

Introduction
ooooooooo

Finding Feasible Solutions
ooo

Proving Optimality
oooooooooooo

Strategy
oooooo

MINLP in SCIP
oooo

Practical Topics
ooo

Mixed-Integer Nonlinear Programming

Ksenia Bestuzheva

Zuse Institute Berlin

CO@Work 2020 · September 17, 2020

Introduction
●ooooooooo

Finding Feasible Solutions
ooo

Proving Optimality
oooooooooooo

Strategy
oooooo

MINLP in SCIP
oooo

Practical Topics
ooo

Introduction

Introduction

- About this lecture
 - What is mixed-integer nonlinear programming
 - Solving a mixed-integer optimisation problem
 - What is special about nonlinear problems

About This Lecture

Goals of the lecture:

- Introduce the viewers to the key concepts of mixed-integer nonlinear programming
- Explain the basics of MINLP solution methods
- Share some practical tips

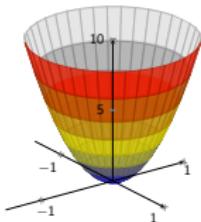
It is assumed that the viewers are familiar with the following:

- Basic notions of optimisation: optimisation problem, feasible set, objective function, feasible and optimal solutions
- Basic notions of mixed-integer linear programming: mixed-integer linear program, integer variables, continuous relaxation
- MILP branch-and-bound: branching and bounding, primal and dual bounds, optimality gap, pruning, cutting planes

Mixed-Integer Nonlinear Programs

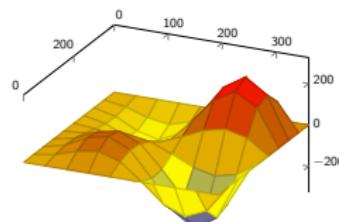
$$\begin{aligned} & \min c^T x \\ \text{s.t. } & g_k(x) \leq 0 \quad \forall k \in [m] \\ & x_i \in [\ell_i, u_i] \quad \forall i \in [n] \\ & x_i \in \mathbb{Z} \quad \forall i \in \mathcal{I} \subseteq [n] \end{aligned}$$

The nonlinear part: functions $g_k \in C^1([\ell, u], \mathbb{R})$:



convex

or



nonconvex

Examples of Nonlinearities

- Variable **fraction** $p \in [0, 1]$ of variable quantity q : qp . Example: **water treatment unit**

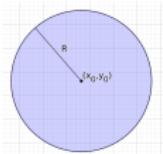


- **AC power flow** - nonlinear function of voltage magnitudes and angles



$$p_{ij} = g_{ij} v_i^2 - g_{ij} v_i v_j \cos(\theta_{ij}) + b_{ij} v_i v_j \sin(\theta_{ij})$$

- **Distance constraints**



$$(x - x_0)^2 + (y - y_0)^2 \leq R$$

- etc.

Solving a Mixed-Integer Optimisation Problem

Two major tasks:

1. Finding and improving feasible solutions (**primal side**)
 - Ensure feasibility, sacrifice optimality
 - Important for practical applications
2. Proving optimality (**dual side**)
 - Ensure optimality, sacrifice feasibility
 - Necessary in order to actually solve the problem

Connected by:

3. Strategy
 - Ensure convergence
 - Divide: branching, decompositions, ...
 - Put together all components

Nonlinearity Brings New Challenges

- More numerical issues
- NLP solvers are less efficient and reliable than LP solvers

1. Finding feasible solutions

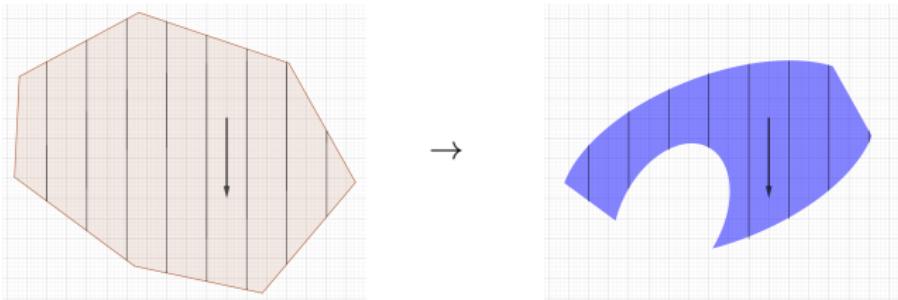
- Feasible solutions must also satisfy nonlinear constraints
- If nonconvex: fixing integer variables and solving the NLP can produce local optima

2. Proving optimality

- NLP or LP relaxations?
- If nonconvex: continuous relaxation no longer provides a lower bound
- "Convenient" descriptions of the feasible set are important

3. Strategy

- Need to account for all of the above
- Warmstart for NLP is much less efficient than for LP



Introduction: Recap

- What is an **MINLP problem**? What do constraints, variables, objective look like?
- **Solving an MINLP** can be roughly divided into **two major tasks**. What are they and how are they **connected**?
- **Adding "nonlinear"** to "mixed-integer" makes the problem even more difficult. How does this **affect different parts** of the solution process?

Introduction
ooooooooo

Finding Feasible Solutions
●oo

Proving Optimality
oooooooooooo

Strategy
oooooo

MINLP in SCIP
oooo

Practical Topics
ooo

Finding Feasible Solutions

Primal Heuristics

The goal of primal heuristics is to find solutions that are:

- feasible (satisfying all constraints) and
- good quality (solutions with lower objective value are preferable).

The best of solutions found so far is referred to as **best feasible** or **incumbent**. It provides an **upper bound** on the optimal value.

Common theme in primal heuristics: restrict the problem to obtain a subproblem for which a **feasible** solution can be found.

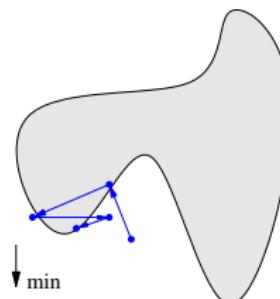
Nonconvex: NLP subproblems are usually solved to local optimality.

- Local optima are still feasible solutions
- Not finding the global optimum affects the quality of upper bounds

Primal Heuristics for MINLPs

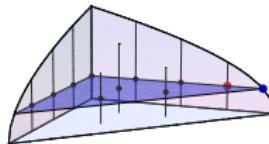
NLP local search

- Fix **integer variables** to values at reference point; solve the NLP.
- Reference point: integer feasible solution of the LP relaxation.



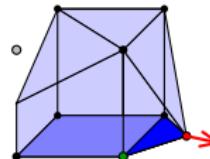
Undercover

- Fix **some** variables so that the subproblem is **linear** and solve the MIP.



Sub-MINLP

- Search around **promising** solutions.
- The region is restricted by additional constraints and/or fixing variables.



Introduction
ooooooooo

Finding Feasible Solutions
ooo

Proving Optimality
●oooooooooooo

Strategy
oooooo

MINLP in SCIP
oooo

Practical Topics
ooo

Proving Optimality

Proving Optimality

- Using relaxations for finding lower bounds
- Relaxations for convex MINLPs
- Managing cuts: initial cuts and dynamically added cuts
- Relaxations for nonconvex MINLPs
- How to strengthen the relaxations

Finding Lower Bounds: Relaxations

Key task: describe the **feasible set** in a **convenient** way.

Requirement: the relaxed problem should be **efficiently** solvable to **global** optimality.

It is **preferable** to have relaxations that are:

- **Convex**: NLP solutions are globally optimal, infeasibility detection is reliable
- **Linear**: solving is more efficient, good for warmstarting

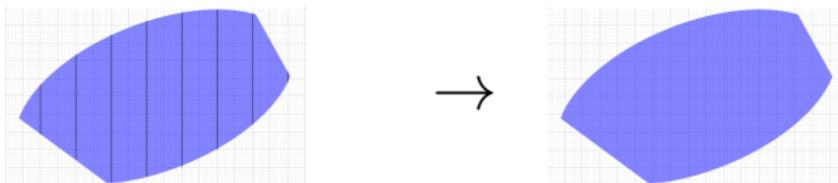
and to **avoid**:

- Very large numbers of constraints and variables
- Bad numerics

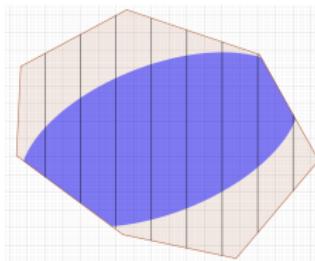
Let F be the **feasible set**. We look for a **relaxation**: a set R such that $F \subseteq R$ which satisfies some of the above.

Relaxations for Convex MINLPs

- Relax integrality



- Replace the nonlinear set with a linear outer approximation



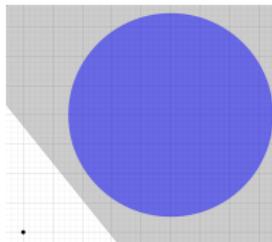
- Linear outer approximation + relax integrality → LP relaxation

Outer Approximating Convex Constraints

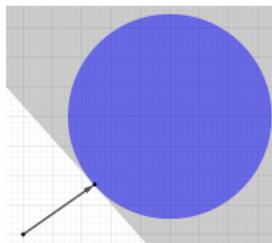
A linear inequality $ax \leq b$ is valid if $x \in F \Rightarrow ax \leq b$ (such inequalities are called cutting planes, or cuts)

Given constraint $g(x) \leq 0$ (g convex, differentiable) and a reference point \hat{x} , one can build:

Gradient cuts (Kelley):
$$g(\hat{x}) + \nabla g(\hat{x})(x - \hat{x}) \leq 0$$

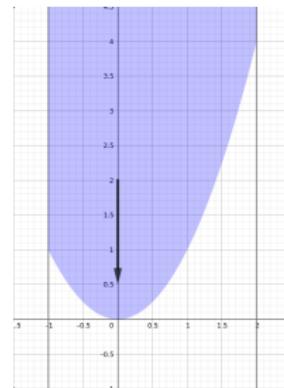


Projected cuts: same, but move \hat{x} to the boundary of F



Which Cuts to Add?

There are infinitely many possible cuts, how to choose them?

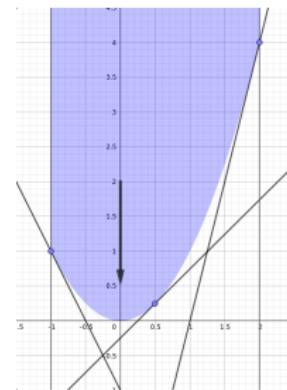


Which Cuts to Add?

There are infinitely many possible cuts, how to choose them?

Initial cuts

- Added **before** the first LP relaxation is solved
- Reference points chosen based on feasible set only

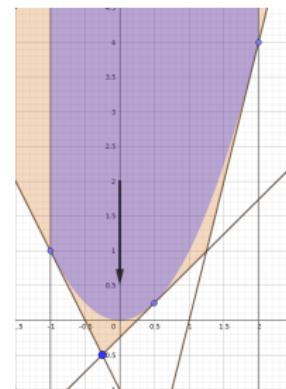


Which Cuts to Add?

There are infinitely many possible cuts, how to choose them?

Initial cuts

- Added **before** the first LP relaxation is solved
- Reference points chosen based on feasible set only



Which Cuts to Add?

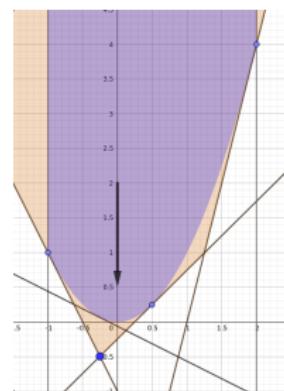
There are infinitely many possible cuts, how to choose them?

Initial cuts

- Added **before** the first LP relaxation is solved
- Reference points chosen based on feasible set only

Separation

- Reference point is a **relaxation solution** $\hat{x} \notin F$
- Valid inequalities $ax \leq b$ violated by \hat{x} : $a\hat{x} > b$
- Thus \hat{x} is **separated** from F



Which Cuts to Add?

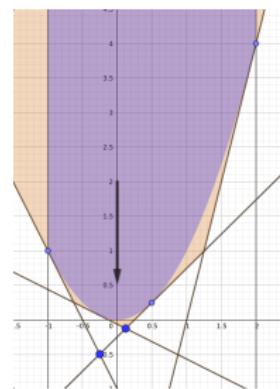
There are infinitely many possible cuts, how to choose them?

Initial cuts

- Added **before** the first LP relaxation is solved
- Reference points chosen based on feasible set only

Separation

- Reference point is a **relaxation solution** $\hat{x} \notin F$
- Valid inequalities $ax \leq b$ violated by \hat{x} : $a\hat{x} > b$
- Thus \hat{x} is **separated** from F



Which Cuts to Add?

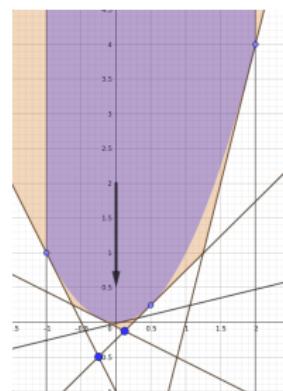
There are infinitely many possible cuts, how to choose them?

Initial cuts

- Added **before** the first LP relaxation is solved
- Reference points chosen based on feasible set only

Separation

- Reference point is a **relaxation solution** $\hat{x} \notin F$
- Valid inequalities $ax \leq b$ violated by \hat{x} : $a\hat{x} > b$
- Thus \hat{x} is **separated** from F



Which Cuts to Add?

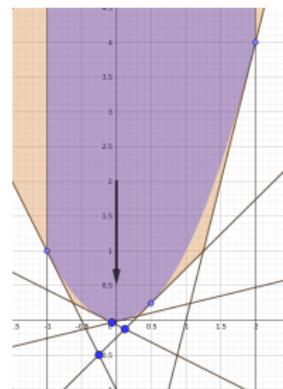
There are infinitely many possible cuts, how to choose them?

Initial cuts

- Added **before** the first LP relaxation is solved
- Reference points chosen based on feasible set only

Separation

- Reference point is a **relaxation solution** $\hat{x} \notin F$
- Valid inequalities $ax \leq b$ violated by \hat{x} : $a\hat{x} > b$
- Thus \hat{x} is **separated** from F



Which Cuts to Add?

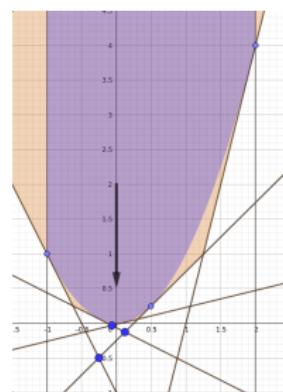
There are infinitely many possible cuts, how to choose them?

Initial cuts

- Added **before** the first LP relaxation is solved
- Reference points chosen based on feasible set only

Separation

- Reference point is a **relaxation solution** $\hat{x} \notin F$
- Valid inequalities $ax \leq b$ violated by \hat{x} : $a\hat{x} > b$
- Thus \hat{x} is **separated** from F



Cut selection: choose from violated cuts using various criteria for cut "usefulness".

Convex Relaxations for Nonconvex MINLPs

Only relaxing integrality no longer provides a lower bound, and gradient cuts might no longer be valid \Rightarrow construct a convex relaxation.

The best relaxation is $\text{conv}(F)$: **convex hull** of F , i.e. the smallest convex set containing F . In general, **cannot be constructed explicitly**.

Convex Relaxations for Nonconvex MINLPs

Only relaxing integrality no longer provides a lower bound, and gradient cuts might no longer be valid \Rightarrow construct a convex relaxation.

The best relaxation is $\text{conv}(F)$: **convex hull** of F , i.e. the smallest convex set containing F . In general, **cannot be constructed explicitly**. Therefore:

- Relax sets given by **individual constraints**: $g_k(x) \leq 0$

Convex Relaxations for Nonconvex MINLPs

Only relaxing integrality no longer provides a lower bound, and gradient cuts might no longer be valid \Rightarrow construct a convex relaxation.

The best relaxation is $\text{conv}(F)$: **convex hull** of F , i.e. the smallest convex set containing F . In general, **cannot be constructed explicitly**. Therefore:

- Relax sets given by **individual constraints**: $g_k(x) \leq 0$
 \uparrow
- Find **convex underestimators** g_k^{cv} of functions g_k :
 $g_k^{cv}(x) \leq g_k(x) \quad \forall x \in [l, u]$

Convex Relaxations for Nonconvex MINLPs

Only relaxing integrality no longer provides a lower bound, and gradient cuts might no longer be valid \Rightarrow construct a convex relaxation.

The best relaxation is $\text{conv}(F)$: **convex hull** of F , i.e. the smallest convex set containing F . In general, **cannot be constructed explicitly**. Therefore:

- Relax sets given by **individual constraints**: $g_k(x) \leq 0$
 \uparrow
- Find **convex underestimators** g_k^{cv} of functions g_k :
 $g_k^{cv}(x) \leq g_k(x) \quad \forall x \in [l, u]$
 \uparrow
- Find and combine relaxations of **simple functions**

Examples of **simple functions**: x^2 , x^k , \sqrt{x} , xy , etc.

Exercise: write the tightest possible convex underestimators for $-x^2$ and x^3 , given $x \in [-1, 1]$ (hint: for x^3 , you need more than one function for each estimator).

Combining Relaxations

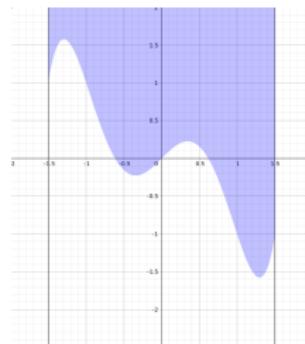
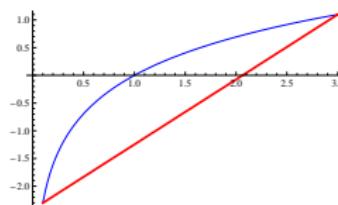
Find underestimator for $g(x) = \phi(\psi_1(x), \dots, \psi_p(x))$, where functions ϕ and ψ_j are "simple", i.e. can be convexified directly.

- **McCormick relaxations** for factorable functions: piecewise continuous relaxations utilising convex and concave envelopes of ϕ and ψ_j .
- **Auxiliary variable** method: introduce variables $y_j = \psi_j(x)$. Then $g(x) = \phi(y_1, \dots, y_p)$. Enables individual handling of each function.

Linear Relaxations for Nonconvex MINLPs

Gradient cuts might no longer be valid!

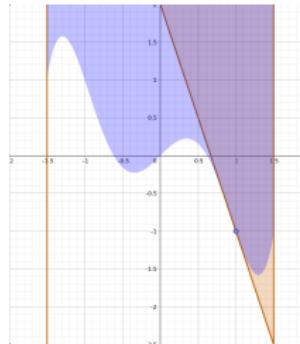
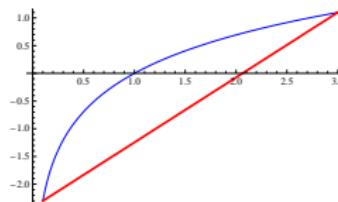
- If possible, **directly** construct linear underestimators for nonconvex functions
 - Secants for concave functions
 - McCormick envelopes for bilinear products
 - etc.
- Construct **gradient cuts** for a **convex relaxation**



Linear Relaxations for Nonconvex MINLPs

Gradient cuts might no longer be valid!

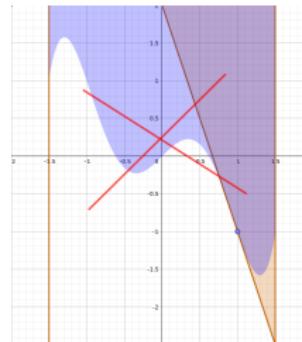
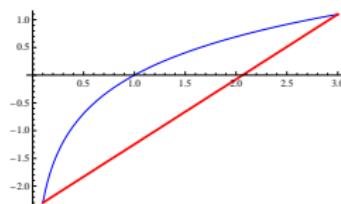
- If possible, **directly** construct linear underestimators for nonconvex functions
 - Secants for concave functions
 - McCormick envelopes for bilinear products
 - etc.
- Construct **gradient cuts** for a **convex relaxation**



Linear Relaxations for Nonconvex MINLPs

Gradient cuts might no longer be valid!

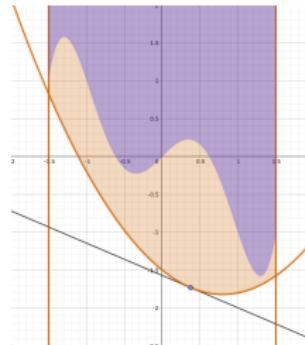
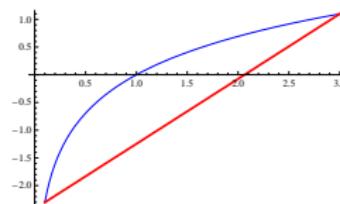
- If possible, **directly** construct linear underestimators for nonconvex functions
 - Secants for concave functions
 - McCormick envelopes for bilinear products
 - etc.
- Construct **gradient cuts** for a **convex relaxation**



Linear Relaxations for Nonconvex MINLPs

Gradient cuts might no longer be valid!

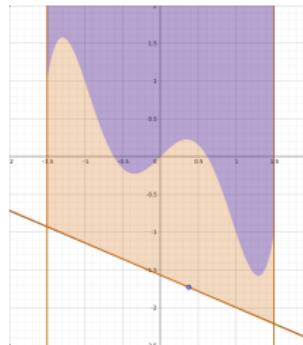
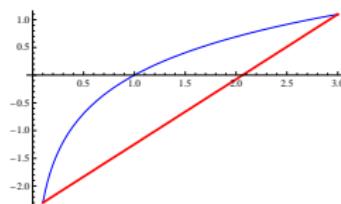
- If possible, **directly** construct linear underestimators for nonconvex functions
 - Secants for concave functions
 - McCormick envelopes for bilinear products
 - etc.
- Construct **gradient cuts** for a **convex relaxation**



Linear Relaxations for Nonconvex MINLPs

Gradient cuts might no longer be valid!

- If possible, **directly** construct linear underestimators for nonconvex functions
 - Secants for concave functions
 - McCormick envelopes for bilinear products
 - etc.
- Construct **gradient cuts** for a **convex relaxation**



Impact of Variable Bounds

Tighter bounds \Rightarrow tighter relaxations.

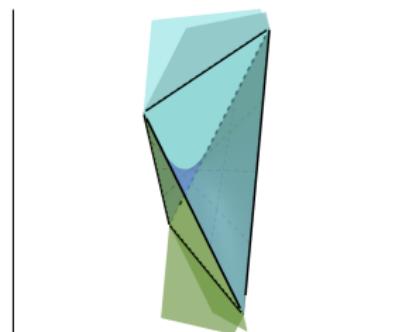
Example: McCormick relaxation of a bilinear product relation $z = xy$:

$$z \leq x^u y + x y^l - x^u y^l$$

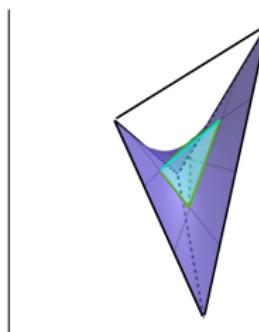
$$z \leq x^l y + x y^u - x^l y^u$$

$$z \geq x^l y + x y^l - x^l y^l$$

$$z \geq x^u y + x y^u - x^u y^u$$



$$(x, y) \in [-1, 2] \times [-2, 2]$$



$$(x, y) \in [0, 1] \times [-1, 1]$$

Tighter bounds obtained from:

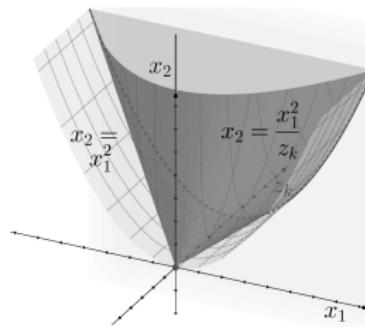
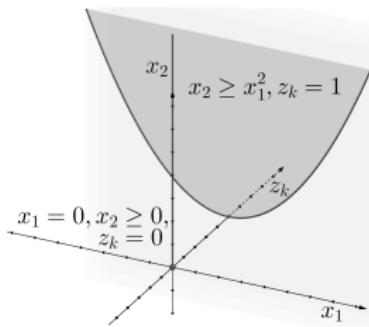
- Branching
- Specialised bound tightening techniques (see linked materials)

Strengthening Relaxations: Using More Constraints

More constraints \Rightarrow tighter relaxations.

Example: **perspective cuts**. Use an additional constraint that requires x to be **semicontinuous**.

$$g(x) \leq 0, \quad l z \leq x \leq u z$$



Proving Optimality: Recap

- What are **relaxations** used for? What are some common types of relaxations?
- What are **gradient cuts** and when can they be applied?
- What is a **convex hull** and what is its practical significance?
- **Auxiliary variable method:** how does it reformulate a function $g(x) = \phi(\psi_1(x), \dots, \psi_p(x))$ and what it is used for.
- How can **relaxations** of nonconvex problems be **strengthened**?

Introduction
ooooooooo

Finding Feasible Solutions
ooo

Proving Optimality
oooooooooooo

Strategy
●ooooo

MINLP in SCIP
oooo

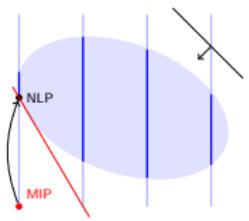
Practical Topics
ooo

Strategy

Strategy

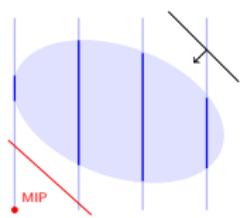
- Goal: bring together the primal and dual side, i.e. find the optimal solution and prove that it is optimal
- A brief overview of algorithms for convex MINLPs
- A closer look at spatial branch and bound

Algorithms for Convex MINLP: Overview



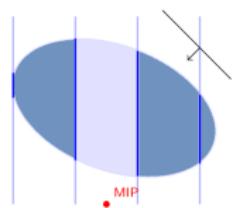
Outer Approximation:

- Solve **MIP relaxations** and **NLP subproblems**
- Add cuts at solutions of NLP subproblems
- Uses the equivalence of MINLP to MILP (see notes)



Extended Cutting Planes:

- Solve MIP relaxations
- Add cuts at solutions of **MIP relaxations**



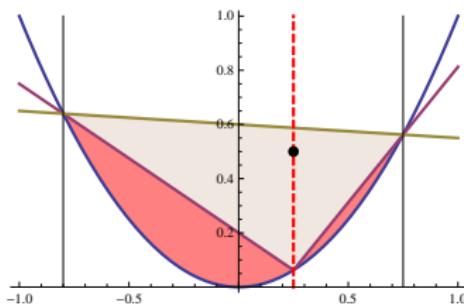
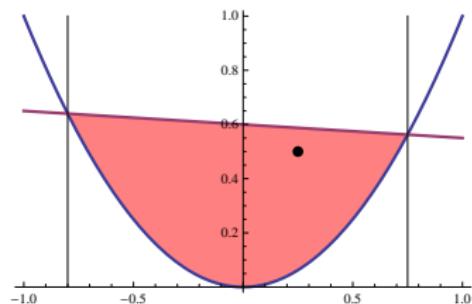
Branch and Bound:

- Generalisation of MILP B&B
- The continuous relaxation is **nonlinear** (but convex)
- Different choices between LP and NLP relaxations

Algorithms for Nonconvex MINLP: Spatial Branching

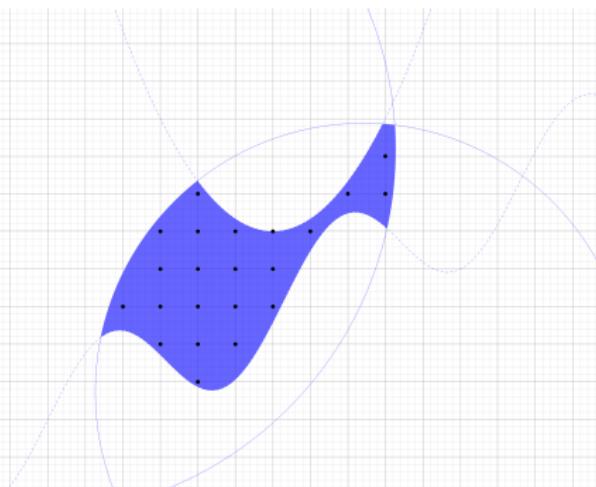
Branching on variables in violated **nonconvex** constraints, because variable bounds determine the convex relaxation, e.g.,

$$x^2 \leq \ell^2 + \frac{u^2 - \ell^2}{u - \ell} (x - \ell) \quad \forall x \in [\ell, u].$$



Spatial Branch and Bound

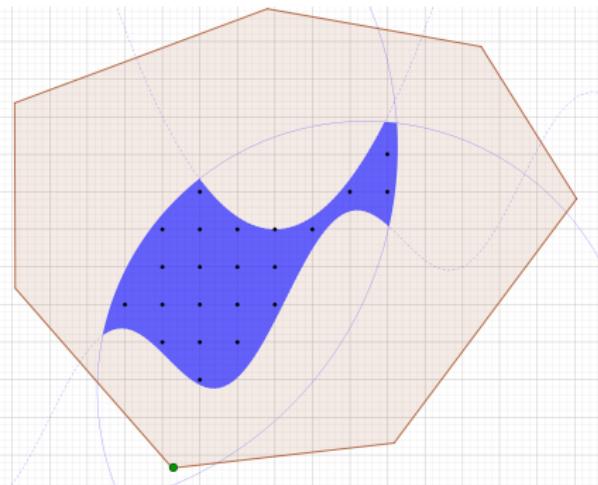
- Solve a **relaxation** → lower bound
- Run heuristics to **look for feasible solutions** → upper bound
- **Branch** on a suitable variable
- **Discard** parts of the tree that are infeasible or where lower bound > best known upper bound
- Repeat **until gap is below** given tolerance



Smaller domains → improved relaxations → improved bounds.

Spatial Branch and Bound

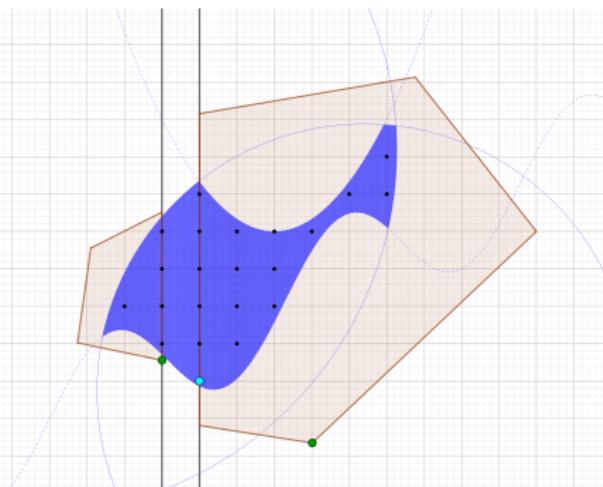
- Solve a **relaxation** → lower bound
- Run heuristics to **look for feasible solutions** → upper bound
- **Branch** on a suitable variable
- **Discard** parts of the tree that are infeasible or where lower bound > best known upper bound
- Repeat **until gap is below** given tolerance



Smaller domains → improved relaxations → improved bounds.

Spatial Branch and Bound

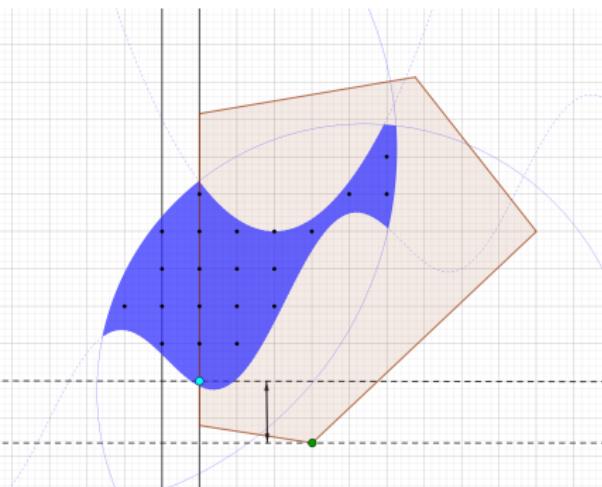
- Solve a **relaxation** → lower bound
- Run heuristics to **look for feasible solutions** → upper bound
- **Branch** on a suitable variable
- **Discard** parts of the tree that are infeasible or where lower bound > best known upper bound
- Repeat **until gap is below** given tolerance



Smaller domains → improved relaxations → improved bounds.

Spatial Branch and Bound

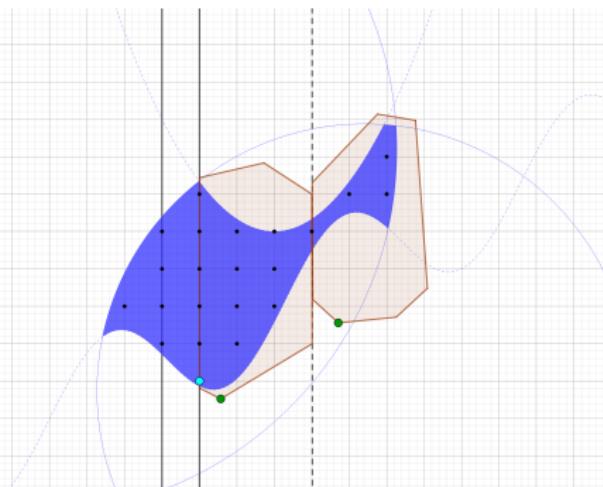
- Solve a **relaxation** → lower bound
- Run heuristics to **look for feasible solutions** → upper bound
- **Branch** on a suitable variable
- **Discard** parts of the tree that are infeasible or where lower bound > best known upper bound
- Repeat **until gap is below** given tolerance



Smaller domains → improved relaxations → improved bounds.

Spatial Branch and Bound

- Solve a **relaxation** → lower bound
- Run heuristics to **look for feasible solutions** → upper bound
- **Branch** on a suitable variable
- **Discard** parts of the tree that are infeasible or where lower bound > best known upper bound
- Repeat **until gap is below** given tolerance



Smaller domains → improved relaxations → improved bounds.

Strategy: Recap

- There are several different approaches to solving convex MINLPs.
- In addition to branching to enforce integrality, what other type of branching does spatial B&B employ?
- Can you recall the main steps of the spatial B&B algorithm?

Introduction
ooooooooo

Finding Feasible Solutions
ooo

Proving Optimality
oooooooooooo

Strategy
oooooo

MINLP in SCIP
●ooo

Practical Topics
ooo

MINLP in SCIP

MINLP in SCIP

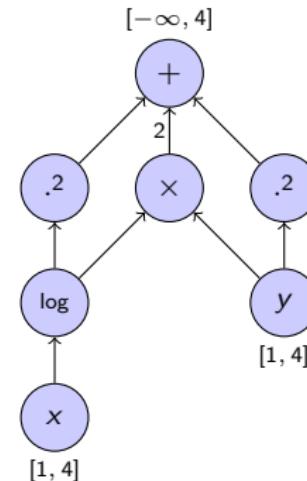
- SCIP implements LP-based spatial B&B
- Convex relaxations are constructed via the auxiliary variable method
- The handling of nonlinear constraints is based on expression graphs

Expression Trees

Algebraic structure of nonlinear constraints is stored in one directed acyclic graph:

- nodes: variables, operations, constraints
- arcs: flow of computation

$$\log(x)^2 + 2 \log(x)y + y^2$$



Expression Trees

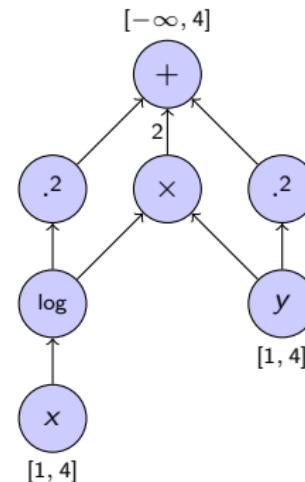
Algebraic structure of nonlinear constraints is stored in one directed acyclic graph:

- nodes: variables, operations, constraints
- arcs: flow of computation

Operators:

- variable index, constant
- $+, -, *, \div$
- $.^2, \sqrt{\cdot}, .^p$ ($p \in \mathbb{R}$), $.^n$ ($n \in \mathbb{Z}$),
 $x \mapsto x|x|^{p-1}$ ($p > 1$)
- exp, log
- min, max, abs
- \sum, \prod , affine-linear, quadratic, signomial
- (user)

$$\log(x)^2 + 2 \log(x)y + y^2$$



Expression Trees

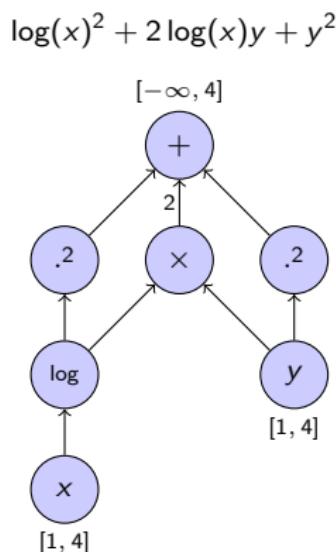
Algebraic structure of nonlinear constraints is stored in one directed acyclic graph:

- nodes: variables, operations, constraints
- arcs: flow of computation

Operators:

- variable index, constant
- $+, -, *, \div$
- $.^2$, $\sqrt{\cdot}$, $.^p$ ($p \in \mathbb{R}$), $.^n$ ($n \in \mathbb{Z}$),
 $x \mapsto x|x|^{p-1}$ ($p > 1$)
- \exp , \log
- \min , \max , abs
- \sum , \prod , affine-linear, quadratic, signomial
- (user)

Additional constraint handlers: quadratic,
abspower ($x \mapsto x|x|^{p-1}$, $p > 1$), SOC

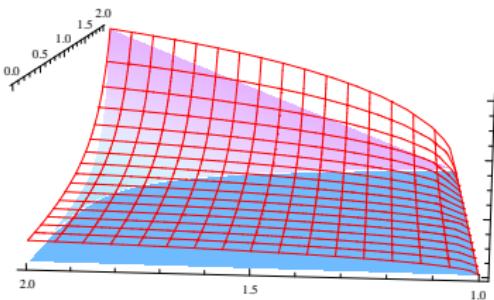


Reformulation (During Presolve)

Goal: **reformulate constraints** such that only **elementary cases** (convex, concave, odd power, quadratic) remain. Implements the **auxiliary variable method**.

Example:

$$g(x) = \sqrt{\exp(x_1^2) \ln(x_2)}$$



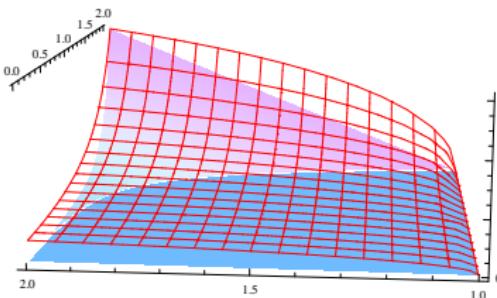
Introduces **new variables** and **new constraints**.

Reformulation (During Presolve)

Goal: **reformulate constraints** such that only **elementary cases** (convex, concave, odd power, quadratic) remain. Implements the **auxiliary variable method**.

Example:

$$g(x) = \sqrt{\exp(x_1^2) \ln(x_2)}$$



Reformulation:

$$g = \sqrt{y_1}$$

$$y_1 = y_2 y_3$$

$$y_2 = \exp(y_4)$$

$$y_3 = \ln(x_2)$$

$$y_4 = x_1^2$$

Introduces **new variables** and **new constraints**.

Introduction
ooooooooo

Finding Feasible Solutions
ooo

Proving Optimality
oooooooooooo

Strategy
oooooo

MINLP in SCIP
oooo

Practical Topics
●oo

Practical Topics

Impact of Modelling

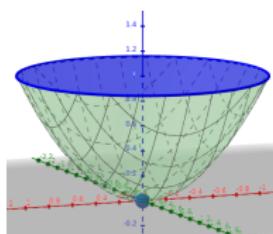
If you know your **problem structure** - use it!

Example: x and y contained in circle of radius c if $z = 1$ and are both zero if $z = 0$.

One could model this as:

$$x^2 + y^2 \leq cz$$

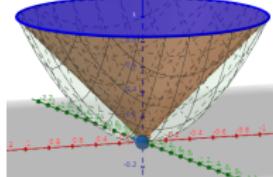
$$x, y \in \mathbb{R}, z \in \{0, 1\}$$



Or as:

$$x^2 + y^2 \leq cz^2$$

$$x, y \in \mathbb{R}, z \in \{0, 1\}$$



These describe the same feasible set ($z^2 = z$ if $z \in \{0, 1\}$). But the second formulation leads to a **tighter continuous relaxation** ($z^2 < z$ if $z \in (0, 1)$).

How to Experiment

- Performance variability
 - Significant changes in performance caused by small changes in model/algorithms
 - Occurs in MILP, but tends to be even more pronounced in MINLP
- Obtaining more reliable results
 - If possible and makes sense, use large and heterogeneous testsets
 - Take advantage of performance variability: model permutations (reordering variables and constraints) can help against random effects (in SCIP, this is controlled by a parameter)
- Using solver statistics
 - Information on tree nodes, primal and dual bounds, effects of solver components
 - Helpful for finding bottlenecks
- Isolating feature effects
 - Turn off some components to get rid of some random effects...
 - or to analyse interaction: some component might make the feature redundant, etc.