := \Big\{ d \in \mathbb{R}^n : \exists (x^k)_k \in \Omega, (t_k)_k \in \mathbb{R}_{\geq 0} \text{ such that } \lim_{k\to\infty} x^k = x,\ \lim_{k\to\infty} t_k = 0, \text{ and } \lim_{k\to\infty} \frac{x^k - x}{t_k} = d \Big\}$$

In other words: $T_\Omega(x)$ contains all directions d that are tangential to Ω in $x \in \Omega$

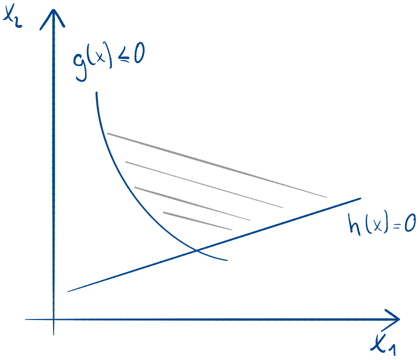
Tangential cone of Ω in x

$$T_\Omega(x) := \Big\{ d \in \mathbb{R}^n : \exists (x^k)_k \in \Omega, (t_k)_k \in \mathbb{R}_{\geq 0} \text{ such that} \\ \lim_{k\to\infty} x^k = x,\ \lim_{k\to\infty} t_k = 0,\ \text{and}\ \lim_{k\to\infty} \frac{x^k - x}{t_k} = d \Big\}$$

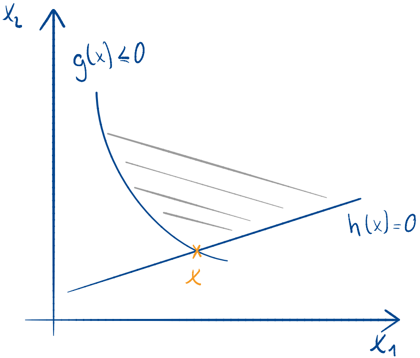
In other words: $T_\Omega(x)$ contains all directions d that are tangential to Ω in $x \in \Omega$

In other words: "$T_\Omega(x)$ contains all feasible directions approaching x"

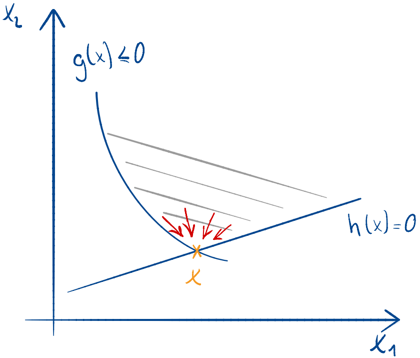
Tangential cone of Ω in x



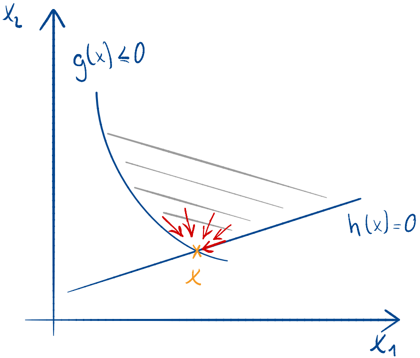
Tangential cone of Ω in x



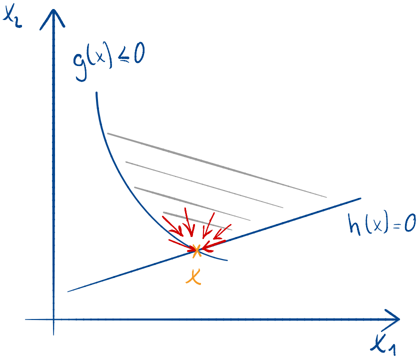
Tangential cone of Ω in x



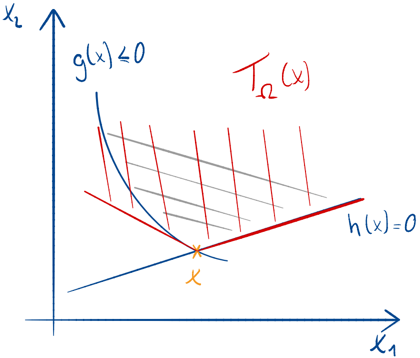
Tangential cone of Ω in x



Tangential cone of Ω in x



Tangential cone of Ω in x

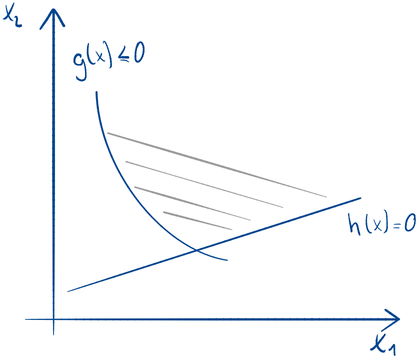


Linearized cone of Ω in x

$$T_\Omega^{\text{lin}} := \{d \in \mathbb{R}^n : d^\top \nabla c_i(x) \leq 0,\ i \in \mathcal{A}(x)\}$$

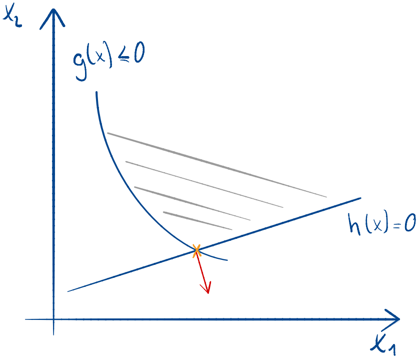
Linearized cone of Ω in x

$$T_\Omega^{\text{lin}} := \{d \in \mathbb{R}^n : d^\top \nabla c_i(x) \leq 0,\ i \in \mathcal{A}(x)\}$$



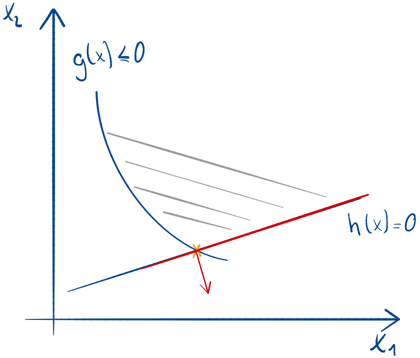
Linearized cone of Ω in x

$$T_\Omega^{\text{lin}} := \{d \in \mathbb{R}^n : d^\top \nabla c_i(x) \leq 0,\ i \in \mathcal{A}(x)\}$$



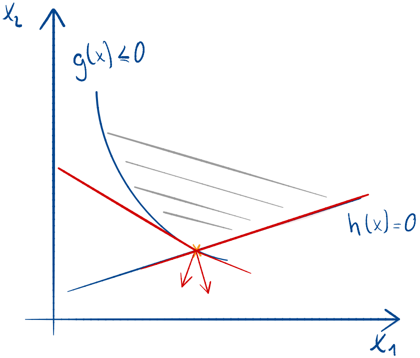
Linearized cone of Ω in x

$$T_{\Omega}^{\text{lin}} := \{d \in \mathbb{R}^n : d^\top \nabla c_i(x) \leq 0,\ i \in \mathcal{A}(x)\}$$



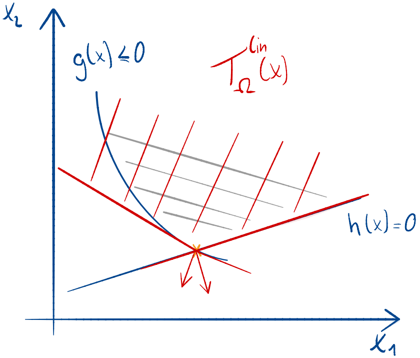
Linearized cone of Ω in x

$$T_{\Omega}^{\text{lin}} := \{d \in \mathbb{R}^n : d^\top \nabla c_i(x) \leq 0,\ i \in \mathcal{A}(x)\}$$



Linearized cone of Ω in x

$$T_\Omega^{\text{lin}} := \{d \in \mathbb{R}^n : d^\top \nabla c_i(x) \leq 0,\ i \in \mathcal{A}(x)\}$$



Obviously, $T_\Omega(x) = T_\Omega^{\text{lin}}(x)$ holds in our example

Obviously, $T_\Omega(x) = T_\Omega^{\text{lin}}(x)$ holds in our example

Abadie Constraint Qualification: A feasible point $x \in \Omega$ fulfills the ACQ if $T_\Omega(x) = T_\Omega^{\text{lin}}$ holds

Obviously, $T_\Omega(x) = T_\Omega^{\text{lin}}(x)$ holds in our example

Abadie Constraint Qualification: A feasible point $x \in \Omega$ fulfills the ACQ if $T_\Omega(x) = T_\Omega^{\text{lin}}$ holds

Optimality condition: If x^* is a local solution and f is continuously differentiable, then

$$\nabla f(x^*)^\top d \geq 0 \quad \text{for all} \quad d \in T_\Omega(x^*)$$

The Simple and Good Idea

Add first-order approximations for solutions x^j of the subproblem $(\mathrm{S}(x_I^j))$

$$p(x; x^j) = c(x^j) + \nabla c(x^j)^\top (x - x^j)$$

1. Since c is convex, $p(x; x^j) \leq 0$ is a valid inequality for all $x \in \Omega$
2. For x^j, $p(x; x^j) \leq 0$ does not cut off x^j

Where the Magic Happens

The set of possible integer assignments

$$\mathcal{X} := \{x^j \in X : x^j \text{ solves } (\mathsf{S}(x_I^j))\}$$

Where the Magic Happens

The set of possible integer assignments

$$\mathcal{X} := \{x^j \in X : x^j \text{ solves } (\mathsf{S}(x_I^j))\}$$

Assume that the Abadie Constraint Qualification holds for all $x \in \mathcal{X}$

The set of possible integer assignments

$$\mathcal{X} := \{x^j \in X : x^j \text{ solves } (\mathsf{S}(x_I^j))\}$$

Assume that the Abadie Constraint Qualification holds for all $x \in \mathcal{X}$

Let $\mathcal{X}^k \subseteq \mathcal{X}$ and consider the **master problem**

$$\begin{aligned} \min_{x\in\mathbb{R}^n,\eta\in\mathbb{R}} \quad & \eta \\ \text{s.t.} \quad & f(x^j) + \nabla f(x^j)^\top (x - x^j) \le \eta, \quad \forall x^j \in \mathcal{X}^k, \\ & c(x^j) + \nabla c(x^j)^\top (x - x^j) \le 0, \quad \forall x^j \in \mathcal{X}^k, \\ & x \in X, \\ & x_i \in \mathbb{Z}, \quad i \in I \end{aligned} \tag{M(k)}$$

Assume x to be a solution of (M(k)) such that $x_I = x_I^\ell$ for a $x^\ell \in \mathcal{X}^k$

Where the Magic Happens

Assume x to be a solution of (M(k)) such that $x_I = x_I^\ell$ for a $x^\ell \in \mathcal{X}^k$

- We already solved the subproblem (S(x_I))

Assume x to be a solution of (M(k)) such that $x_I = x_I^\ell$ for a $x^\ell \in \mathcal{X}^k$

- We already solved the subproblem (S(x_I))
- x must fulfill

$$
\begin{aligned}
f(x^\ell) + \nabla f(x^\ell)^\top (x - x^\ell) &\leq \eta \\
c(x^\ell) + \nabla c(x^\ell)^\top (x - x^\ell) &\leq 0
\end{aligned}
$$

Where the Magic Happens

If $c_i(x^\ell) < 0$ is not active, then any direction is feasible. In particular:

$$(x - x^\ell) \in T_\Omega^{\text{lin}}(x^\ell).$$

Where the Magic Happens

If $c_i(x^\ell) < 0$ is not active, then any direction is feasible. In particular:

$$(x - x^\ell) \in T_\Omega^{\text{lin}}(x^\ell).$$

If $c_i(x^\ell) = 0$ active, then

$$\nabla c_i(x^\ell)^\top (x - x^\ell) \leq 0 \iff (x - x^\ell) \in T_\Omega^{\text{lin}}(x^\ell).$$

If $c_i(x^\ell) < 0$ is not active, then any direction is feasible. In particular:

$$(x - x^\ell) \in T_\Omega^{\text{lin}}(x^\ell).$$

If $c_i(x^\ell) = 0$ active, then

$$\nabla c_i(x^\ell)^\top (x - x^\ell) \leq 0 \iff (x - x^\ell) \in T_\Omega^{\text{lin}}(x^\ell).$$

Since the ACQ holds at x^ℓ, we have

$$(x - x^\ell) \in T_\Omega(x^\ell)$$

Where the Magic Happens

From $(x - x^\ell) \in T_\Omega(x^\ell)$ we know

$$\nabla f(x^\ell)^\top (x - x^\ell) \geq 0$$

Where the Magic Happens

From $(x - x^\ell) \in T_\Omega(x^\ell)$ we know

$$\nabla f(x^\ell)^\top (x - x^\ell) \geq 0$$

Because x is feasible for the master problem

$$f(x^\ell) + \nabla f(x^\ell)(x - x^\ell) \leq \eta \quad \Longleftrightarrow \quad f(x^\ell) \leq \eta$$

Where the Magic Happens

From $(x - x^\ell) \in T_\Omega(x^\ell)$ we know

$$\nabla f(x^\ell)^\top (x - x^\ell) \geq 0$$

Because x is feasible for the master problem

$$f(x^\ell) + \nabla f(x^\ell)(x - x^\ell) \leq \eta \quad \iff \quad f(x^\ell) \leq \eta$$

Because $x^\ell \in \Omega \implies \Phi \leq f(x^\ell)$

Where the Magic Happens

From $(x - x^{\ell}) \in T_{\Omega}(x^{\ell})$ we know

$$\nabla f(x^{\ell})^{\top}(x - x^{\ell}) \geq 0$$

Because x is feasible for the master problem

$$f(x^{\ell}) + \nabla f(x^{\ell})(x - x^{\ell}) \leq \eta \quad \Longleftrightarrow \quad f(x^{\ell}) \leq \eta$$

Because $x^{\ell} \in \Omega \Longrightarrow \Phi \leq f(x^{\ell})$

Altogether, this gives

$$\Phi \leq f(x^{\ell}) \leq \eta = \phi$$

We Just Proved

Lemma
Whenever an integer solution of the master problem appears for the second time, then the corresponding objective function value is greater or equal to the best upper bound.

We Just Proved

Lemma
Whenever an integer solution of the master problem appears for the second time, then the corresponding objective function value is greater or equal to the best upper bound.

Again, in other words:
At most, we need to check one integer solution twice.

We indeed “cut” the integer solutions.

There Is One Thing Left

What if a subproblem (S(x_I^j)) is infeasible?

There Is One Thing Left

What if a subproblem $(\mathrm{S}(x_I^j))$ is infeasible?

Duran and Grossmann add no-good-cuts, but this is still no good ...

There Is One Thing Left

What if a subproblem $(S(x_I^j))$ is infeasible?

Duran and Grossmann add no-good-cuts, but this is still no good . . .

Fletcher and Leyffer (1994) have a solution

Mathematical Programming 66 (1994) 327–349

Solving mixed integer nonlinear programs by outer approximation**

Roger Fletcher, Sven Leyffer*

Department of Mathematics and Computer Science, University of Dundee, Dundee DD1 4HN, Scotland, UK

Received 15 June 1992; revised manuscript received 1 December 1994

Infeasible Subproblems

If a subproblem $(S(x_I^j))$ is infeasible, then solve the feasibility problem

$$\begin{aligned} \min_{x\in\mathbb{R}^n} \quad & \sum_{i\in J^\perp} w_i c_i^+(x) \\ \text{s.t.} \quad & c_i(x) \leq 0, \quad i \in J \\ & x \in X \\ & x_I = x_I^j \end{aligned} \tag{F(x_I^j)}$$

with

- $c_i^+(x) = \max\{c_i(x), 0\}$
- using the weights $w_i > 0$ we can model, e.g., the ℓ_1 or ℓ_∞ norm
- J a set of constraints that can be fulfilled
- $J^\perp$ the set of infeasible constraints

Interpretation: the feasibility problem minimizes the infeasibility

Let $(S(x_I^j))$ be infeasible and x^j be a solution of the feasibility problem $(F(x_I^j))$

Fletcher and Leyffer proved that all x with $x_I = x_I^j$ violate the constraints

$$f(x^j) + \nabla f(x^j)^\top (x - x^j) \le \eta$$
$$c(x^j) + \nabla c(x^j)^\top (x - x^j) \le 0$$

Outer Approximation

Given x^0, set $\phi \leftarrow -\infty$, $\Phi \leftarrow +\infty$, $j \leftarrow 0$, and $\mathcal{X}^{-1} \leftarrow \emptyset$
while $\phi < \Phi$ **do**
  Solve (S(x_I^j)) or (F(x_I^j)) and let the solution be x^j
  if (S(x_I^j)) is feasible and $f(x^j) < \Phi$ **then**
    Update current best point $x^* \leftarrow x^j$ and $\Phi \leftarrow f(x^j)$
  end if
  Linearize f and c at x^j and set $\mathcal{X}^j \leftarrow \mathcal{X}^{j-1} \cup \{x^j\}$
  Solve (M(j)) and let the solution be x^{j+1}. Set $\phi \leftarrow f(x^{j+1})$ and $j \leftarrow j+1$
end while

Theorem:
If the Abadie constraint qualification holds at the solution of every subproblem $(\mathrm{S}(x_I^j))$ and if the number of integer points in X is finite, then the outer-approximation algorithm terminates in a finite number of steps with an optimal solution or with an indication that the problem is infeasible.

Theorem:
If the Abadie constraint qualification holds at the solution of every subproblem $(S(x_I^j))$ and if the number of integer points in X is finite, then the outer-approximation algorithm terminates in a finite number of steps with an optimal solution or with an indication that the problem is infeasible.

Proof:
Follows directly from the previous slides.

Extensions

- Hotstart the master problems: initial values, cutoff values, etc.
- Stop master problem with first "improving solution"
- Add linearization cuts for all feasible points of the master problem that we encounter while solving the master problem

(Generalized) Benders Decomposition

(Generalized) Benders Decomposition

- Invented by Jacques Benders in 1962

(Generalized) Benders Decomposition

- Invented by Jacques Benders in 1962
- Algorithm for problems with a special structure
  - Problem has "easy" and "complicated" variables
  - Fixing the complicated variables results in a linear problem

Outer Approximation vs. Generalized Benders Decomposition

(Generalized) Benders Decomposition

- Invented by Jacques Benders in 1962
- Algorithm for problems with a special structure
  - Problem has "easy" and "complicated" variables
  - Fixing the complicated variables results in a linear problem
- Generalized by Geoffrion in 1972 to nonlinear subproblems

(Generalized) Benders Decomposition

- Invented by Jacques Benders in 1962
- Algorithm for problems with a special structure
  - Problem has "easy" and "complicated" variables
  - Fixing the complicated variables results in a linear problem
- Generalized by Geoffrion in 1972 to nonlinear subproblems
- Algorithmic idea
  - Decompose the variables
  - Relaxed master problem over the complicated variables
  - Subproblem with fixed complicated variables
  - Successively derive cuts from the subproblem by duality theory

(Generalized) Benders Decomposition

- Invented by Jacques Benders in 1962
- Algorithm for problems with a special structure
  - Problem has "easy" and "complicated" variables
  - Fixing the complicated variables results in a linear problem
- Generalized by Geoffrion in 1972 to nonlinear subproblems
- Algorithmic idea
  - Decompose the variables
  - Relaxed master problem over the complicated variables
  - Subproblem with fixed complicated variables
  - Successively derive cuts from the subproblem by duality theory

This sounds a lot like outer approximation, doesn't it?!

Outer Approximation vs. Generalized Benders Decomposition

Outer approximation cuts

$$
\begin{aligned}
f(x^j) + \nabla f(x^j)^\top (x - x^j) &\leq \eta, \\
c(x^j) + \nabla c(x^j)^\top (x - x^j) &\leq 0
\end{aligned}
$$

Outer Approximation vs. Generalized Benders Decomposition

Outer approximation cuts

$$\begin{aligned} f(x^j) + \nabla f(x^j)^\top (x - x^j) &\leq \eta, \\ c(x^j) + \nabla c(x^j)^\top (x - x^j) &\leq 0 \end{aligned}$$

Generalized Benders cut

$$f(x^j) + \left(\nabla_I f(x^j) + \sum_{i=1}^{m} \lambda^j \nabla_I c_i^j \right)^\top (x_I - x_I^j) \leq \eta$$

Outer Approximation vs. Generalized Benders Decomposition

Outer approximation cuts

$$
\begin{aligned}
f(x^j) + \nabla f(x^j)^\top (x - x^j) &\le \eta, \\
c(x^j) + \nabla c(x^j)^\top (x - x^j) &\le 0
\end{aligned}
$$

Generalized Benders cut

$$
f(x^j) + \left(\nabla_I f(x^j) + \sum_{i=1}^{m} \lambda^j \nabla_I c_i^j \right)^\top (x_I - x_I^j) \le \eta
$$

Benders cuts are “weighted” outer approximation cuts

Outer Approximation vs. Generalized Benders Decomposition

Outer approximation cuts

$$f(x^j) + \nabla f(x^j)^\top (x - x^j) \le \eta,$$
$$c(x^j) + \nabla c(x^j)^\top (x - x^j) \le 0$$

Generalized Benders cut

$$f(x^j) + \left(\nabla_I f(x^j) + \sum_{i=1}^{m} \lambda^j \nabla_I c_i^j \right)^\top (x_I - x_I^j) \le \eta$$

Benders cuts are "weighted" outer approximation cuts

Benders cuts are dense and weaker than outer approximation cuts

Outer Approximation vs. Branch-and-Bound

Hardness of MINLPs stems from **nonlinearities** and **integrality constraints**

Outer Approximation vs. Branch-and-Bound

Hardness of MINLPs stems from **nonlinearities** and **integrality constraints**

Rationale of Branch-and-Bound

- Get rid of the integer variables by branching and solve only NLP relaxations
- Single-tree but many NLPs

Outer Approximation vs. Branch-and-Bound

Hardness of MINLPs stems from **nonlinearities** and **integrality constraints**

Rationale of Branch-and-Bound

- Get rid of the integer variables by branching and solve only NLP relaxations
- Single-tree but many NLPs

Rationale of Kelley

- Approximate nonlinearities by only solving MILPs
- Multi-tree but no NLPs

Outer Approximation vs. Branch-and-Bound

Hardness of MINLPs stems from **nonlinearities** and **integrality constraints**

Rationale of Branch-and-Bound

- Get rid of the integer variables by branching and solve only NLP relaxations
- Single-tree but many NLPs

Rationale of Kelley

- Approximate nonlinearities by only solving MILPs
- Multi-tree but no NLPs

Rationale of Outer Approximation

- Approximate nonlinearities and get rid of integrality constraints by fixing
- Solve MILPs and NLPs alternatingly
- Multi-tree with few NLPs

Outer Approximation vs. Branch-and-Bound

Searching multiple branch-and-bound trees sounds inefficient

Outer Approximation vs. Branch-and-Bound

Searching multiple branch-and-bound trees ~~sounds~~ **is** inefficient

Outer Approximation vs. Branch-and-Bound

Best of Both Worlds: LP/NLP-Based Branch-and-Bound

- Introduced by Quesada and Grossmann (1992)
- Can be seen as a hybrid algorithm between nonlinear branch-and-bound and outer approximation

Best of Both Worlds: LP/NLP-Based Branch-and-Bound

- Introduced by Quesada and Grossmann (1992)
- Can be seen as a hybrid algorithm between nonlinear branch-and-bound and outer approximation

Rationale

- Relax nonlinearities **and** integrality constraints
- Branch on integralities and solve **LPs** at every branch-and-bound node
- Whenever a node solution is integral, solve the corresponding NLP
- Globally add the outer-approximation cuts for this NLP solution

Best of Both Worlds: LP/NLP-Based Branch-and-Bound

Given x_I^0, set $\phi \leftarrow -\infty$, $\Phi \leftarrow +\infty$, $j \leftarrow 0$, $\mathcal{X}^j \leftarrow \emptyset$,
initialize the set of open node problems $O \leftarrow \{\text{LP}(\mathcal{X}^j, -\infty, \infty)\}$
while $O \neq \emptyset$ **do**
  Pick an LP: $O = O \setminus \{\text{LP}(\mathcal{X}^j, l, u)\}$
  Solve $\text{LP}(\mathcal{X}^j, l, u)$ and let its solution be $x^{(l,u)}$
  if $\text{LP}(\mathcal{X}^j, l, u)$ is infeasible or $f(x^{(l,u)}) \geq \Phi$ **then**
    Node can be pruned
  else if $x_I^{(l,u)}$ is integral **then**
    Set $x_I^j = x_I^{(l,u)}$ and solve $(\text{S}(x_I^j))$ or $(\text{F}(x_I^j))$ and let its solution be x^j
    Linearize f and c at x^j and set $\mathcal{X}^{j+1} \leftarrow \mathcal{X}^j \cup \{x^j\}$
    if $(\text{S}(x_I^j))$ is feasible and $f(x^j) < \Phi$ **then**
      Update best point $x^* \leftarrow x^j$ and $\Phi \leftarrow f(x^j)$
    end if
    Re-add the LP: $O = O \cup \{\text{LP}(\mathcal{X}^{j+1}, l, u)\}$
    Set $j \leftarrow j + 1$
  else
    Branch on a fractional variable and update O
  end if
end while

LP/NLP-Based Branch-and-Bound in Practice

LP/NLP-Based Branch-and-Bound in Practice

- Should be implemented within modern MILP solvers
  - Advanced MILP search (strong branching, adaptive node selection)
  - Effective cut management

LP/NLP-Based Branch-and-Bound in Practice

- Should be implemented within modern MILP solvers
  - Advanced MILP search (strong branching, adaptive node selection)
  - Effective cut management
- Add cuts also for points that are not integer-feasible

LP/NLP-Based Branch-and-Bound in Practice

- Should be implemented within modern MILP solvers
  - Advanced MILP search (strong branching, adaptive node selection)
  - Effective cut management
- Add cuts also for points that are not integer-feasible
- See Abishek et al. (2010) and Bonami et al. (2008) for details

LP/NLP-Based Branch-and-Bound in Practice

- Should be implemented within modern MILP solvers
  - Advanced MILP search (strong branching, adaptive node selection)
  - Effective cut management
- Add cuts also for points that are not integer-feasible
- See Abishek et al. (2010) and Bonami et al. (2008) for details

Is it faster then multi-tree outer approximation?

LP/NLP-Based Branch-and-Bound in Practice

- Should be implemented within modern MILP solvers
  - Advanced MILP search (strong branching, adaptive node selection)
  - Effective cut management
- Add cuts also for points that are not integer-feasible
- See Abishek et al. (2010) and Bonami et al. (2008) for details

Is it faster then multi-tree outer approximation?

It depends . . .

LP/NLP-Based Branch-and-Bound in Practice

- Should be implemented within modern MILP solvers
  - Advanced MILP search (strong branching, adaptive node selection)
  - Effective cut management
- Add cuts also for points that are not integer-feasible
- See Abishek et al. (2010) and Bonami et al. (2008) for details

Is it faster then multi-tree outer approximation?

It depends . . .

- Limitations of MILP solvers (callbacks!)

LP/NLP-Based Branch-and-Bound in Practice

- Should be implemented within modern MILP solvers
  - Advanced MILP search (strong branching, adaptive node selection)
  - Effective cut management
- Add cuts also for points that are not integer-feasible
- See Abishek et al. (2010) and Bonami et al. (2008) for details

Is it faster then multi-tree outer approximation?

It depends ...

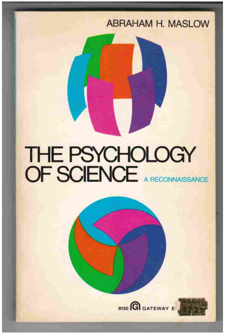
- Limitations of MILP solvers (callbacks!)
- Problem-specific

3. MILP-Based Reformulations: Overview

3. MILP-Based Reformulations

Law of the instrument

> *"If all you have is a hammer,*
> *everything looks like a nail."*
> *— Abraham Maslow, 1966*



The Law of the Instrument in the MILP World

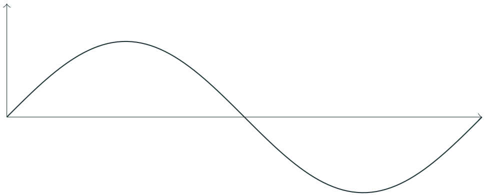
- Discrete optimizers like integer variables and linear problems
- Problem: nonlinearities

The Law of the Instrument in the MILP World

- Discrete optimizers like integer variables and linear problems
- Problem: nonlinearities
- Remedy: linearization

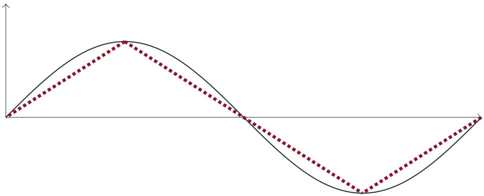
The Law of the Instrument in the MILP World

- Discrete optimizers like integer variables and linear problems
- Problem: nonlinearities
- Remedy: linearization



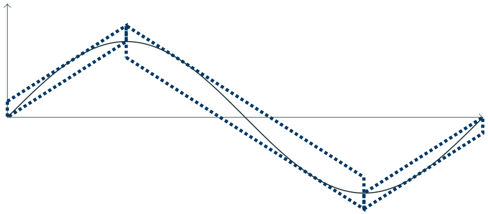
The Law of the Instrument in the MILP World

- Discrete optimizers like integer variables and linear problems
- Problem: nonlinearities
- Remedy: linearization



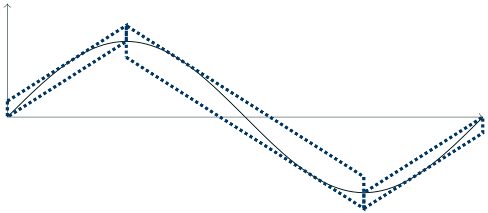
The Law of the Instrument in the MILP World

- Discrete optimizers like integer variables and linear problems
- Problem: nonlinearities
- Remedy: linearization



The Law of the Instrument in the MILP World

- Discrete optimizers like integer variables and linear problems
- Problem: nonlinearities
- Remedy: linearization



Resulting problem is linear and mixed-integer:

Application of branch-and-bound based methods

The Law of the Instrument in the NLP World

- Continuous optimizers like nonlinear problems
- Problem: integrality constraints
- Remedy: continuous reformulation

- Continuous optimizers like nonlinear problems
- Problem: integrality constraints
- Remedy: continuous reformulation

Continuous Reformulation

Replace integer variable with

- one or more continuous variables and
- one or more (potentially nonlinear) constraints.

The Law of the Instrument in the NLP World

- Continuous optimizers like nonlinear problems
- Problem: integrality constraints
- Remedy: continuous reformulation

Continuous Reformulation

Replace integer variable with

- one or more continuous variables and
- one or more (potentially nonlinear) constraints.

Resulting problem is continuous:

Application of nonlinear optimization techniques

Drawbacks and Advantages of the NLP Approach

Drawbacks

- Continuous reformulation
  $\rightarrow$ nonconvex problems
  - NLP methods only yield local minima
- NLP methods are not as stable as MILP methods
- Badly suited for problems with many discrete variables

Drawbacks and Advantages of the NLP Approach

Drawbacks

- Continuous reformulation $\rightarrow$ nonconvex problems
  - NLP methods only yield local minima
- NLP methods are not as stable as MILP methods
- Badly suited for problems with many discrete variables

Advantages

- Significantly faster running times compared to the MILP approach
- Well suited for problems with only a few discrete variables but many nonlinearities
- Physical accuracy is easier to achieve

The easy case ...

Separable Functions

A function $\phi : \mathbb{R}^d \to \mathbb{R}$ is called *separable* if it can be written as a sum of univariate functions $\phi_i : \mathbb{R} \to \mathbb{R}$, $i = 1, \ldots, d$:

$$\phi(x_1, \ldots, x_d) = \sum_{i=1}^{d} \phi_i(x_i).$$

Given

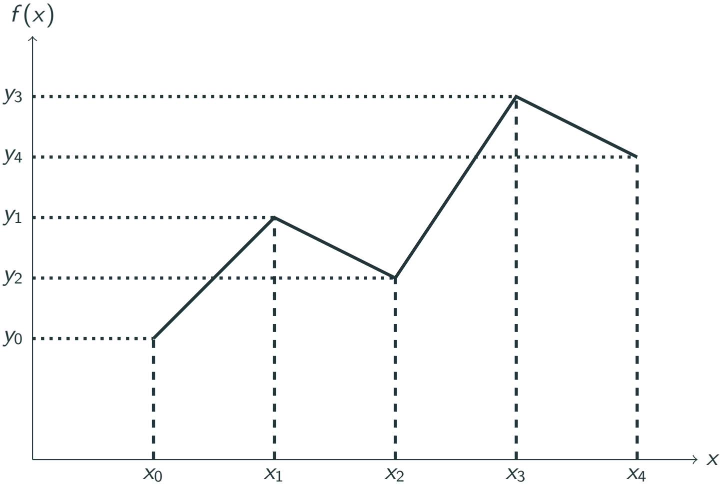
Continuous and univariate function

$$\phi : \mathbb{R} \to \mathbb{R}.$$

Goal

Integration of a piecewise linearization f of ϕ over a given finite interval $[a, b] \subset \mathbb{R}$ into an MILP model

The idea dates back to Markowitz and Manne (1957) as well as Dantzig (1960)


$f(x)$
y_3
y_4
y_1
y_2
y_0
x_0
x_1
x_2
x_3
x_4
x

1d Functions: Convex Combination Method

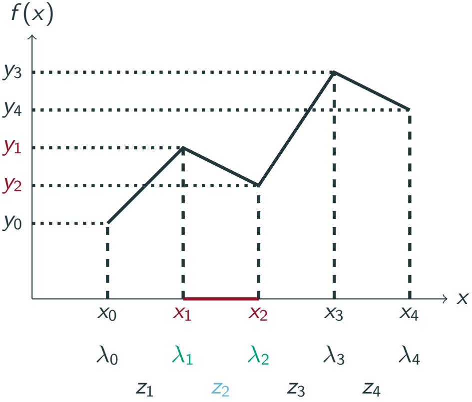
Idea: Express the function value at point x as a convex combination of the function values at the neighboring sampling points.

Idea: Express the function value at point x as a convex combination of the function values at the neighboring sampling points.

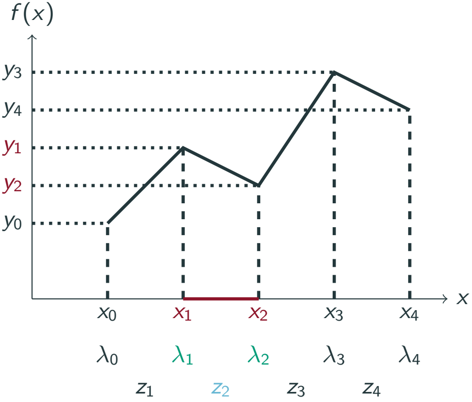
Set $z_0 = z_{n+1} = 0$ and

$$
\begin{aligned}
x &= \sum_{i=0}^{n} \lambda_i x_i, & y &= \sum_{i=0}^{n} \lambda_i y_i, \\
\sum_{i=0}^{n} \lambda_i &= 1, & \sum_{i=1}^{n} z_i &= 1, \\
\lambda_i &\leq z_i + z_{i+1} & &\text{for all } i = 0, \dots, n, \\
\lambda_i &\geq 0 & &\text{for all } i = 0, \dots, n, \\
z_i &\in \{0, 1\} & &\text{for all } i = 1, \dots, n.
\end{aligned}
$$

1d Functions: Convex Combination Method (Example)

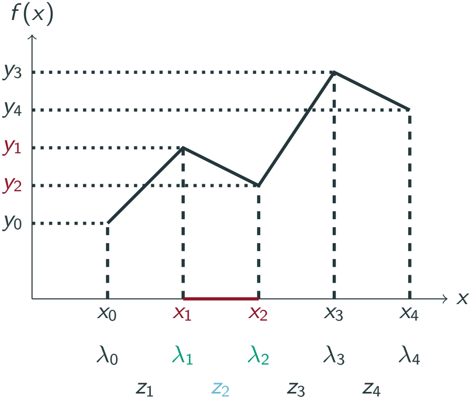


1d Functions: Convex Combination Method (Example)



- $z_{i+1} = 1$ denotes the active interval $i+1$
- $\lambda_i, \lambda_{i+1} \geq 0$ if $x \in [x_i, x_{i+1}]$
- $\lambda_i + \lambda_{i+1} = 1$
- $\lambda_j = 0$ for all $j \notin \{i, i+1\}$
- $x = \sum_{i=0}^{n} \lambda_i x_i$: convex combination of x values
- $y = \sum_{i=0}^{n} \lambda_i y_i$: convex combination of y values

1d Functions: Convex Combination Method (Example)



- $z_{i+1} = 1$ denotes the active interval $i + 1$
- $\lambda_i, \lambda_{i+1} \geq 0$ if $x \in [x_i, x_{i+1}]$
- $\lambda_i + \lambda_{i+1} = 1$
- $\lambda_j = 0$ for all $j \notin \{i, i+1\}$
- $x = \sum_{i=0}^{n} \lambda_i x_i$: convex combination of x values
- $y = \sum_{i=0}^{n} \lambda_i y_i$: convex combination of y values

By the way: the λ_i variables form an SOS-2 set

SOS-1 Type Binary Variables and Branching

Why are SOS-1 type binary variables so nice?

SOS-1 Type Binary Variables and Branching

Why are SOS-1 type binary variables so nice?

- Think of an integer variable $z_i \in \mathbb{Z}$ and a fractional node solution, e.g., $z_i = 42.5$
- Branching on this variable leads to two subproblems with the additional constraints
  - $z_i \leq 42$
  - $z_i \geq 43$
- Branching on fractional binary variables leads to two new subproblems with the fixations $z_i = 0$ and $z_i = 1$

... and that's it!

SOS-1 Type Binary Variables and Branching

Why are SOS-1 type binary variables so nice?

- Think of an integer variable $z_i \in \mathbb{Z}$ and a fractional node solution, e.g., $z_i = 42.5$
- Branching on this variable leads to two subproblems with the additional constraints
  - $z_i \leq 42$
  - $z_i \geq 43$
- Branching on fractional binary variables leads to two new subproblems with the fixations $z_i = 0$ and $z_i = 1$

... and that's it!

For an SOS-1 set of binary variables $z_i \in \{0, 1\}^n$ with $\sum_{i=1}^n z_i = 1$,
a branching with $z_i = 1$ fixes all other variables to 0!

1d Functions: Incremental Method

Idea: A value $x \in [x_{i-1}, x_i]$ can be written as

$$x = x_{i-1} + (x_i - x_{i-1})\delta_i, \quad \delta_i \in [0, 1].$$

1d Functions: Incremental Method

Idea: A value $x \in [x_{i-1}, x_i]$ can be written as

$$x = x_{i-1} + (x_i - x_{i-1})\delta_i, \quad \delta_i \in [0,1].$$

Model of the incremental method:

$$x = x_0 + \sum_{i=1}^{n}(x_i - x_{i-1})\delta_i, \qquad y = y_0 + \sum_{i=1}^{n}(y_i - y_{i-1})\delta_i,$$

$$z_i \le \delta_i \qquad \text{for all } i = 1, \dots, n-1,$$

$$\delta_{i+1} \le z_i \qquad \text{for all } i = 1, \dots, n-1,$$

$$z_i \in \{0,1\} \qquad \text{for all } i = 1, \dots, n-1,$$

$$\delta_1 \le 1, \delta_n \ge 0$$

Idea: A value $x \in [x_{i-1}, x_i]$ can be written as

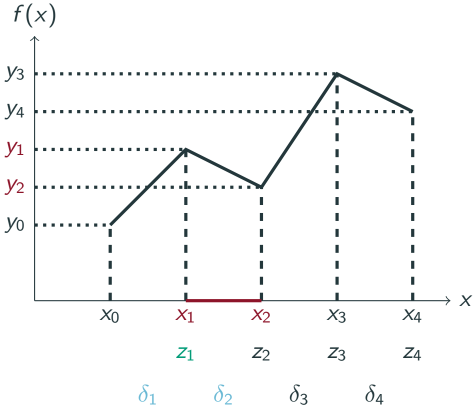
$$x = x_{i-1} + (x_i - x_{i-1})\delta_i, \quad \delta_i \in [0,1].$$

Model of the incremental method:

$$\begin{aligned}
x &= x_0 + \sum_{i=1}^{n}(x_i - x_{i-1})\delta_i, & y &= y_0 + \sum_{i=1}^{n}(y_i - y_{i-1})\delta_i,\\
z_i &\leq \delta_i & &\text{for all } i = 1, \dots, n-1,\\
\delta_{i+1} &\leq z_i & &\text{for all } i = 1, \dots, n-1,\\
z_i &\in \{0,1\} & &\text{for all } i = 1, \dots, n-1,\\
\delta_1 &\leq 1, \delta_n \geq 0 & &
\end{aligned}$$

Filling condition: $\delta_{i+1} \leq z_i \leq \delta_i \Rightarrow (\delta_{i+1} > 0 \Rightarrow \delta_i = 1)$

1d Functions: Incremental Method (Example)



- $x = x_0 + \sum_{i=1}^{n}(x_i - x_{i-1})\delta_i$
  $\Rightarrow \delta_1 = 1, \delta_2 \geq 0,$
  $\delta_3 = \delta_4 = 0$
- $z_1 = 1, z_2 = z_3 = z_4 = 0$

Comparison: Convex Combination Method vs. Incremental Method

Setting

Minimization of a piecewise linear function subject to linear constraints

Comparison: Convex Combination Method vs. Incremental Method

Setting

Minimization of a piecewise linear function subject to linear constraints

Properties

- LP relaxation of the incremental method
  always gives an integer-feasible point
- This is not the case for the convex combination method
- Polyhedron of the incremental method is strictly contained in the polyhedron of the convex combination method

Piecewise Linear Modeling: Pros & Cons

Pros

- Enables us to model nonlinear functions approximately in an MILP

Piecewise Linear Modeling: Pros & Cons

Pros

- Enables us to model nonlinear functions approximately in an MILP

Cons

- Linearization error: Let $\phi \in \mathcal{C}^2([x_0, x_n])$ be the given nonlinear function and let f be the corresponding piecewise linear approximation over $[x_0, x_n]$. Then, we have

$$\|\phi - f\|_\infty \leq h^2 \frac{\|\phi''\|_\infty}{8}, \quad h := \max_{i=1,\ldots,n} \{x_i - x_{i-1}\}.$$

Piecewise Linear Modeling: Pros & Cons

Pros

- Enables us to model nonlinear functions approximately in an MILP

Cons

- Linearization error: Let $\phi \in \mathcal{C}^2([x_0, x_n])$ be the given nonlinear function and let f be the corresponding piecewise linear approximation over $[x_0, x_n]$. Then, we have

$$\|\phi - f\|_\infty \leq h^2 \frac{\|\phi''\|_\infty}{8}, \quad h := \max_{i=1,\ldots,n} \{x_i - x_{i-1}\}.$$

- Error can be controlled by h

Piecewise Linear Modeling: Pros & Cons

Pros

- Enables us to model nonlinear functions approximately in an MILP

Cons

- Linearization error: Let $\phi \in \mathcal{C}^2([x_0, x_n])$ be the given nonlinear function and let f be the corresponding piecewise linear approximation over $[x_0, x_n]$. Then, we have

$$\|\phi - f\|_\infty \leq h^2 \frac{\|\phi''\|_\infty}{8}, \quad h := \max_{i=1,\dots,n} \{x_i - x_{i-1}\}.$$

- Error can be controlled by h
- Problem: Reduction of $h \to$ more binary variables!

Piecewise Linear Modeling: Pros & Cons

Pros

- Enables us to model nonlinear functions approximately in an MILP

Cons

- Linearization error: Let $\phi \in \mathcal{C}^2([x_0, x_n])$ be the given nonlinear function and let f be the corresponding piecewise linear approximation over $[x_0, x_n]$. Then, we have

$$\|\phi - f\|_\infty \leq h^2 \frac{\|\phi''\|_\infty}{8}, \quad h := \max_{i=1,\dots,n} \{x_i - x_{i-1}\}.$$

- Error can be controlled by h
- Problem: Reduction of $h \rightarrow$ more binary variables!
- Compromise between accuracy and tractability/practability

. . . and there's a lot more!

- Multiple-Choice Method
- Disaggregated Convex Combination Method
- Logarithmic Model
- . . .

What if the nonlinearity is not separable?

Consider the reformulation

$$x_1 x_2 = y_1^2 - y_2^2$$

with

$$y_1 = \frac{1}{2}(x_1 + x_2), \quad y_1 = \frac{1}{2}(x_1 - x_2)$$

and $y_1, y_2 \in \mathbb{R}$.

What if the nonlinearity is not separable?

Other idea: take the logarithm

The constraint

$$y = x_1 x_2$$

with $x_1, x_2 > 0$ is equivalent to

$$\ln(y) = \ln(x_1) + \ln(x_2).$$

What if the nonlinearity is not separable?

Other idea: take the logarithm

The constraint

$$y = x_1 x_2$$

with $x_1, x_2 > 0$ is equivalent to

$$\ln(y) = \ln(x_1) + \ln(x_2).$$

However, . . .

- there is no systematic way to reduce multivariate to univariate functions
- errors introduced by piecewise linearizing the separate univariate functions may accumulate and amplify

Further Topics

- How to obtain the piecewise linear approximations
  - Approximation theory
  - Best choice of break points can itself be considered as an optimization problem that needs to be solved up-front
- Multivariate linearizations
  - Convex combination method
  - Incremental method
- From piecewise linear approximations to piecewise linear relaxations

Nonconvex MINLP: What's the challenge?

$$\begin{array}{ll} \min & f(x) \\ \text{s.t.} & c(x) \leq 0, \quad x \in X, \quad x_i \in \mathbb{Z} \quad \text{for all } i \in I \end{array}$$

Problem

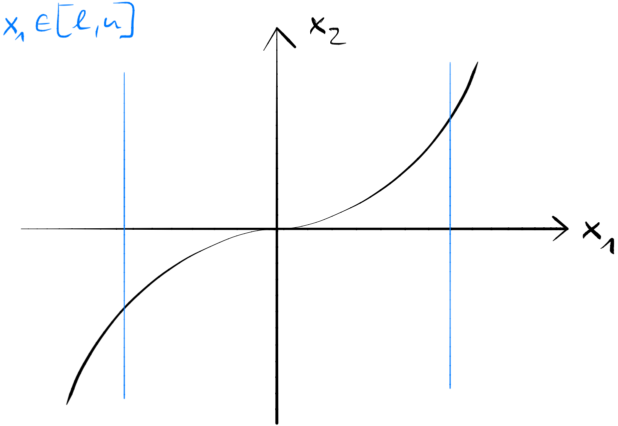
- f and/or at least one of the c_j are nonconvex
- Feasible set is nonconvex ... even after relaxing the integrality constraints
- Local solutions do not define valid bounds
- Very hard problem

Piecewise Linearization

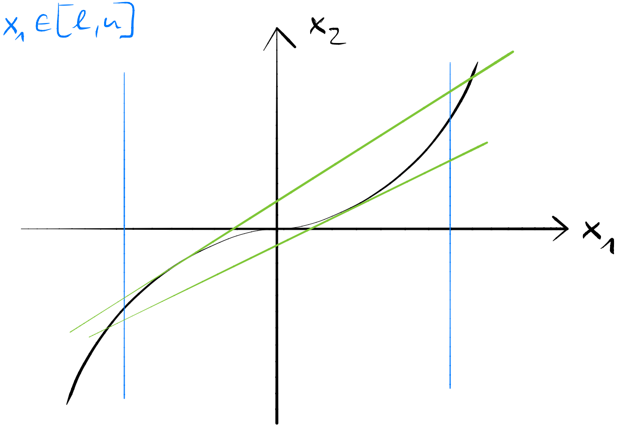
We already know a solution approach: piecewise linear approximations

- Replace nonlinear and nonconvex functions
  with piecewise linear approximations
- Allows to use MILP solvers
- If possible, use separability of multivariate and nonconvex functions
- 2 goals:
  - Compute a sufficiently accurate approximation
  - Minimize the number of additionally required binary variables
- Methods
  - Convex combination method
  - Incremental method
  - Multiple choice method
  - ...

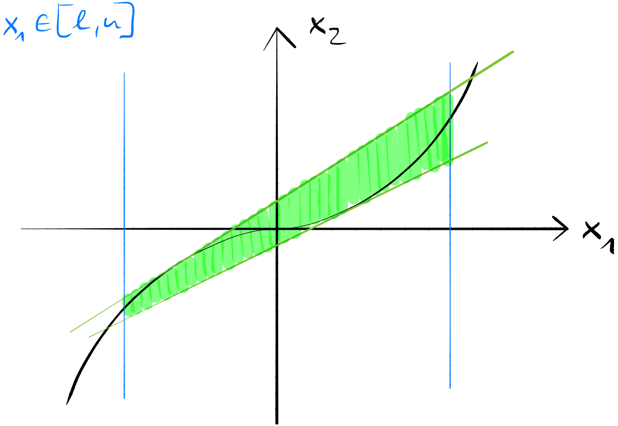
What about using branch-and-bound "directly"?



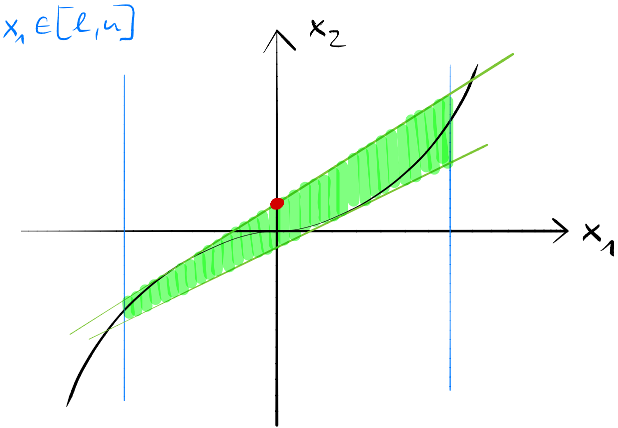
What about using branch-and-bound “directly”?



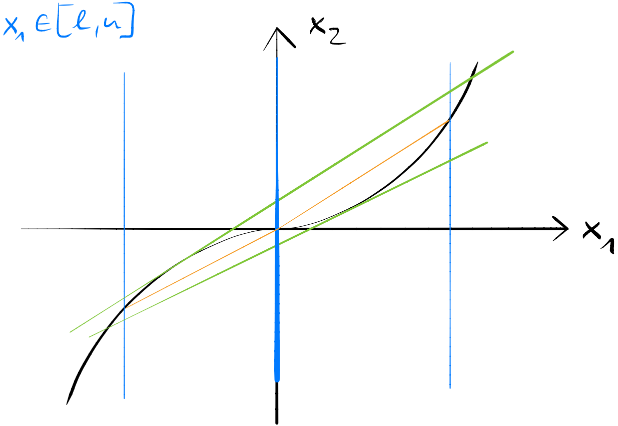
What about using branch-and-bound “directly”?



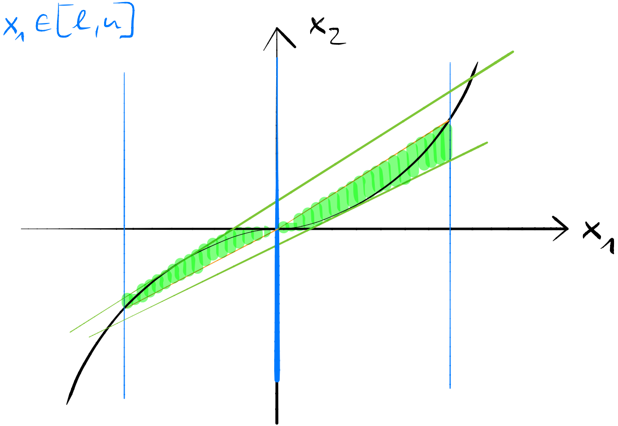
What about using branch-and-bound “directly”?



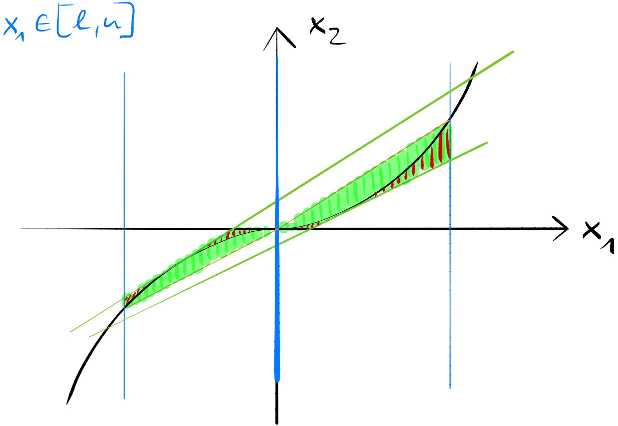
What about using branch-and-bound "directly"?



What about using branch-and-bound "directly"?



What about using branch-and-bound “directly”?



The Nonconvex-MINLP-To-Do-List

We need to know how ...

1. ... to automatically construct polyhedral
   and/or convex relaxations of nonconvex constraints
   - → This leads to lower bounds on the optimal objective function value
2. ... to set up a branching on continuous variables
   - → This leads to a procedure for partitioning the feasible set of a subproblem

The Nonconvex-MINLP-To-Do-List

We need to know how ...

1. ... to automatically construct polyhedral
   and/or convex relaxations of nonconvex constraints
   - $\rightarrow$ This leads to lower bounds on the optimal objective function value
2. ... to set up a branching on continuous variables
   - $\rightarrow$ This leads to a procedure for partitioning the feasible set of a subproblem

Open question: Does this lead to finite termination/convergence?

Factorable Functions

Definition

A function $f : \mathbb{R}^n \to \mathbb{R}$ is called *factorable* if it can be written as a sum of products of univariate functions out of a given set $\mathcal{O}$, whose arguments are variables, constants, or other factorable functions.

Definition

A function $f : \mathbb{R}^n \to \mathbb{R}$ is called *factorable* if it can be written as a sum of products of univariate functions out of a given set $\mathcal{O}$, whose arguments are variables, constants, or other factorable functions.

- Example

$$\mathcal{O} = \{+, \times, /, \hat{}, \sin, \cos, \exp, \log, |\cdot|\}$$

- Examples of non-factorable functions
  - Integrals $\int_{x_0}^{x} f(x)\,\mathrm{d}x$ with unknown antiderivative
  - Black-box functions (e.g., function evaluation = simulation run)

Definition

A function $f : \mathbb{R}^n \to \mathbb{R}$ is called *factorable* if it can be written as a sum of products of univariate functions out of a given set $\mathcal{O}$, whose arguments are variables, constants, or other factorable functions.

- Example

$$\mathcal{O} = \{+, \times, /, \hat{}, \sin, \cos, \exp, \log, |\cdot|\}$$

- Examples of non-factorable functions
    - Integrals $\int_{x_0}^{x} f(x)\,\mathrm{d}x$ with unknown antiderivative
    - Black-box functions (e.g., function evaluation = simulation run)

We need to know the symbolic information about the functions.

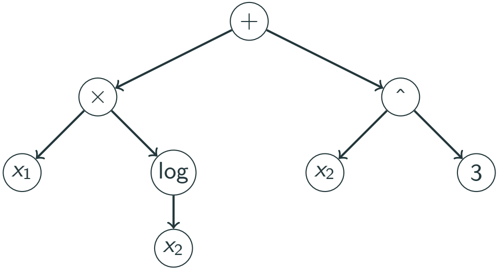
Factorable Functions & Expression Trees

- Factorable functions $\leftrightarrow$ *expression trees*
- Expression tree: rooted tree with constants or variables as leafs and n-ary operations as inner nodes

- Factorable functions $\leftrightarrow$ *expression trees*
- Expression tree: rooted tree with constants or variables as leafs and *n*-ary operations as inner nodes

Example

$$f(x_1, x_2) = x_1 \log x_2 + x_2^3$$



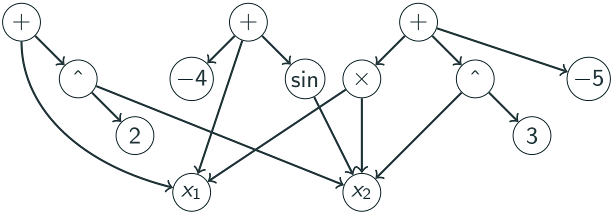
Factorable MINLPs

- If the entire MINLP only contains factorable functions, then the entire MINLP can be represented as a generalized expression tree.
- Result: *DAG* (directed acyclic graph)

Factorable MINLPs

- If the entire MINLP only contains factorable functions, then the entire MINLP can be represented as a generalized expression tree.
- Result: *DAG* (directed acyclic graph)

$$\begin{aligned} \min_{x_1, x_2} \quad & x_1 + x_2^2 \\ \text{s.t.} \quad & x_1 + \sin x_2 \le 4, \quad x_1 x_2 + x_2^3 \le 5, \\ & x_1 \in [-4, 4] \cap \mathbb{Z}, \quad x_2 \in [0, 10] \cap \mathbb{Z} \end{aligned}$$



$$\begin{aligned} \min \quad & x_{n+q} \\ \text{s.t.} \quad & x_k = \theta_k(x), \quad \theta_k \in \mathcal{O}, \quad k = n+1, n+2, \dots, n+q, \\ & l_i \le x_i \le u_i, \quad k = 1, 2, \dots, n+q, \\ & x \in X, \quad x_i \in \mathbb{Z} \quad \text{for all } i \in I \end{aligned}$$

- Variable bounds can be explicitly stated (or implicitly as part of $x \in X$)
- q new *auxiliary* variables
  - Restricted by $\theta_k \in \mathcal{O}$
- Convention: x_{n+q} replaces the objective function

The MINLP

$$\begin{aligned} \min \quad & x_1 + x_2^2 \\ \text{s.t.} \quad & x_1 + \sin x_2 \le 4, \quad x_1 x_2 + x_2^3 \le 5, \\ & x_1 \in [-4, 4] \cap \mathbb{Z}, \quad x_2 \in [0, 10] \cap \mathbb{Z} \end{aligned}$$

becomes

$$\begin{array}{lllll} \min & x_9 & & & \\ \text{s.t.} & x_3 = \sin x_2, & x_7 = x_5 + x_6 - 5, & 0 \le x_2 \le 10, & 0 \le x_6 \le 1000, \\ & x_4 = x_1 + x_3 - 4, & x_8 = x_2^2, & -1 \le x_3 \le 1, & -45 \le x_7 \le 0, \\ & x_5 = x_1 x_2, & x_9 = x_1 + x_8, & -9 \le x_4 \le 0, & 0 \le x_8 \le 100, \\ & x_6 = x_2^3, & -4 \le x_1 \le 4, & -40 \le x_5 \le 40, & -4 \le x_9 \le 104, \\ & x_1, x_2, x_5, x_6, x_7, x_8, x_9 \in \mathbb{Z} & & & \end{array}$$

Nonconvex MINLPs

- Nonconvex sets

$$\Theta_k = \{x \in \mathbb{R}^{n+q} : x_k = \theta_k(x),\ x \in X,\ l \le x \le u,\ x_i \in \mathbb{Z},\ i \in I\}$$

- Idea: Determine convex sets $\bar{\Theta}_k \supseteq \Theta_k$ for all $k = n+1, n+2, \ldots, n+q$
- Convex relaxation

$$\begin{aligned} \min \quad & x_{n+q} \\ \text{s.t.} \quad & x_k \in \bar{\Theta}_k, \quad k = n+1, n+2, \ldots, n+q \\ & l_i \le x_i \le u_i, \quad i = 1, 2, \ldots, n+q \\ & x \in X \end{aligned}$$

- Open question: How to find $\bar{\Theta}_k$?

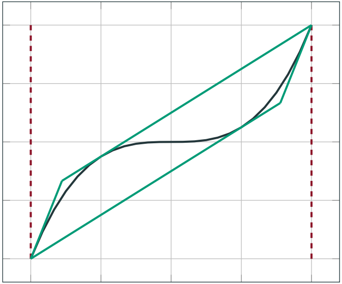
Nonconvex MINLPs

- Open question: How to find $\bar{\Theta}_k$?
- Often, the $\bar{\Theta}_k$ are polyhedral, i.e., described by linear inequalities

$$\bar{\Theta}_k = \{x \in \mathbb{R}^{n+q} \colon B^k x \geq d^k, x \in X, l \leq x \leq u\}$$

- Open question: How to find $\bar{\Theta}_k$?
- Often, the $\bar{\Theta}_k$ are polyhedral, i.e., described by linear inequalities

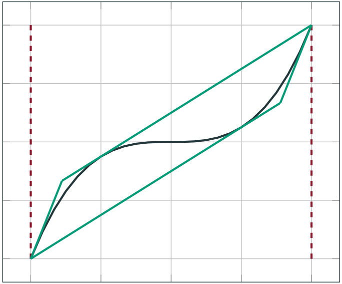
$$\bar{\Theta}_k = \{x \in \mathbb{R}^{n+q} \colon B^k x \geq d^k, x \in X, l \leq x \leq u\}$$

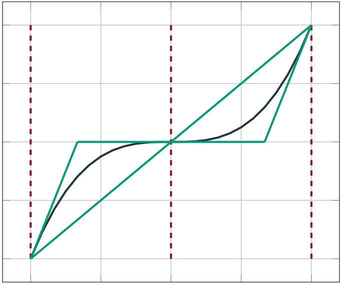


- Open question: How to find $\bar{\Theta}_k$?
- Often, the $\bar{\Theta}_k$ are polyhedral, i.e., described by linear inequalities

$$\bar{\Theta}_k = \{x \in \mathbb{R}^{n+q} \colon B^k x \geq d^k, x \in X, l \leq x \leq u\}$$

- Tightening via *spatial branching* (later more)





Definition

Let $f : \Omega \to \mathbb{R}$ be a function on the convex set $\Omega \subset \mathbb{R}^n$.

1. A function $\xi : \Omega \to \mathbb{R}$ is called a *convex underestimator* of f on Ω, if ξ is a convex function and if $\xi(x) \leq f(x)$ holds for all $x \in \Omega$. The set of all convex underestimators is denoted by $\mathcal{U}(f, \Omega)$.

Definition

Let $f : \Omega \to \mathbb{R}$ be a function on the convex set $\Omega \subset \mathbb{R}^n$.

1. A function $\xi : \Omega \to \mathbb{R}$ is called a *convex underestimator* of f on Ω, if ξ is a convex function and if $\xi(x) \leq f(x)$ holds for all $x \in \Omega$. The set of all convex underestimators is denoted by $\mathcal{U}(f, \Omega)$.
2. A function $\omega : \Omega \to \mathbb{R}$ is called a *concave overestimator* of f on Ω, if ω is a concave function and if $\omega(x) \geq f(x)$ holds for all $x \in \Omega$. The set of all concave overestimators is denoted by $\mathcal{O}(f, \Omega)$.

Definition

Let $f : \Omega \to \mathbb{R}$ be a function on the convex set $\Omega \subset \mathbb{R}^n$.

1. A function $\xi : \Omega \to \mathbb{R}$ is called a *convex underestimator* of f on Ω, if ξ is a convex function and if $\xi(x) \leq f(x)$ holds for all $x \in \Omega$. The set of all convex underestimators is denoted by $\mathcal{U}(f, \Omega)$.
2. A function $\omega : \Omega \to \mathbb{R}$ is called a *concave overestimator* of f on Ω, if ω is a concave function and if $\omega(x) \geq f(x)$ holds for all $x \in \Omega$. The set of all concave overestimators is denoted by $\mathcal{O}(f, \Omega)$.
3. The function $\text{vex}_\Omega[f]$ is defined by

$$\text{vex}_\Omega[f](x) := \sup\{\xi(x) \colon \xi \in \mathcal{U}(f, \Omega)\} \quad \text{for all } x \in \Omega$$

and is called the *convex envelope* of f.

Definition

Let $f : \Omega \to \mathbb{R}$ be a function on the convex set $\Omega \subset \mathbb{R}^n$.

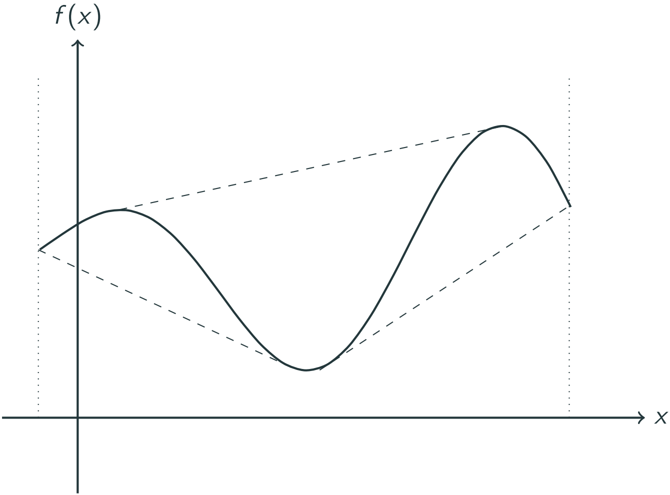
1. A function $\xi : \Omega \to \mathbb{R}$ is called a *convex underestimator* of f on Ω, if ξ is a convex function and if $\xi(x) \leq f(x)$ holds for all $x \in \Omega$. The set of all convex underestimators is denoted by $\mathcal{U}(f, \Omega)$.
2. A function $\omega : \Omega \to \mathbb{R}$ is called a *concave overestimator* of f on Ω, if ω is a concave function and if $\omega(x) \geq f(x)$ holds for all $x \in \Omega$. The set of all concave overestimators is denoted by $\mathcal{O}(f, \Omega)$.
3. The function $\text{vex}_\Omega[f]$ is defined by

   $$\text{vex}_\Omega[f](x) := \sup\{\xi(x) \colon \xi \in \mathcal{U}(f, \Omega)\} \quad \text{for all } x \in \Omega$$

   and is called the *convex envelope* of f.
4. The function $\text{cave}_\Omega[f]$ is defined by

   $$\text{cave}_\Omega[f](x) := \inf\{\omega(x) \colon \omega \in \mathcal{O}(f, \Omega)\} \quad \text{for all } x \in \Omega$$

   and is called the *concave envelope* of f.


f(x)
x

In other words

The function $\text{vex}_\Omega[f]$ minimizes the error $\|f - \xi\|_\infty$ over all functions $\xi \in \mathcal{U}(f, \Omega)$:

$$\text{vex}_\Omega[f] = \min\{\|f - \xi\|_\infty : \xi \in \mathcal{U}(f, \Omega)\}$$

In other words

The function $\text{vex}_\Omega[f]$ minimizes the error $\|f - \xi\|_\infty$ over all functions $\xi \in \mathcal{U}(f, \Omega)$:

$$\text{vex}_\Omega[f] = \min\{\|f - \xi\|_\infty : \xi \in \mathcal{U}(f, \Omega)\}$$

Reason

The pointwise supremum of convex functions is again a convex function.

In other words

The function $\text{vex}_\Omega[f]$ minimizes the error $\|f - \xi\|_\infty$ over all functions $\xi \in \mathcal{U}(f, \Omega)$:

$$\text{vex}_\Omega[f] = \min\{\|f - \xi\|_\infty : \xi \in \mathcal{U}(f, \Omega)\}$$

Reason

The pointwise supremum of convex functions is again a convex function.

In analogy for concave functions.

Theorem

Let $\Omega \subset \mathbb{R}^n$ be a compact set and let $f : \Omega \to \mathbb{R}$ be continuous. Then,

$$\min_{x \in \Omega} f(x) = \min_{x \in \operatorname{conv} \Omega} vex_{\Omega}[f](x)$$

holds.

Theorem

Let $\Omega \subset \mathbb{R}^n$ be a compact set and let $f : \Omega \to \mathbb{R}$ be continuous. Then,

$$\min_{x \in \Omega} f(x) = \min_{x \in \operatorname{conv} \Omega} vex_{\Omega}[f](x)$$

holds.

Moreover, let $\mathcal{M}$ be the set of all global minima of f over Ω and let $\mathcal{N}$ the set of global minima of $vex_{\Omega}[f]$ over $\operatorname{conv} \Omega$. Then, $\mathcal{N} = \operatorname{conv} \mathcal{M}$ holds.

The Problem with Relaxations of Nonconvexities

"*All happy families are alike; each unhappy family is unhappy in its own way.*"

— Leo Tolstoi; first sentence in Anna Karenina

The Problem with Relaxations of Nonconvexities

"*All happy families are alike; each unhappy family is unhappy in its own way.*"

— Leo Tolstoi; first sentence in Anna Karenina

All linear functions are the same . . .

The Problem with Relaxations of Nonconvexities

"*All happy families are alike; each unhappy family is unhappy in its own way.*"

— Leo Tolstoi; first sentence in Anna Karenina

All linear functions are the same . . .

. . . but all nonlinearities are different!

- Every type of nonconvexity needs to be studied separately
- Example: monomials of odd degree ($x_k = x_i^{2p+1}$, $k \in \mathbb{Z}_+$) are tackled in Liberti and Pantelides (2003)

α-Underestimator

- Based on Androulakis et al. (1995), Maranas and Floudas (1994)
- $f : \Omega \to \mathbb{R}$ twice continuously differentiable
- Domain $\Omega \subseteq \mathbb{R}^n$ is given by

$$\Omega = [\underline{x}, \bar{x}] = \prod_{i=1}^{n} [\underline{x}_i, \bar{x}_i]$$

- Define

$$\phi_\alpha(x) = \sum_{i=1}^{n} \alpha_i (\underline{x}_i - x_i)(\bar{x}_i - x_i)$$

and consider

$$\check{f}_\alpha(x) := f(x) + \phi_\alpha(x) = f(x) + \sum_{i=1}^{n} \alpha_i (\underline{x}_i - x_i)(\bar{x}_i - x_i)$$

α-Underestimator

$$\check{f}_\alpha(x) := f(x) + \phi_\alpha(x) = f(x) + \sum_{i=1}^{n} \alpha_i(\underline{x}_i - x_i)(\bar{x}_i - x_i)$$

is an underestimator if $\alpha \geq 0$ since $\phi_\alpha(x)$ is non-positive on Ω.

α-Underestimator

$$\check{f}_\alpha(x) := f(x) + \phi_\alpha(x) = f(x) + \sum_{i=1}^{n} \alpha_i(\underline{x}_i - x_i)(\bar{x}_i - x_i)$$

is an underestimator if $\alpha \geq 0$ since $\phi_\alpha(x)$ is non-positive on Ω.

But what about convexity?

Lemma

Let $\lambda^{\min}$ be the smallest eigenvalue of the Hessian matrix $H_f(x)$ of f on Ω. Then, $\check{f}_\alpha$ is convex on Ω if $\lambda^{\min} + 2\min_i \alpha_i \geq 0$ holds.

Lemma

Let $\lambda^{\min}$ be the smallest eigenvalue of the Hessian matrix $H_f(x)$ of f on Ω. Then, $\check{f}_\alpha$ is convex on Ω, if $\lambda^{\min} + 2\min_i \alpha_i \geq 0$ holds.

Proof.

Consider

$$H_{\check{f}_\alpha}(x) = H_f(x) + 2\mathrm{diag}(\alpha).$$

Lemma

Let $\lambda^{\min}$ be the smallest eigenvalue of the Hessian matrix $H_f(x)$ of f on Ω. Then, $\check{f}_\alpha$ is convex on Ω, if $\lambda^{\min} + 2\min_i \alpha_i \geq 0$ holds.

Proof.

Consider

$$H_{\check{f}_\alpha}(x) = H_f(x) + 2\mathrm{diag}(\alpha).$$

We show that it is positive semi-definite for all $x \in \Omega$.

Lemma

Let $\lambda^{\min}$ be the smallest eigenvalue of the Hessian matrix $H_f(x)$ of f on Ω. Then, $\check{f}_\alpha$ is convex on Ω, if $\lambda^{\min} + 2\min_i \alpha_i \geq 0$ holds.

Proof.

Consider

$$H_{\check{f}_\alpha}(x) = H_f(x) + 2\text{diag}(\alpha).$$

We show that it is positive semi-definite for all $x \in \Omega$.

To show that $H_{\check{f}_\alpha}(x)$ is positive-semidefinite, it suffices to show that

$$h^\top H_{\check{f}_\alpha}(x) h \geq 0 \quad \text{for all } h \in \mathbb{R}^n$$

holds.

Proof . . . continued.

We know that

$$H_{\check{f}_\alpha}(x) = H_f(x) + 2\text{diag}(\alpha)$$

holds.

Proof . . . continued.

We know that

$$H_{\check{f}_\alpha}(x) = H_f(x) + 2\text{diag}(\alpha)$$

holds.

Thus, we have

$$\begin{aligned} h^\top H_{\check{f}_\alpha}(x)h &= h^\top H_f(x)h + 2h^\top \text{diag}(\alpha)h \\ &\geq \lambda_{\min}\|h\|_2^2 + 2(\min_i \alpha_i)\|h\|_2^2 \\ &\geq 0. \end{aligned}$$

□

Back to Bilinearities: McCormick Inequalities

Mathematical Programming 10 (1976) 147–175.
North-Holland Publishing Company

COMPUTABILITY OF GLOBAL SOLUTIONS TO FACTORABLE NONCONVEX PROGRAMS: PART I – CONVEX UNDERESTIMATING PROBLEMS ★

Garth P. McCORMICK
The George Washington University, Washington, D.C., U.S.A.

Received 20 November 1973
Revised manuscript received 4 July 1975

For nonlinear programming problems which are factorable, a computable procedure for obtaining tight underestimating convex programs is presented. This is used to exclude from consideration regions where the global minimizer cannot exist.

Lemma (McCormick (1976))

Consider $w = xy$ *with* $x \in [\underline{x}, \bar{x}]$ *and* $y \in [\underline{y}, \bar{y}]$. *Then, the inequalities*

$$
\begin{aligned}
w &\geq \underline{y}x + \underline{x}y - \underline{x}\underline{y}, \\
w &\geq \bar{y}x + \bar{x}y - \bar{x}\bar{y}, \\
w &\leq \underline{y}x + \bar{x}y - \bar{x}\underline{y}, \\
w &\leq \bar{y}x + \underline{x}y - \underline{x}\bar{y}
\end{aligned}
$$

are valid inequalities.

Lemma (McCormick (1976))

Consider $w = xy$ *with* $x \in [\underline{x}, \bar{x}]$ *and* $y \in [\underline{y}, \bar{y}]$. *Then, the inequalities*

$$\begin{aligned} w &\geq \underline{y}x + \underline{x}y - \underline{x}\underline{y}, \\ w &\geq \bar{y}x + \bar{x}y - \bar{x}\bar{y}, \\ w &\leq \underline{y}x + \bar{x}y - \bar{x}\underline{y}, \\ w &\leq \bar{y}x + \underline{x}y - \underline{x}\bar{y} \end{aligned}$$

are valid inequalities.

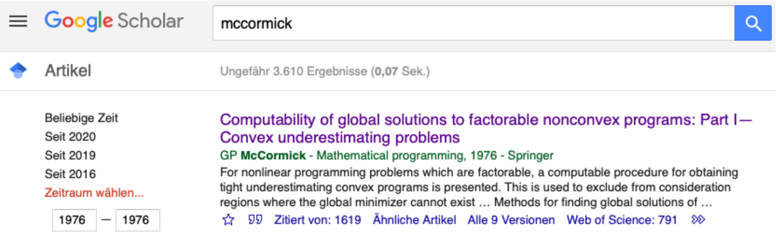
Proof.

Consider, for instance, the first and fourth inequality:

$$\begin{aligned} 0 &\leq (x - \underline{x})(y - \underline{y}) = xy - x\underline{y} - y\underline{x} + \underline{x}\underline{y}, \\ 0 &\leq (x - \underline{x})(\bar{y} - y) = x\bar{y} - xy - \underline{x}\bar{y} + \underline{x}y. \end{aligned}$$

□

Back to Bilinearities: McCormick Inequalities



... first a remainder on the linear case ...

Branch-and-Bound for (Binary) MILPs

$u \leftarrow +\infty$ and $Q \leftarrow \{(\emptyset, \emptyset)\}$.
while $Q \neq \emptyset$ **do**
  Choose $(Z, O) \in Q$ and set $Q \leftarrow Q \setminus \{(Z, O)\}$.
  Solve the Problem (3) with Z and O.
  if (3) with Z and O is infeasible **then**
    Continue.
  end if
  Let $\bar{x}$ be the optimal solution of Problem (3).
  if $c^\top \bar{x} \geq u$ **then**
    Continue.
  end if
  if $\bar{x}$ is integer-feasible **then**
    Set $x^* \leftarrow \bar{x}$, $u \leftarrow c^\top x^*$, and continue.
  end if
  Choose i with $\bar{x}_i \notin \{0, 1\}$.
  Set $Q \leftarrow Q \cup \{(Z \cup \{i\}, O), (Z, O \cup \{i\})\}$.
end while
if $u < +\infty$ **then**
  return optimal solution x^*.
else
  return "The problem is infeasible."
end if

Branch-and-Bound for (Binary) MILPs

$$\min \quad f(x) = c^\top x \quad \text{s.t.} \quad Ax = b, \quad x \geq 0, \quad x = (y, z), \quad y \in \mathbb{R}^m, \quad z \in \{0, 1\}^k$$

Branch-and-Bound for (Binary) MILPs

$$\min \quad f(x) = c^\top x \quad \text{s.t.} \quad Ax = b, \quad x \geq 0, \quad x = (y, z), \quad y \in \mathbb{R}^m, \quad z \in \{0,1\}^k$$

best integer-feasible solution: $u = \infty$

Branch-and-Bound for (Binary) MILPs

$$\min \quad f(x) = c^\top x \quad \text{s.t.} \quad Ax = b, \quad x \geq 0, \quad x = (y, z), \quad y \in \mathbb{R}^m, \quad z \in \{0, 1\}^k$$

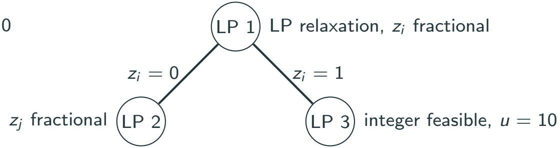
best integer-feasible solution: $u = \infty$

(LP 1) LP relaxation, z_i fractional

Branch-and-Bound for (Binary) MILPs

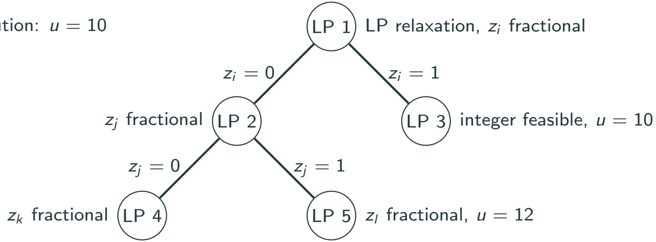
$$\min \quad f(x) = c^\top x \quad \text{s.t.} \quad Ax = b, \quad x \geq 0, \quad x = (y, z), \quad y \in \mathbb{R}^m, \quad z \in \{0, 1\}^k$$

best integer-feasible solution: $u = 10$



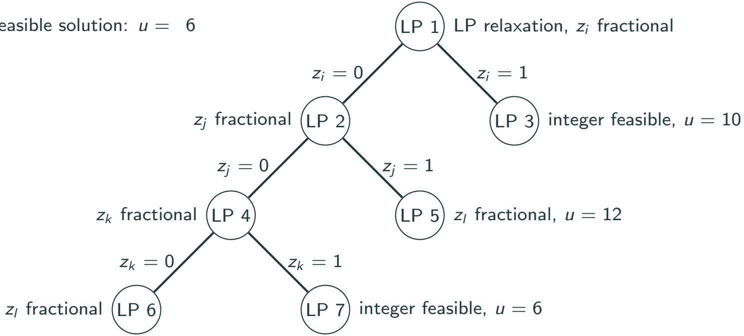
$$\min \quad f(x) = c^\top x \quad \text{s.t.} \quad Ax = b, \quad x \geq 0, \quad x = (y, z), \quad y \in \mathbb{R}^m, \quad z \in \{0,1\}^k$$

best integer-feasible solution: $u = 10$



$$\min \quad f(x) = c^\top x \quad \text{s.t.} \quad Ax = b, \quad x \geq 0, \quad x = (y, z), \quad y \in \mathbb{R}^m, \quad z \in \{0,1\}^k$$

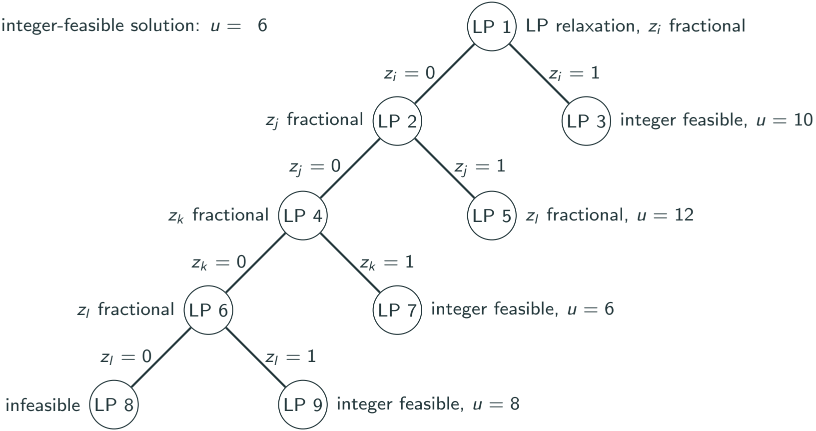
best integer-feasible solution: $u = 6$



Branch-and-Bound for (Binary) MILPs

$$\min \quad f(x) = c^\top x \quad \text{s.t.} \quad Ax = b, \quad x \geq 0, \quad x = (y, z), \quad y \in \mathbb{R}^m, \quad z \in \{0,1\}^k$$

best integer-feasible solution: $u = 6$



Branch-and-Bound for Nonconvex MINLP

The original MINLP

$$
\begin{aligned}
\min_{x \in \mathbb{R}^n} \quad & f(x) \\
\text{s.t.} \quad & c(x) \leq 0 \\
& x \in X \\
& x_i \in \mathbb{Z}, \quad i \in I
\end{aligned}
$$

Subproblems (= nodes of the branch-and-bound tree) are specified by additionally imposed bounds

Required (as before):

1. procedure to compute lower bounds on the optimal objective function value of the subproblem
2. procedure for partitioning the feasible set of a subproblem

The Subproblem

The original MINLP plus additional bounds

$$\begin{aligned}
\min_{x \in \mathbb{R}^n} \quad & f(x) \\
\text{s.t.} \quad & c(x) \leq 0 \\
& x \in X \\
& l_i \leq x_i \leq u_i, \quad i = 1, \ldots, n \\
& x_i \in \mathbb{Z}, \quad i \in I
\end{aligned} \qquad (\text{MINLP}(l, u))$$

The Subproblem

The original MINLP plus additional bounds

$$\begin{aligned}
\min_{x \in \mathbb{R}^n} \quad & f(x) \\
\text{s.t.} \quad & c(x) \leq 0 \\
& x \in X \\
& l_i \leq x_i \leq u_i, \quad i = 1, \ldots, n \\
& x_i \in \mathbb{Z}, \quad i \in I
\end{aligned} \qquad (\text{MINLP}(l, u))$$

Goals

- Obtain a lower bound of the optimal value of $f(x)$
- Solve a convex relaxation or (even) a polyhedral relaxation

Polyhedral Relaxation of the Subproblem

Consider the polyhedral (and thus convex) relaxation

$$\begin{aligned} \min_{x \in \mathbb{R}^{n+q}} \quad & x_{n+q} \\ \text{s.t.} \quad & B^k x \geq d^k, \quad k = n+1, n+2, \ldots, n+q \\ & x \in X \\ & l_i \leq x_i \leq u_i, \quad i = 1, \ldots, n+q \end{aligned} \tag{LP(l, u)}$$

of MINLP(l, u) and let $\hat{x}$ be an optimal solution.

After Solving the Polyhedral Relaxation of the Subproblem

Two Situations

1. $\hat{x}$ is feasible for the MINLP(l, u)
    - Thus, $\hat{x}$ is also feasible for the original MINLP
    - The subproblem can be eliminated (i.e., the node can be pruned)

After Solving the Polyhedral Relaxation of the Subproblem

Two Situations

1. $\hat{x}$ is feasible for the MINLP(l, u)
   - Thus, $\hat{x}$ is also feasible for the original MINLP
   - The subproblem can be eliminated (i.e., the node can be pruned)
2. $\hat{x}$ is infeasible for the MINLP(l, u)

After Solving the Polyhedral Relaxation of the Subproblem

Two Situations

1. $\hat{x}$ is feasible for the MINLP(l, u)
   - Thus, $\hat{x}$ is also feasible for the original MINLP
   - The subproblem can be eliminated (i.e., the node can be pruned)
2. $\hat{x}$ is infeasible for the MINLP(l, u)
   (a) There is an index $i \in I$ with $x_i \notin \mathbb{Z}$ (i.e., the point is not integer feasible)
   → Branching on integer variables (as usual): Create new subproblems MINLP(l^-, u^-) and MINLP(l^+, u^+) by imposing the additional constraints $x_i \leq \lfloor \hat{x}_i \rfloor$ and $\lceil \hat{x}_i \rceil \leq x_i$, respectively

After Solving the Polyhedral Relaxation of the Subproblem

Two Situations

1. $\hat{x}$ is feasible for the MINLP(l, u)
   - Thus, $\hat{x}$ is also feasible for the original MINLP
   - The subproblem can be eliminated (i.e., the node can be pruned)
2. $\hat{x}$ is infeasible for the MINLP(l, u)
   (a) There is an index $i \in I$ with $x_i \notin \mathbb{Z}$ (i.e., the point is not integer feasible)
   → Branching on integer variables (as usual): Create new subproblems MINLP(l^-, u^-) and MINLP(l^+, u^+) by imposing the additional constraints $x_i \leq \lfloor \hat{x}_i \rfloor$ and $\lceil \hat{x}_i \rceil \leq x_i$, respectively
   (b) There is an index $k \in \{n+1, n+2, \ldots, n+q\}$ with $\hat{x}_k \neq \theta_k(\hat{x})$
   → Branching on a continuous variable (spatial branching)

Spatial Branching

- Suppose x_i is one of the arguments of θ_k
- Branching example:

$$x_i \leq \hat{x}_i \quad \vee \quad \hat{x}_i \leq x_i$$

- Note: the feasible sets of the two new subproblems have non-empty intersection
  - This is different to branching on integer variables

Spatial Branching

- Suppose x_i is one of the arguments of θ_k
- Branching example:

$$x_i \leq \hat{x}_i \quad \vee \quad \hat{x}_i \leq x_i$$

- Note: the feasible sets of the two new subproblems have non-empty intersection
  - This is different to branching on integer variables

Further Difference

For

- purely integer nonconvex MINLPs and
- convex MINLPs

only branching on integer variables is required.

Spatial Branching

- Suppose x_i is one of the arguments of θ_k
- Branching example:

$$x_i \leq \hat{x}_i \quad \vee \quad \hat{x}_i \leq x_i$$

- Note: the feasible sets of the two new subproblems have non-empty intersection
  - This is different to branching on integer variables

Further Difference

For

- purely integer nonconvex MINLPs and
- convex MINLPs

only branching on integer variables is required.

In These Cases:

Finite bounds on integers variables ensures finite termination of the algorithm

Bounding Operations

Let

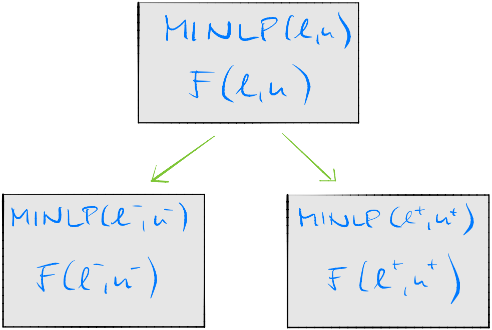
$$\Omega(l, u) = \{x \in [l, u] \colon c(x) \leq 0,\, x \in X,\, x_i \in \mathbb{Z} \text{ for all } i \in I\}$$

be the feasible set of MINLP(l, u).

Let

$$\Omega(l,u) = \{x \in [l,u] \colon c(x) \leq 0,\ x \in X,\ x_i \in \mathbb{Z} \text{ for all } i \in I\}$$

be the feasible set of MINLP(l,u).



Definition

A *bounding operation* yields

1. two subproblems

$$\mathrm{MINLP}(l^-, u^-), \quad \mathrm{MINLP}(l^+, u^+)$$

   by applying a branching rule and
2. lower bounds

$$\lambda_{\Omega(l^-,u^-)}, \quad \lambda_{\Omega(l^+,u^+)}$$

   as well as upper bounds

$$\mu_{\Omega(l^-,u^-)}, \quad \mu_{\Omega(l^+,u^+)}$$

   for the new subproblems

Definition

A bounding operation is called *consistent* if, at every step, the subsets

$$\Omega(l^-, u^-), \quad \Omega(l^+, u^+)$$

are either pruned or can be further refined in such a way that, for any finite sequence $(\Omega_h)_h$ resulting from applying bounding operations, one can guarantee that

$$\lim_{h \to \infty} \mu_{\Omega_h} - \lambda_{\Omega_h} = 0$$

holds.

Finite Consistence and Finite Termination

Definition

In addition, a bounding operation is called *finitely consistent* if any sequence $(\Omega_h)_h$ of successively refined partitions of Ω is finite.

Definition

In addition, a bounding operation is called *finitely consistent* if any sequence $(\Omega_h)_h$ of successively refined partitions of Ω is finite.

Theorem (McCormick 1976, Horst & Tuy 1993)

If the bounding operation in the branch-and-bound algorithm is finitely consistent, then the algorithm terminates after a finite number of steps.

How to do spatial branching?

- Branching means partitioning the feasible set of subproblem MINLP(l, u) into $h \geq 2$ feasible sets of the subproblems MINLP($l^{(1)}, u^{(1)}$), $\dots$, MINLP($l^{(h)}, u^{(h)}$)
- The lower bounds $\lambda_{\Omega(l^{(1)},u^{(1)})}$, $\lambda_{\Omega(l^{(2)},u^{(2)})}$, $\dots$, $\lambda_{\Omega(l^{(h)},u^{(h)})}$ should be no smaller than the lower bound for MINLP(l, u).
- For the ease of presentation: two new subproblems MINLP(l^{-}, u^{-}) and MINLP(l^{+}, u^{+}) are created

How to do spatial branching?

- Branching means partitioning the feasible set of subproblem MINLP(l, u) into $h \geq 2$ feasible sets of the subproblems MINLP($l^{(1)}, u^{(1)}$), ..., MINLP($l^{(h)}, u^{(h)}$)
- The lower bounds $\lambda_{\Omega(l^{(1)},u^{(1)})}$, $\lambda_{\Omega(l^{(2)},u^{(2)})}$, ..., $\lambda_{\Omega(l^{(h)},u^{(h)})}$ should be no smaller than the lower bound for MINLP(l, u).
- For the ease of presentation: two new subproblems MINLP(l^-, u^-) and MINLP(l^+, u^+) are created
- Most implementations use *variable branching*:

$$x_i \leq b \quad \vee \quad x_i \geq b$$

How to do spatial branching?

- Branching means partitioning the feasible set of subproblem MINLP(l, u) into $h \geq 2$ feasible sets of the subproblems MINLP($l^{(1)}, u^{(1)}$), ..., MINLP($l^{(h)}, u^{(h)}$)
- The lower bounds $\lambda_{\Omega(l^{(1)},u^{(1)})}$, $\lambda_{\Omega(l^{(2)},u^{(2)})}$, ..., $\lambda_{\Omega(l^{(h)},u^{(h)})}$ should be no smaller than the lower bound for MINLP(l, u).
- For the ease of presentation: two new subproblems MINLP(l^-, u^-) and MINLP(l^+, u^+) are created
- Most implementations use *variable branching*:

$$x_i \leq b \quad \vee \quad x_i \geq b$$

- But how?
- The performance of the overall method strongly depends on the choice of i and b

How to do spatial branching?

- A fractional integer variable is an obvious candidate for branching
- Suppose that all integer variables are already integer-valued so that we "only" need to branch on continuous variables in the following
- Thus, branching is done because of a variable x_k with $\hat{x}_k \neq \theta_k(\hat{x})$

Nice-to-haves

An ideal choice of i should

- increase the lower bounds $\lambda_{\Omega(l^-,u^-)}$ and $\lambda_{\Omega(l^+,u^+)}$,
- reduce the feasible sets $\Omega(l^-, u^-)$ and $\Omega(l^+, u^+)$,
- ...

How to do spatial branching?

Let $\hat{x}$ be a solution of a relaxation of MINLP(l, u).

A continuous variable x_i is a *branching candidate* if

- it is not fixed (i.e., its lower and upper bound do not coincide)
- it is an argument of a function θ_k with $\hat{x}_k \neq \theta_k(\hat{x})$

How to do spatial branching?

Let $\hat{x}$ be a solution of a relaxation of MINLP(l, u).

A continuous variable x_i is a *branching candidate* if

- it is not fixed (i.e., its lower and upper bound do not coincide)
- it is an argument of a function θ_k with $\hat{x}_k \neq \theta_k(\hat{x})$

Example

If $x_k = \theta_k(x) = x_i x_j$, $\hat{x}_k \neq \hat{x}_i \hat{x}_j$, and $l_i < u_i$, then x_i is a branching candidate.

How to do spatial branching?

Let $\hat{x}$ be a solution of a relaxation of MINLP(l, u).

A continuous variable x_i is a *branching candidate* if

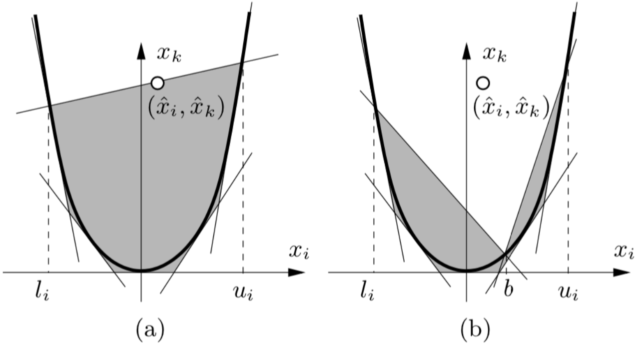
- it is not fixed (i.e., its lower and upper bound do not coincide)
- it is an argument of a function θ_k with $\hat{x}_k \neq \theta_k(\hat{x})$

Example

If $x_k = \theta_k(x) = x_i x_j$, $\hat{x}_k \neq \hat{x}_i \hat{x}_j$, and $l_i < u_i$, then x_i is a branching candidate.

After branching, the two generated subproblems both will have a lower bound no smaller than the one of their ancestor node since branching leads to tighter relaxations.

Tightened Polyhedral Relaxations after Branching



pictures taken from Belotti et al. (2013)

Choosing the Branching Point

- The choice of the branching point is crucial
- The degree of freedom differs from branch-and-bound for mixed-integer linear optimization
  - This means, the branching point may differ from $\hat{x}_i$
- The branching rule should ensure that $\hat{x}$ is infeasible for both MINLP(l^-, u^-) and MINLP(l^+, u^+)
  - Thus, $x_i \leq \hat{x}_i \vee x_i \geq \hat{x}_i$ will not suffice in general

Bound Tightening

- The performance of nonconvex MINLP solvers crucially depends on the tightness of the convex relaxations
- The tightness of the convex relaxations strongly depends on the variable bounds
- MINLP solvers spend a lot of effort in *bound tightening*

Bound Tightening

Let

$$\Omega = \{x \in [l, u] \colon c(x) \le 0,\ x \in X,\ x_i \in \mathbb{Z} \text{ for } i \in I\}$$

be the feasible set and let $\hat{x}$ be a feasible point with objective function value $\hat{z}$.

Let

$$\Omega = \{x \in [l, u] \colon c(x) \leq 0,\ x \in X,\ x_i \in \mathbb{Z} \text{ for } i \in I\}$$

be the feasible set and let $\hat{x}$ be a feasible point with objective function value $\hat{z}$.

We could solve the $2n$ problems

$$l_i' = \min \{x_i \colon x \in \Omega,\ f(x) \leq \hat{z}\}$$

and

$$u_i' = \max \{x_i \colon x \in \Omega,\ f(x) \leq \hat{z}\}$$

Bound Tightening

Let

$$\Omega = \{x \in [l, u] \colon c(x) \leq 0,\ x \in X,\ x_i \in \mathbb{Z} \text{ for } i \in I\}$$

be the feasible set and let $\hat{x}$ be a feasible point with objective function value $\hat{z}$.

We could solve the $2n$ problems

$$l_i' = \min \{x_i \colon x \in \Omega,\ f(x) \leq \hat{z}\}$$

and

$$u_i' = \max \{x_i \colon x \in \Omega,\ f(x) \leq \hat{z}\}$$

Problem

These $2n$ problems can be as hard as the original problem!

Feasibility-Based Bound Tightening (FBBT)

Idea

Infer a tighter bound on a variable x_i because a bound on another variable x_j has changed

Feasibility-Based Bound Tightening (FBBT)

Idea

Infer a tighter bound on a variable x_i because a bound on another variable x_j has changed

Example

Consider $x_j = x_i^3$ and $x_i \in [l_i, u_i]$.

Then, bounds on x_j can be tightened to $[l_j, u_j] \cap [l_i^3, u_i^3]$.

Affine-Linear Functions

Consider the constraint

$$x_k = a_o + \sum_{j=1}^{n} a_j x_j \quad \text{with} \quad k > n.$$

Affine-Linear Functions

Consider the constraint

$$x_k = a_o + \sum_{j=1}^{n} a_j x_j \quad \text{with} \quad k > n.$$

Define

$$J^+ = \{j \in \{1, \dots, n\} \colon a_j > 0\}, \quad J^- = \{j \in \{1, \dots, n\} \colon a_j < 0\}.$$

Then, valid bounds for x_k are given by

$$a_0 + \sum_{j \in J^-} a_j u_j + \sum_{j \in J^+} a_j l_j \le x_k \le a_0 + \sum_{j \in J^-} a_j l_j + \sum_{j \in J^+} a_j u_j$$

Affine-Linear Functions

Consider the constraint

$$x_k = a_o + \sum_{j=1}^{n} a_j x_j \quad \text{with} \quad k > n.$$

Define

$$J^+ = \{j \in \{1, \ldots, n\} \colon a_j > 0\}, \quad J^- = \{j \in \{1, \ldots, n\} \colon a_j < 0\}.$$

Then, valid bounds for x_k are given by

$$a_0 + \sum_{j \in J^-} a_j u_j + \sum_{j \in J^+} a_j l_j \le x_k \le a_0 + \sum_{j \in J^-} a_j l_j + \sum_{j \in J^+} a_j u_j$$

These bounds can then be used to derive tighter bounds for x_j, $j = 1, \ldots, n$.

Feasibility-Based Bound Tightening (FBBT)

- For nonlinear problems we can apply bound propagation by using the corresponding DAG
- Leads to an iterative procedure that is/can be continued as long as variable bounds change
- Does *not* need to converge

Optimality-Based Bound Tightening (OBBT)

The problems

$$l_i' = \min\{x_i\colon x \in \Omega,\ f(x) \leq \hat{z}\}$$

and

$$u_i' = \max\{x_i\colon x \in \Omega,\ f(x) \leq \hat{z}\}$$

are usually to hard to be solved for bound tightening.

Optimality-Based Bound Tightening (OBBT)

The problems

$$l_i' = \min\{x_i : x \in \Omega,\ f(x) \leq \hat{z}\}$$

and

$$u_i' = \max\{x_i : x \in \Omega,\ f(x) \leq \hat{z}\}$$

are usually to hard to be solved for bound tightening.

- In practice, one often uses polyhedral relaxations for the feasible sets instead.
- This leads to valid bounds as well ...
- ... but still is expensive.

Modeling Languages

- Modeling languages allow to state optimization problems
- They provide interfaces to solvers that solve the stated problem

Modeling Languages

- Modeling languages allow to state optimization problems
- They provide interfaces to solvers that solve the stated problem

The Two Classics

- AMPL: A Mathematical Programming Language (`http://www.ampl.com`)
- GAMS: General Algebraic Modeling System (`https://www.gams.com`)

Example: Gurobi via Python

Code Example

The Knapsack problem coded in Python and solved with Gurobi

"Pyomo is a Python package that supports the formulation and analysis of mathematical models for complex optimization applications. This capability is commonly associated with commercially available algebraic modeling languages (AMLs) such as AMPL, AIMMS, and GAMS."

Installation and Usage

If you are using Anaconda:

- `conda install -c conda-forge pyomo`
- `conda install -c conda-forge ipopt coincbc glpk`

If you are not using Anaconda:

- `pip3 install pyomo`

Usage:

- `import pyomo.environ as *`
- `from pyomo.opt import SolverFactory`

$$\min_{x} \quad f_0(x)$$
$$\text{s.t.} \quad f_i(x) \leq b_i, \; i \in I.$$

What do we need?

1. Model
2. Sets
3. Parameters
4. Variables
5. Objective
6. Constraints
7. Interaction with solvers

Concrete Model vs. Abstract Model

Abstract Model

$$\begin{aligned} \min_x \quad & \sum_{j=1}^{n} c_j x_j \\ \text{s.t.} \quad & \sum_{j=1}^{n} a_{ij} x_j \geq b_i, \ \ i \in I \\ & x_j \geq 0, \, j \in J \end{aligned}$$

```
m = AbstractModel()
```

Concrete Model

$$\begin{aligned} \min_x \quad & 2x_1 + 3x_2 \\ \text{s.t.} \quad & 3x_1 + 4x_2 \geq 1 \\ & x_1, x_2 \geq 0 \end{aligned}$$

```
m = ConcreteModel()
```

Initialization:

```
m.I = Set()
```

Useful arguments:

- dimen: Dimension of the members of the set.
- initialize: An iterable containing the initial members of the set, or function that returns an iterable of the initial members the set.

Example:

```
m.I = Set(dimen=2,initialize=[(1,1),(1,2)])
```

Operations:

- `m.I = m.A | m.D # union`
- `m.J = m.A & m.D # intersection`
- `m.K = m.A - m.D # difference`
- `m.L = m.A ^ m.D # exclusive-or`

Parameters

Initialization:

```
m.A = Set()
m.B = Set()
m.P = Param(m.A, m.B)
```

Useful arguments:

- default: The default value if no other specification is available.

Example:

```
m.S = Param(m.A, m.A, default=0)
```

Initialization:

```
m.A = Set()
m.B = Set()
m.x = Var(m.A, m.B)
```

Useful arguments:

- bounds: A function (or Python object) that gives a (lower, upper) bound pair for the variable
- domain: A set that is a super-set of the values the variable can take on.

Example:

```
m.x = Var(m.A, domain=PositiveIntegers, bounds=(0,6))
```

Objective and Constraints

Initialization of the Objective:

```
def ObjRule(m):
    return sum(m.x[a] for a in m.A) + m.y
m.Obj = Objective(rule=ObjRule, sense=maximize)
```

Initialization of a typical constraint:

```
def Cons1_rule(m, a):
    return m.P[a,a]*m.x[a] <= a
m.Cons1 = Constraint(m.A, rule=Cons1_rule)
```

Instantiate models using dictionaries

Example:

```
m.I = Set()
m.p = Param()
m.q = Param(m.I)
m.r = Param(m.I, m.I, default=0)
data = {None: {
'I': {None: [1, 2, 3]},
'p': {None: 100},
'q': {1:10, 2:20, 3:30},
'r': {(1,1):110, (1,2):120, (2,3):230}}}
i = m.create_instance(data)
```

Example:

```
i = m.create_instance(data)
opt = SolverFactory('ipopt')
opt.solve(i)
```

Useful arguments of the solve method:

- tee: Boolean argument, which controls if the solver output is printed.
- warmstart: Boolean argument, which controls if the solver is warm started using the values given in the variables.
- timelimit: Time in seconds after which the solver is told to stop computing and to return the best solution found.

Using Pyomo with GAMS

Pyomo instance → GAMS → Solver → GAMS → Pyomo Instance

Example:

```
i = m.create_instance(data)
opt = SolverFactory('gams')
opt.solve(i)
```

Useful arguments of the solve method:

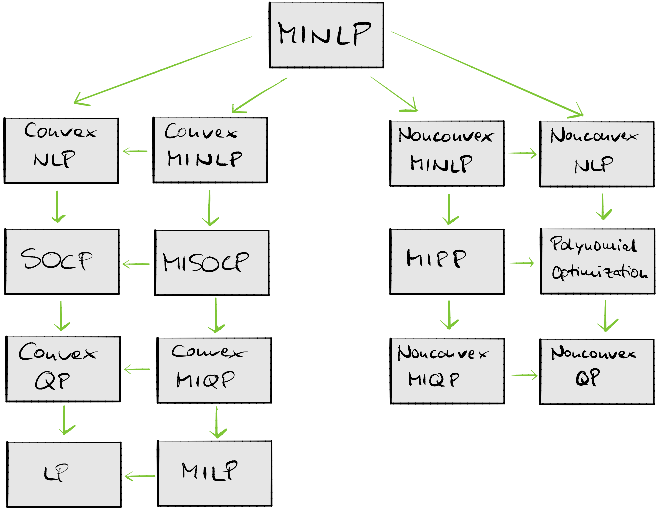
- tee: Boolean argument, which controls if the solver output is printed.
- solver: Solver used for the computation.
- warmstart: Boolean argument, which controls if the solver is warm started using the values given in the variables.
- add_options: List of additional lines to write directly into model file before the solve statement.
- mtype: Model type.

Warmstarting and Retrieving Variable Values

Example:

```
m.a = Var()
i = m.create_instance(data)
i.a.value = 2
opt = SolverFactory('gams')
opt.solve(i, warmstart=True)
value_a_after_computation = i.a.value
```

6. Solvers


MINLP
Convex NLP
Convex MINLP
Nonconvex MINLP
Nonconvex NLP
SOCP
MISOCP
MIPP
Polynomial Optimization
Convex QP
Convex MIQP
Nonconvex MIQP
Nonconvex QP
LP
MILP

Solvers for Nonconvex MINLP

- ANTIGONE
- BARON
- Couenne (open-source)
- LINDOGlobal
- SCIP (open-source)

Solvers for Convex MINLP

- α-ECP
- Bonmin (open-source)
- DICOPT
- FilMINT
- KNITRO
- MINLP-BB
- MINOTAUR (open-source)
- SBB

Solvers for MIQP

$$\min_{x \in \mathbb{R}^n} \quad x^\top Q x + c^\top x \quad \text{s.t.} \quad Ax = b, \quad x \geq 0$$

Convex MIQP

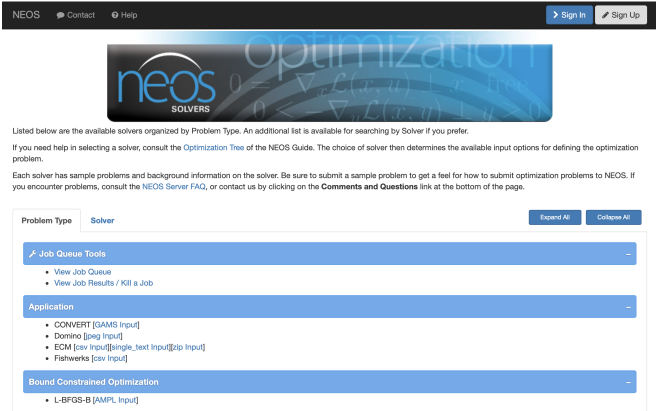
- CPLEX
- GUROBI
- MOSEK
- XPRESS

Nonconvex MIQP

- GLOMIQO

The NEOS Server

http://www.neos-server.org/neos/solvers/index.html



7. What Else?

Mixed-Integer Quadratic Problems (MIQPs)

Minimization of quadratic objective over a mixed-integer polyhedral feasible set

Mixed-Integer Quadratic Problems (MIQPs)

Minimization of quadratic objective over a mixed-integer polyhedral feasible set

Mixed-Integer Second-Order Cone Problems (MISOCPs)

Includes constraints of the form

$$\|Ax + b\|_2 - p^\top x + q \leq 0$$

Mixed-Integer Quadratic Problems (MIQPs)

Minimization of quadratic objective over a mixed-integer polyhedral feasible set

Mixed-Integer Second-Order Cone Problems (MISOCPs)

Includes constraints of the form

$$\|Ax + b\|_2 - p^\top x + q \leq 0$$

Mixed-Integer Polynomial Problems (MIPPs)

Objective function and constraints may be general polynomials

Algorithms

- Generalized Benders Decomposition (GBD)
- Extended Cutting Plane (ECP) method
- Presolve techniques
- Bound tightening
- Primal heuristics

Valid Inequalities

- Disjunctive/split cuts
- Perspective cuts
- Chvátal–Gomory rounding and mixed-integer rounding cuts for conic MINLP
- Intersection cuts
- Reformulation-Linearization Technique (RLT)
- Cut generating functions

Problem Classes

Mixed-integer . . .

- optimal control problems
- stochastic problems
- robust problems
- problems with black-box functions
- bilevel problems (leader-follower games)

8. Literature

- Ioannis Androulakis, Costas Maranas, and C. Floudas. “αBB: A Global Optimization Method for General Constrained Nonconvex Problems.” In: *Journal of Global Optimization* 7 (Dec. 1995), pp. 337–363. DOI: 10.1007/BF01099647.
- Pietro Belotti et al. “Mixed-integer nonlinear optimization.” In: *Acta Numerica* 22 (2013), pp. 1–131. DOI: 10.1017/S0962492913000032.
- Marco A. Duran and Ignacio E. Grossmann. “An outer-approximation algorithm for a class of mixed-integer nonlinear programs.” In: *Mathematical Programming* 36.3 (1986), pp. 307–339. DOI: 10.1007/BF02592064.
- Roger Fletcher and Sven Leyffer. “Solving mixed integer nonlinear programs by outer approximation.” In: *Mathematical Programming* 66.1 (1994), pp. 327–349. ISSN: 1436-4646. DOI: 10.1007/BF01581153.
- William E. Hart et al. *Pyomo-optimization modeling in python*. Vol. 67. Springer, 2017.

- Reiner Horst and Hoang Tuy. *Global Optimization*. Springer, 1993. DOI: 10.1007/978-3-662-02598-7.
- James E. Kelley Jr. "The Cutting-Plane Method for Solving Convex Programs." In: *Journal of the Society for Industrial and Applied Mathematics* 8.4 (1960), pp. 703–712. DOI: 10.1137/0108053.
- A. H. Land and A. G. Doig. "An Automatic Method of Solving Discrete Programming Problems." In: *Econometrica* 28.3 (1960), pp. 497–520. ISSN: 00129682, 14680262. URL: http://www.jstor.org/stable/1910129.
- Leo Liberti and Constantinos C. Pantelides. "Convex Envelopes of Monomials of Odd Degree." In: *Journal of Global Optimization* 25.2 (2003), pp. 157–168. ISSN: 1573-2916. DOI: 10.1023/A:1021924706467.

Garth P. McCormick. “Computability of global solutions to factorable nonconvex programs: Part I — Convex underestimating problems.” In: *Mathematical Programming* 10.1 (1976), pp. 147–175. ISSN: 1436-4646. DOI: 10.1007/BF01580665.

Pyomo Read the Docs. https://pyomo.readthedocs.io/en/stable/index.html. Accessed: 2019-01-06.

Ignacio Quesada and Ignacio E. Grossmann. “An LP/NLP based branch and bound algorithm for convex MINLP optimization problems.” In: *Computers & Chemical Engineering* 16.10-11 (1992), pp. 937–947. DOI: 10.1016/0098-1354(92)80028-8.


"That's all Folks!"