Global Optimization of Mixed-Integer Nonlinear Programs

Ksenia Bestuzheva,
bestuzheva@zib.de

CO@Work · Berlin, Germany · September 19, 2024

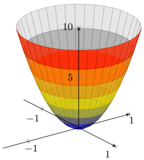
SPONSORED BY THE
Federal Ministry of Education and Research

Introduction

Mixed-Integer Nonlinear Programs

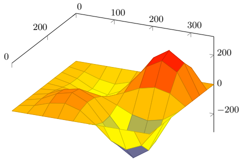
$$
\begin{aligned}
\min\ & c^\mathsf{T} x \\
\text{s.t. } & Ax \le b \\
& g_k(x) \le 0 && \forall k \in [m] \\
& x_i \in [\ell_i, u_i] && \forall i \in [n] \\
& x_i \in \mathbb{Z} && \forall i \in \mathcal{I} \subseteq [n]
\end{aligned}
$$

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



convex

or



nonconvex

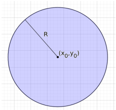
Examples of Nonlinearities

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

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

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

- **Distance** constraints



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

- etc.

Solving a Mixed-Integer Optimization Problem

Two major tasks:

1. Finding and improving feasible solutions (**primal side**)
   - **Ensure feasibility**, not necessarily maintaining optimality
   - Important for many practical applications

2. Proving optimality (**dual side**)
   - **Ensure optimality**, not necessarily maintaining feasibility
   - Necessary in order to actually solve the problem
   - Provides guarantees on solution quality

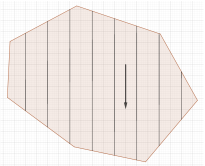
Linked by:

3. Strategy
   - **Ensure convergence**
   - Divide: branching, decompositions, ...
   - Put together all components to find a solution that is **feasible** and **optimal** (or within a proven gap from the optimum)

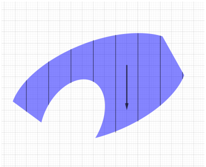
Nonlinearity Brings New Challenges

1. Primal side
   - Feasible solutions must also satisfy nonlinear constraints
   - If nonconvex: local optima, local infeasibility
2. Dual side
   - NLP relaxations capture the problem better, LP relaxations are faster
   - Strong cuts needed for various nonlinearities
   - If nonconvex: straightforward continuous relaxation no longer provides a lower bound
3. Strategy
   - Need to account for all of the above
   - Warmstart for NLP is less efficient than for LP

- More numerical issues
- NLP solving is less efficient and reliable than LP



$\rightarrow$



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 solution found so far is referred to as **best feasible** or **incumbent**. It provides an **upper/primal bound** on the optimal value.

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

Nonconvex case: 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

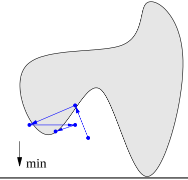
Primal Heuristics for MINLPs

MILP heuristics

- Can be applied to MINLPs (solutions violating nonlinear constraints can be passed to NLP local search).

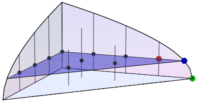
NLP local search

- **Fix integer variables** to values at reference point; solve the NLP.
- Reference point examples: integer feasible solution of the LP relaxation, solution from an MILP heuristic.



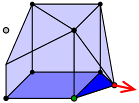
Undercover

- **Fix some variables** so that constraints become **linear**; solve the MILP.



Sub-MINLP

- Extensions of MILP large neighbourhood search heuristics.
- Search around **promising** solutions.
- The region is restricted by additional constraints and/or fixing variables.



Proving Optimality

Proving Optimality

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

Finding Lower Bounds: Relaxations

A **relaxation** R of a **feasible set** F is a set such that $F \subseteq R$.

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

Relaxations can be:

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

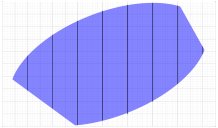
MILP and MINLP relaxations are sometimes used as well.

It is preferable that relaxations:

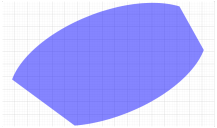
- Are **tight**: small $R \setminus F$, dual bound close to the optimal value
- Are **compact**: avoid excessive numbers of constraints and variables
- Have **reasonable numerics**

Relaxations for Convex MINLPs

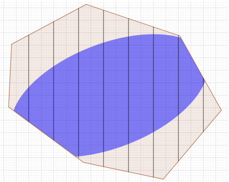
- Relax integrality



$\rightarrow$



- Replace the nonlinear set with a linear outer approximation



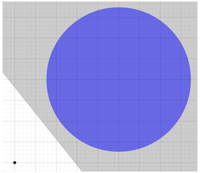
- Relax integrality + linear outer approximation $\rightarrow$ LP relaxation

Outer Approximating Convex Constraints

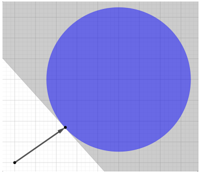
A **linear inequality** $ax \leq b$ is **valid** if $x \in F \Rightarrow ax \leq b$ (**cutting planes**, or **cuts**, are valid inequalities)

Given constraint $g(x) \leq 0$ 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



Convex Relaxations for Nonconvex MINLPs

Relaxing integrality no longer provides a lower bound, and gradient cuts are no longer valid $\Rightarrow$ construct a convex relaxation.

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

Convex Relaxations for Nonconvex MINLPs

Relaxing integrality no longer provides a lower bound, and gradient cuts are no longer valid $\Rightarrow$ construct a convex relaxation.

The best relaxation is *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

Relaxing integrality no longer provides a lower bound, and gradient cuts are no longer valid $\Rightarrow$ construct a convex relaxation.

The best relaxation is $conv(F)$: **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: $g_k^{cv}(x) \leq g_k(x)\ \forall x \in [l, u]$

Convex Relaxations for Nonconvex MINLPs

Relaxing integrality no longer provides a lower bound, and gradient cuts are no longer valid $\Rightarrow$ construct a convex relaxation.

The best relaxation is $conv(F)$: **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: $g_k^{cv}(x) \leq g_k(x) \; \forall x \in [l, u]$

  $\uparrow$

- Find and combine relaxations of **simple functions**

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

Combining Relaxations

Expression trees

Algebraic structure of nonlinear constraints can be stored in **one** directed acyclic graph:

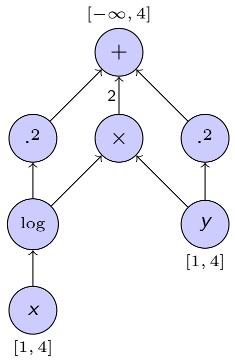
- nodes: variables, operations, constraints
- arcs: flow of computation

Underestimators of compositions

Find underestimator for $g(x) = \phi(\psi_1(x), \ldots, \psi_p(x))$, where functions can be convexified directly.

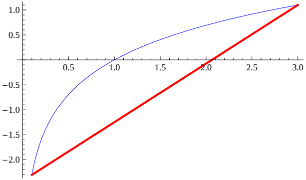
- **McCormick relaxations** for factorable functions: piecewise continuous relaxations utilising convex and concave envelopes of ϕ and ψ_j.
- **Auxiliary variable** method: introduce variables $y_j = \psi_j(x)$. Then $g(x) = \phi(y_1, \ldots, y_p)$. Enables individual handling of each function.

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

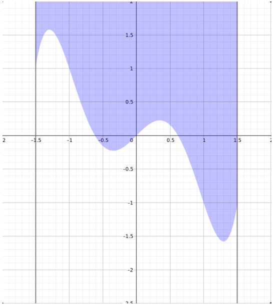


Linear Relaxations for Nonconvex MINLPs

- If possible, **directly** construct linear underestimators for nonconvex functions
  - Secants for concave functions
  - McCormick envelopes for bilinear products
  - etc.

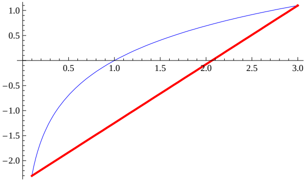


- Construct **gradient cuts** for a **convex relaxation**

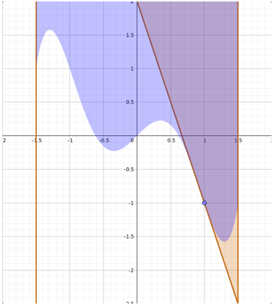


Linear Relaxations for Nonconvex MINLPs

- If possible, **directly** construct linear underestimators for nonconvex functions
  - Secants for concave functions
  - McCormick envelopes for bilinear products
  - etc.

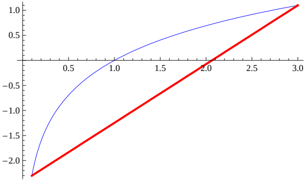


- Construct **gradient cuts** for a **convex relaxation**

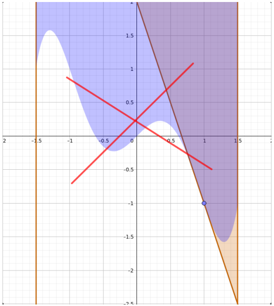


Linear Relaxations for Nonconvex MINLPs

- If possible, **directly** construct linear underestimators for nonconvex functions
  - Secants for concave functions
  - McCormick envelopes for bilinear products
  - etc.

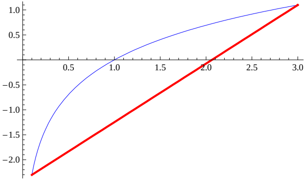


- Construct **gradient cuts** for a **convex relaxation**

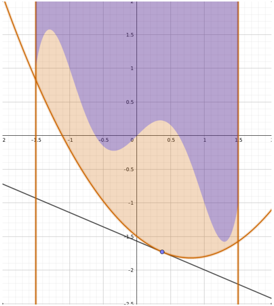


Linear Relaxations for Nonconvex MINLPs

- If possible, **directly** construct linear underestimators for nonconvex functions
  - Secants for concave functions
  - McCormick envelopes for bilinear products
  - etc.

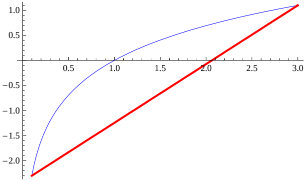


- Construct **gradient cuts** for a **convex relaxation**

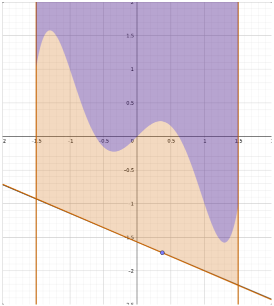


Linear Relaxations for Nonconvex MINLPs

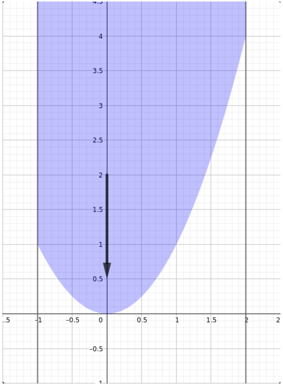
- If possible, **directly** construct linear underestimators for nonconvex functions
  - Secants for concave functions
  - McCormick envelopes for bilinear products
  - etc.



- Construct **gradient cuts** for a **convex relaxation**



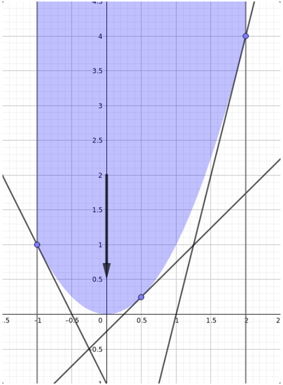
Cut Generation



Cut Generation

Initial cuts

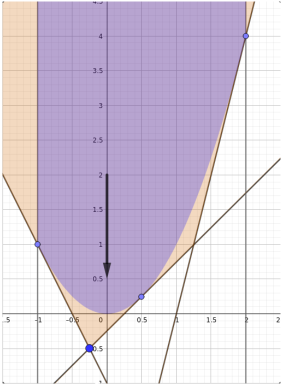
- Added **before** the first LP relaxation is solved
- Reference points chosen based on feasible set only
- Aiming for a compact formulation that roughly captures F and yields a reference point for separation



Cut Generation

Initial cuts

- Added **before** the first LP relaxation is solved
- Reference points chosen based on feasible set only
- Aiming for a compact formulation that roughly captures F and yields a reference point for separation



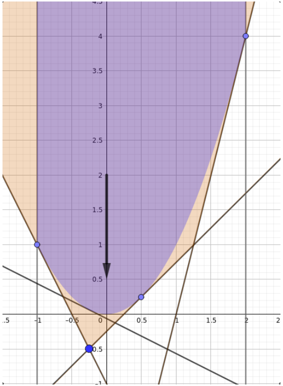
Cut Generation

Initial cuts

- Added **before** the first LP relaxation is solved
- Reference points chosen based on feasible set only
- Aiming for a compact formulation that roughly captures F and yields a reference point for separation

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



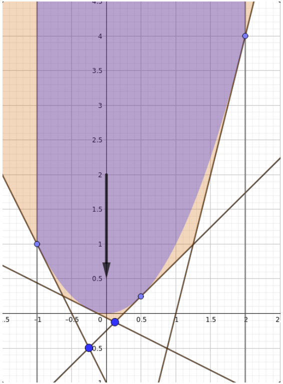
Cut Generation

Initial cuts

- Added **before** the first LP relaxation is solved
- Reference points chosen based on feasible set only
- Aiming for a compact formulation that roughly captures F and yields a reference point for separation

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



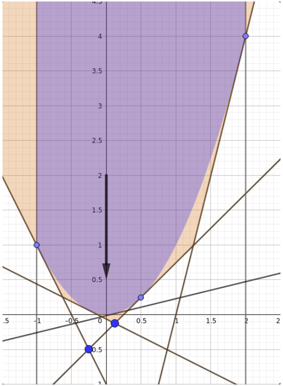
Cut Generation

Initial cuts

- Added **before** the first LP relaxation is solved
- Reference points chosen based on feasible set only
- Aiming for a compact formulation that roughly captures F and yields a reference point for separation

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



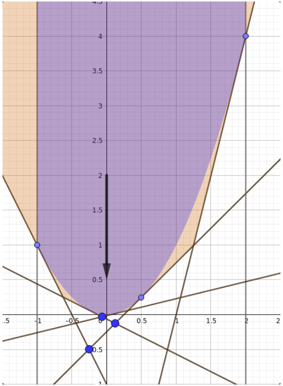
Cut Generation

Initial cuts

- Added **before** the first LP relaxation is solved
- Reference points chosen based on feasible set only
- Aiming for a compact formulation that roughly captures F and yields a reference point for separation

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



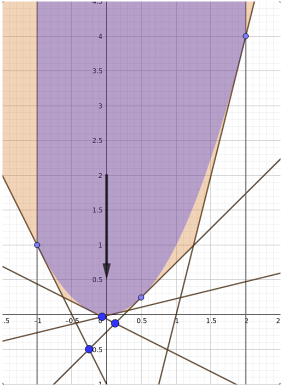
Cut Generation

Initial cuts

- Added **before** the first LP relaxation is solved
- Reference points chosen based on feasible set only
- Aiming for a compact formulation that roughly captures F and yields a reference point for separation

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



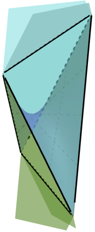
Cut selection: choose from violated cuts using various criteria for cut usefulness (e.g. violation, orthogonality, density, etc.).

Strengthening Relaxations: Tighter Variable Bounds

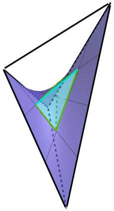
Tighter bounds $\Rightarrow$ tighter relaxations.

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:

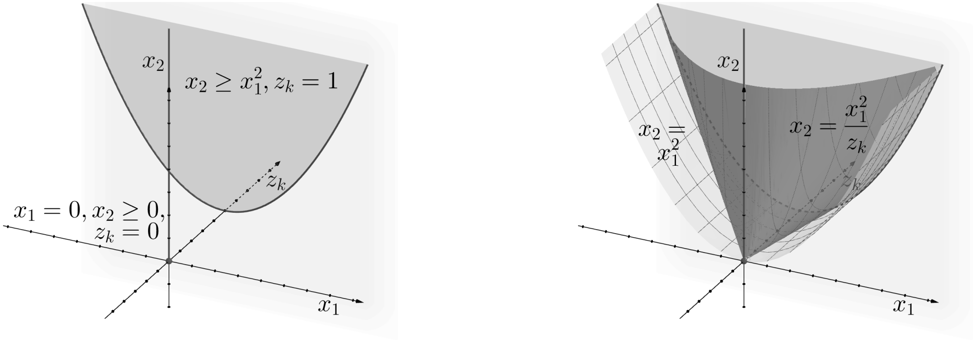
- **Branching** (more on this in the Strategy section)
- Specialized **bound tightening** techniques
- **Piecewise-linear** relaxations

Strengthening Relaxations: Using More Constraints

More constraints $\Rightarrow$ tighter relaxations.

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

$$g(x) \leq 0, \;\; lz \leq x \leq uz$$



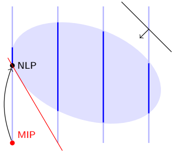
Strategy

Strategy

- **Goal**: bring together the primal and dual side, i.e. find the optimal solution and prove that it is optimal
  - Sometimes finding a solution within a certain proven gap is enough

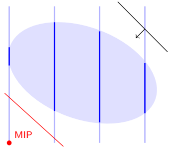
- A brief overview of algorithms for convex MINLPs
- Spatial branch and bound for nonconvex MINLPs

Algorithms for Convex MINLP: Overview



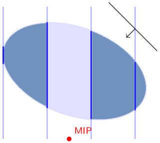
Outer Approximation:

- Solve **MILP relaxations** and **NLP subproblems**
- Add gradient cuts at solutions of NLP subproblems
- Uses the equivalence of MINLP to MILP



Extended Cutting Planes:

- Solve MILP relaxations
- Add gradient cuts at solutions of **MILP relaxations**



Branch and Bound:

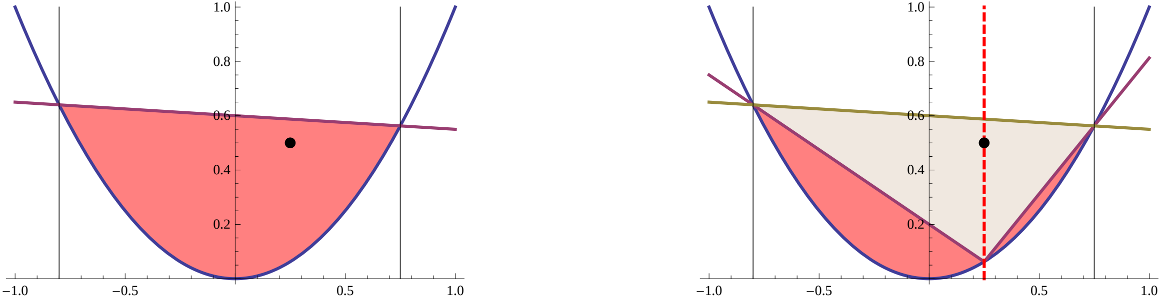
- Generalization of MILP B&B
- The continuous relaxation is **nonlinear** (but convex)
- Different choices between LP and NLP relaxations

Algorithms for Nonconvex MINLP: Spatial Branching

- Recall: variable bounds determine the convex relaxation, e.g.,

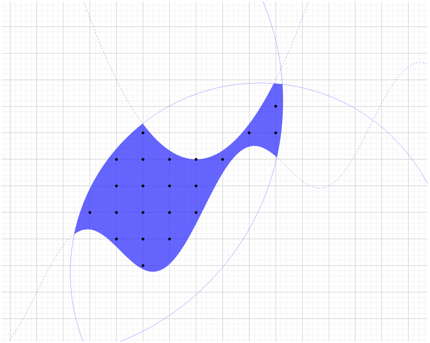
$$x^2 \leq \ell^2 + \frac{u^2 - \ell^2}{u - \ell}(x - \ell) \quad \forall x \in [\ell, u]$$

- Branch on variables in **violated nonconvex** constraints to improve relaxations



Spatial Branch and Bound for MINLP

- Solve a **relaxation** $\rightarrow$ lower bound
- **Branch** on a suitable variable (integer, or continuous in a violated nonconvex constraint)
- If a solution is **integer feasible** and **satisfies nonlinear constraints** $\rightarrow$ upper bound
- **Discard** parts of the tree that are infeasible or where lower bound $\geq$ best known upper bound
- **Repeat** until gap is below given tolerance

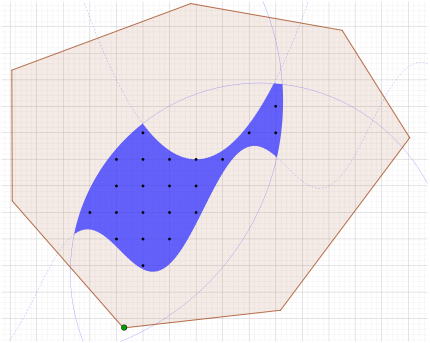


Smaller domains $\rightarrow$ improved relaxations $\rightarrow$ improved bounds.

In practice algorithms also include presolving, heuristics, propagation, etc.

Spatial Branch and Bound for MINLP

- Solve a **relaxation** → lower bound
- **Branch** on a suitable variable (integer, or continuous in a violated nonconvex constraint)
- If a solution is **integer feasible** and **satisfies nonlinear constraints** → upper bound
- **Discard** parts of the tree that are infeasible or where lower bound ≥ best known upper bound
- **Repeat** until gap is below given tolerance

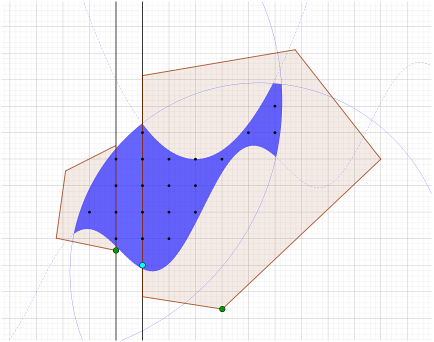


Smaller domains → improved relaxations → improved bounds.

In practice algorithms also include presolving, heuristics, propagation, etc.

Spatial Branch and Bound for MINLP

- Solve a **relaxation** $\rightarrow$ lower bound
- **Branch** on a suitable variable (integer, or continuous in a violated nonconvex constraint)
- If a solution is **integer feasible** and **satisfies nonlinear constraints** $\rightarrow$ upper bound
- **Discard** parts of the tree that are infeasible or where lower bound $\geq$ best known upper bound
- **Repeat** until gap is below given tolerance

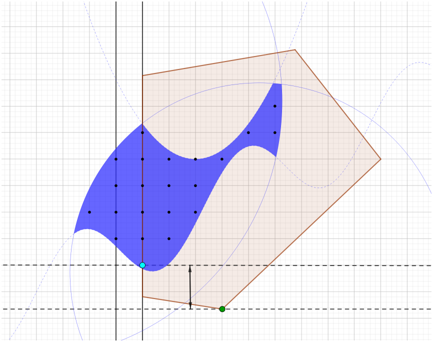


Smaller domains $\rightarrow$ improved relaxations $\rightarrow$ improved bounds.

In practice algorithms also include presolving, heuristics, propagation, etc.

Spatial Branch and Bound for MINLP

- Solve a **relaxation** → lower bound
- **Branch** on a suitable variable (integer, or continuous in a violated nonconvex constraint)
- If a solution is **integer feasible** and **satisfies nonlinear constraints** → upper bound
- **Discard** parts of the tree that are infeasible or where lower bound ≥ best known upper bound
- **Repeat** until gap is below given tolerance

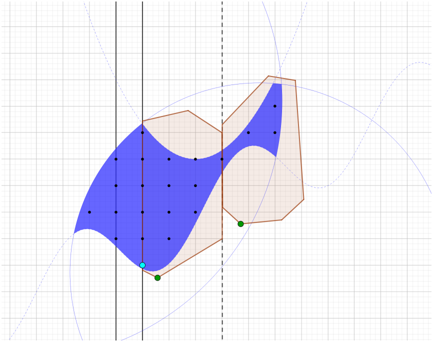


Smaller domains → improved relaxations → improved bounds.

In practice algorithms also include presolving, heuristics, propagation, etc.

Spatial Branch and Bound for MINLP

- Solve a **relaxation** → lower bound
- **Branch** on a suitable variable (integer, or continuous in a violated nonconvex constraint)
- If a solution is **integer feasible** and **satisfies nonlinear constraints** → upper bound
- **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.

In practice algorithms also include presolving, heuristics, propagation, etc.

MINLP in SCIP

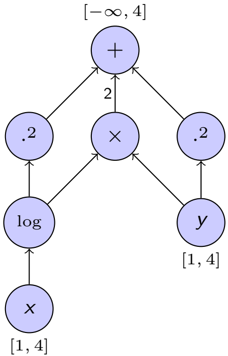
MINLP in SCIP

- SCIP is a constraint integer programming (CIP) solver
- CIP is a generalization of MILP allowing for arbitrary constraints as long as a tractable relaxation can be built
- CIP includes MINLP (and a few other problem classes)

- 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**
- The nonlinear constraint handler coordinates sub-plugins handling various nonlinearities

Expression Trees in SCIP

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

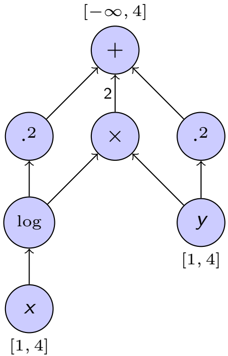


Expression Trees in SCIP

Operators:

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

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

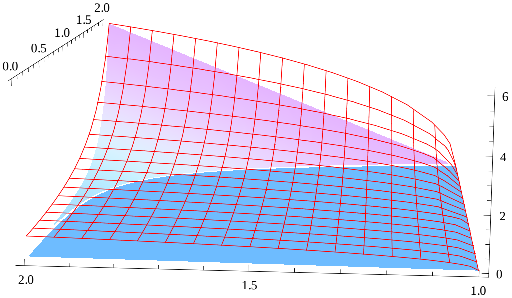


Reformulation (During Presolve)

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

Example:

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



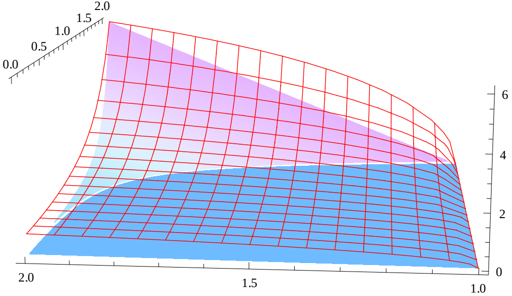
Introduces **new variables and new constraints**.

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**.

Problem with classic approach

Explicit reformulation of constraints ...

- ... **loses the connection to the original problem**.
- ... **loses distinction between original and auxiliary variables**. Thus, we may branch on auxiliary variables.
- ... **prevents simultaneous exploitation of overlapping structures**.

SCIP's Handling of Reformulations

- Avoid explicit split-up of constraints
- Introduce **extended formulation** as **annotation to the original** formulation
- Use extended formulation for **relaxation**
- Use original formulation for **feasibility checking**
- To resolve infeasibility in original constraints, tighten relaxation of extended formulation
- The original formulation is kept
- This avoids wrong feasibility checks

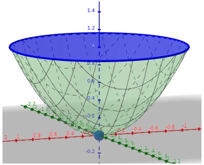
Practical Topics

Impact of Modeling

The choice of formulation makes a difference.

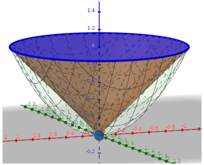
Example: x and y contained in circle of radius c if $z = 1$ and are both zero if $z = 0$.

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/algorithm**
  - 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 (for example, 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.

Recap

- MINLPs combine integrality and nonlinearity
- Algorithms are based on finding and improving primal and dual bounds
- Primal bounds are found by heuristics; there are many extensions of MILP techniques
- Dual bounds are found via relaxations (usually convex or linear)
- Spatial branch and bound solves nonconvex MINLPs globally by also branching on continuous variables

Questions?