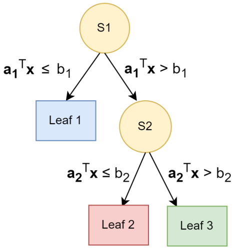
Global Optimization: A Machine Learning Approach [1]

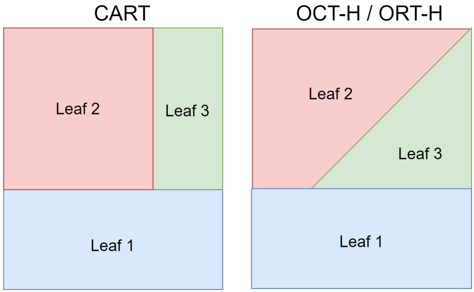
Georgios Margaritis (MIT ORC),
Joint work with Prof. Dimitris Bertsimas.

May 29, 2025

[1] Bertsimas, D., Margaritis, G. Global optimization: a machine learning approach. J Glob Optim 91, 1–37 (2025).

Primer: OCT-H





Optimal Classification/Regression Trees with Hyperplanes (OCT-H/ORT-H)[2]:

- Generalization of CART trained using Optimization
- Splits → general hyperplanes
- Leaves → polyhedras

[2] Bertsimas, D., & Dunn, J. (2017). Optimal classification trees. Machine Learning, 106(7), 1039–1082.

Outline

Outline

Introduction

Approach

Results

Conclusion

Introduction: Problem

- Solve Global Optimization problems:

$$\begin{aligned} \min \quad & f(\boldsymbol{x}) \\ \text{s.t.} \quad & g_i(\boldsymbol{x}) \leq 0, \; i \in \bar{I} \\ & \boldsymbol{x} \in \mathbb{Z}^m \times \mathbb{R}^{n-m} \end{aligned}$$

  where f, g_i can be non-convex.
- Very general family of problems.
- **Applications**: Chemical Engineering, Computational Biology, Mechanical & Aerospace engineering etc.

Introduction: Other approaches - Pitfalls

BARON:

- Arguably the best commercial optimizer for Global Optimization.
- Uses branch-and-reduce and constraint relaxations.

Such methods:

- Utilize the specific mathematical structure of the constraint:
  - e.g. BARON only allow a subset of primitives: $\exp(x), \ln(x), x^\alpha$ and b^x
- What happens with:
  - More general primitives?
  - Black-box and implicit constraints? (e.g. simulation-based)
  - Data-driven constraints?

Introduction: OCTHaGOn

OCTHaGOn (Bertsimas & Ozturk, 2022):

- Global Optimization framework without optimality guarantees
- Applicable to very general constraints, both explicit and black-box
- Key idea: $\rightarrow$ Model non-convexities using hyperplane-based Decision Trees (OCT-H) and MIO

Our approach:

- Extension of OCTHaGOn:
  - More ML models
  - Better sampling
  - Robust Optimization
  - Constraint Relaxations
- Significantly improves solution quality

Outline

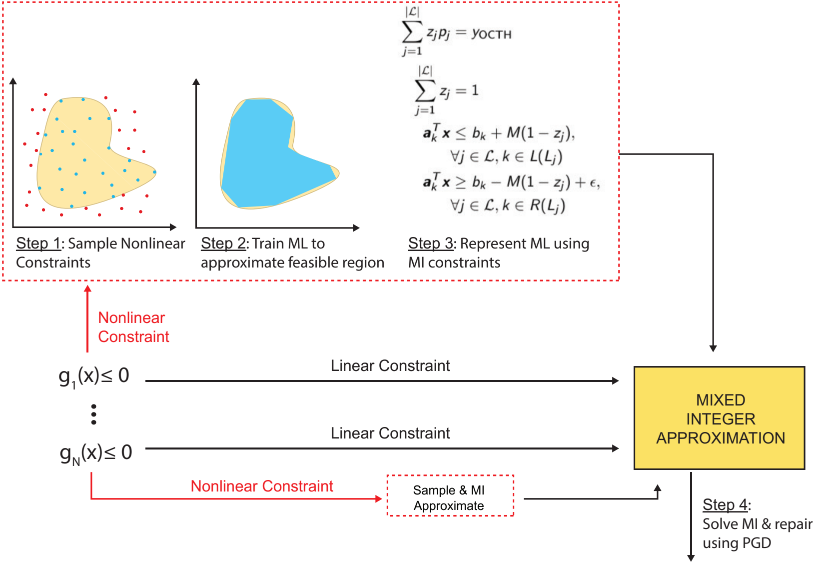
Problem

- Problem:

$$\begin{aligned} \min \quad & f(\boldsymbol{x}) \\ \text{s.t.} \quad & g_i(\boldsymbol{x}) \leq 0,\ i \in \bar{I} \\ & \boldsymbol{x} \in \mathbb{Z}^m \times \mathbb{R}^{n-m} \end{aligned}$$

- **Idea**: Construct and solve a Mixed Integer approximation of the problem.
  1. Use MI-representable ML surrogates to approximate nonlinear constraints.
  2. Model nonlinearities/non-convexities using MI optimization.

Pipeline



$\sum_{j=1}^{|\mathcal{L}|} z_j p_j = y_{\text{OCTH}}$
$\sum_{j=1}^{|\mathcal{L}|} z_j = 1$
$\mathbf{a}_k^T \mathbf{x} \leq b_k + M(1 - z_j),$
$\forall j \in \mathcal{L}, k \in L(L_j)$
$\mathbf{a}_k^T \mathbf{x} \geq b_k - M(1 - z_j) + \epsilon,$
$\forall j \in \mathcal{L}, k \in R(L_j)$
Step 1: Sample Nonlinear Constraints
Step 2: Train ML to approximate feasible region
Step 3: Represent ML using MI constraints
Nonlinear Constraint
$g_1(x) \leq 0$
Linear Constraint
$g_N(x) \leq 0$
Linear Constraint
Nonlinear Constraint
Sample & MI Approximate
MIXED INTEGER APPROXIMATION
Step 4: Solve MI & repair using PGD

Step 1: Sample Nonlinear Constraints

- Goal → Obtain binary samples of constraint satisfaction:

$$D_{\mathcal{C}}^{i} = \{(\widetilde{\boldsymbol{x}}_{\boldsymbol{k}}, \mathbb{1}\{g_i(\widetilde{\boldsymbol{x}}_{\boldsymbol{k}}) \leq 0\})\}_{k=1}^{\tilde{n}}$$

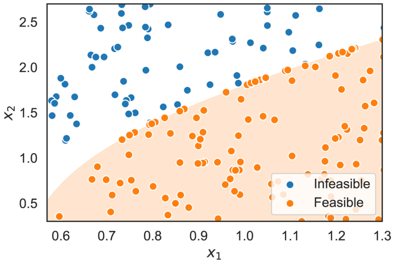
- Static sampling methods - independent of function landscape:

  1. Boundary Sampling:
     - Goal → Sample extreme points
  2. Optimal Latin Hypercube Sampling (OLH):
     - Space-filling characteristics

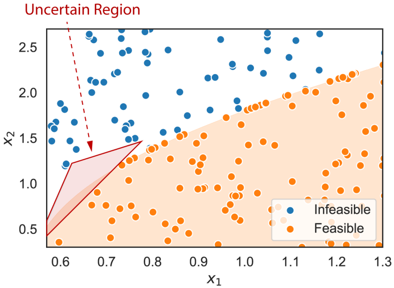
- Adaptive sampling method - adapts to function landscape:
  1. OCT Sampling:
     - Recursively resamples hard-to-approximate areas.

Step 1: Sample Nonlinear Constraints

- Oversample hard-to-approximate areas of the constraints:
  - High nonlinearities.
  - Close to the decision boundary.

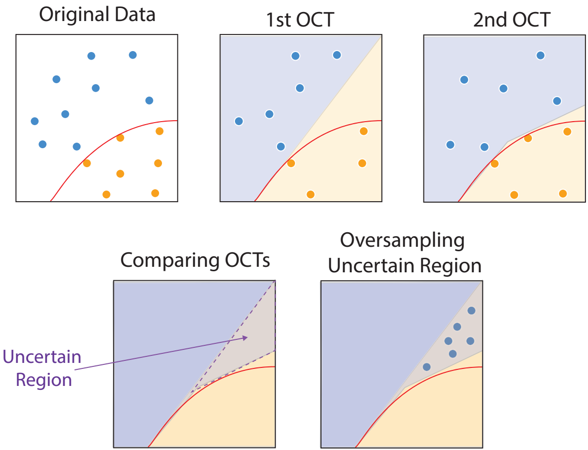


(a) Initial Random Samples



(b) Resampling Uncertain Regions

Step 1: Sample Nonlinear Constraints



- OCT Sampling:
  1. Train multiple OCTs on subsets of the samples.
  2. Find areas/leaf intersections with high disagreements.
  3. Resample these areas using hit-and-run sampling.

Step 1: Sample Nonlinear Constraints

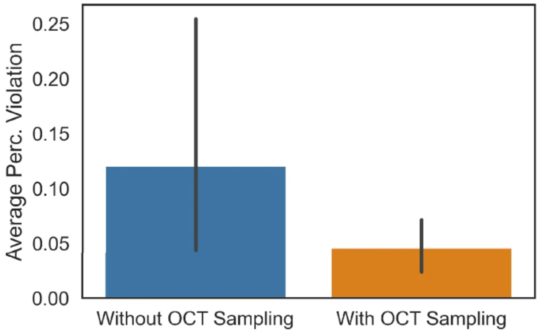


Figure: Effect of OCT Sampling on Constraint Violation.

Step 2: Constraint Approximation with ML models

- For each nonlinear constraint:
  - Approximate constraint with MI representable ML models.
  - Models trained on samples from Step 1
- Some models may be better for different types of constraints
- We use:
  - Decision Trees with Hyperplane splits (OCT-H)
  - Support Vector Machines (SVMs)
  - Use Gradient Boosted Trees (GBMs)
  - ReLU-based multi-layer perceptrons (ReLU MLPs)
- Use cross-validation procedure to select the best model for each constraint

Step 3: MI representation of ML models → SVMs

SVM:

- Train SVM on binary classification samples $D_{\mathcal{C}}$

$$\begin{aligned} \min_{\beta_0 \in \mathbb{R},\, \boldsymbol{\beta}, \boldsymbol{z} \in \mathbb{R}^{\bar{n}}} \quad & ||\boldsymbol{\beta}||_2^2 + C \sum_{i=1}^{\bar{n}} z_i \\ \text{s.t.} \quad & z_i \geq y_i - \beta_0 - \boldsymbol{x}_i^T \boldsymbol{\beta}, \quad i \in [\tilde{n}] \\ & z_i \geq -y_i + \beta_0 + \boldsymbol{x}_i^T \boldsymbol{\beta}, \quad i \in [\tilde{n}] \end{aligned} \tag{1}$$

- Approximate the constraint $g(\boldsymbol{x}) \leq 0$ with:

$$\beta_0 + \boldsymbol{\beta}^T \boldsymbol{x} \geq 0 \tag{2}$$

Step 3: MI representation of ML models → DTs

Decision Tree:

$$
\begin{aligned}
\sum_{j=1}^{|\mathcal{L}|} z_j p_j &= y_{\mathrm{DT}} \\
\sum_{j=1}^{|\mathcal{L}|} z_j &= 1 \\
\boldsymbol{a}_k^T \boldsymbol{x} &\leq b_k + M(1 - z_j), & \forall j \in \mathcal{L}, k \in L(L_j) \\
\boldsymbol{a}_k^T \boldsymbol{x} &\geq b_k - M(1 - z_j) + \epsilon, & \forall j \in \mathcal{L}, k \in R(L_j)
\end{aligned} \tag{3}
$$

- Approximate constraint $g(\boldsymbol{x}) \leq 0$ with:

$$y_{\mathrm{DT}} \geq 0.5$$

Step 3: MI representation of ML models → GBMs

GBTs:

- Train Gradient Boosted Trees on binary classification samples $D^i_{\mathcal{C}}$
- If GBM consists of K tree learners:

$$y_{\text{GBM}} = \sum_{i=1}^{K} w_i y^i_{DT}$$

- w_i: Weight of each learner
- y^i_{DT}: MI representation of the i-th Decision Tree
- Approximate constraint $g(\boldsymbol{x}) \leq 0$ with:

$$y_{\text{GBM}} \geq 0.5$$

Step 3: MI representation of ML models → MLPs

MLP: Nonlinear Constraints:

- Multi-layer perceptron with:
  - ReLU activations
  - Sigmoid activation on output neuron
- Train MLP on classification samples $D^i_{\mathcal{C}}$
- Model constraint $g(\boldsymbol{x}) \leq 0$ the following way:

$$\beta^L_{00} + \sum_{j \in N^{L-1}} \beta^L_{0j} v^{L-1}_j = y_{MLP} \geq 0$$

$$u^l_i \geq \beta^l_{i0} + \sum_{j \in N^{l-1}} \beta^l_{ij} v^{l-1}_j, \quad \forall l = \{2, \dots, L-1\}, i \in N^l$$

$$u^l_i \leq \beta^l_{i0} + \sum_{j \in N^{l-1}} \beta^l_{ij} v^{l-1}_j + M(1 - z_{il}), \quad \forall l = \{2, \dots, L-1\}, i \in N^l$$

$$u^l_i \leq M z_{il}, \quad \forall l = \{2, \dots, L-1\}, i \in N^l$$

$$u^l_i \geq 0, \quad u^l_i \geq 0, \quad \forall l = \{2, \dots, L-1\}, i \in N^l$$

$$u^1_i = x_i, \quad \forall i \in [n]$$

Step 4: Generate MI Approximation

- After training, we end up with the following MI approximation:

$$
\begin{aligned}
\min \quad & y_{\text{ML\_REGR}}(\boldsymbol{x}) \\
\text{s.t.} \quad & y^{i}_{\text{ML\_CLS}}(\boldsymbol{x}) \geq \tau_i, \ i \in \boldsymbol{I} \\
& \boldsymbol{Ax} \geq \boldsymbol{b}, \ \boldsymbol{Cx} = \boldsymbol{d} \\
& x_k \in [\underline{x}_k, \overline{x}_k], \ k \in [n]
\end{aligned}
$$

  where $\tau_i = 0$ for MLPs and SVMs and 0.5 otherwise
- Learners are approximate
- The approximation can be infeasible
- **Cause of many infeasibilities in OCTHaGOn**

Step 4: Generate MI Approximation

- Relax the constraints:

$$
\begin{aligned}
\text{min} \quad & y_{\text{ML\_REGR}}(\boldsymbol{x}) + \lambda \sum_{i \in I} u_i \\
\text{s.t.} \quad & y^{i}_{\text{ML\_CLS}}(\boldsymbol{x}) + u_i \geq t_i, \ i \in I \\
& \boldsymbol{Ax} \geq \boldsymbol{b}, \ \boldsymbol{Cx} = \boldsymbol{d} \\
& x_k \in [\underline{x}_k, \overline{x}_k], \ k \in [n] \\
& u_i \geq 0
\end{aligned}
$$

- Add a relaxation penalty λ in the objective
- Ideal scenario: $u_i = 0$

Step 5: Robust Optimization

- Model training $\rightarrow$ Sample-dependent
- Uncertainty in the trained model parameters
- We try to arrive at solutions $\boldsymbol{x}$ where models are more "certain" that the constraints are feasible
- We use RO in MI approximations, NOT in model training!

Step 5: Robust Optimization

Robustifying SVM:

- Original SVM constraint:

$$\bar{\beta}_0 + \bar{\boldsymbol{\beta}}^T \boldsymbol{x} \geq 0$$

- Robust SVM (Multiplicative Uncertainty):

$$\bar{\beta}_0 + (\bar{\boldsymbol{\beta}} \odot (\mathbf{1} + \boldsymbol{z}))^T \boldsymbol{x} \geq 0, \quad \forall \boldsymbol{z} : ||\boldsymbol{z}||_p \leq \rho$$

- Robust Counterpart:

$$\bar{\beta}_0 + \bar{\boldsymbol{\beta}}^T \boldsymbol{x} - \rho ||\bar{\boldsymbol{\beta}} \odot \boldsymbol{x}||_q \geq 0$$

Step 5: Robust Optimization

Robustifying Decision Trees (OCTs & GBMs):

- Original leaf constraint:

$$\bar{\boldsymbol{a}}_j^T \boldsymbol{x} \leq b_j + M(1 - z_i), \qquad \forall i \in \mathcal{L}, j \in L(L_i)$$
$$\bar{\boldsymbol{a}}_j \boldsymbol{x} \geq b_j - M(1 - z_i) + \epsilon, \qquad \forall i \in \mathcal{L}, j \in R(L_i)$$

- Robust Tree (Multiplicative Uncertainty):

$$(\bar{\boldsymbol{a}}_j \odot (\mathbf{1} + \boldsymbol{u}))^T \boldsymbol{x} \leq b_j + M(1 - z_i), \quad \forall \boldsymbol{u} : ||\boldsymbol{u}||_p \leq \rho, \quad \forall i \in \mathcal{L}, j \in L(L_i)$$
$$(\bar{\boldsymbol{a}}_j \odot (\mathbf{1} + \boldsymbol{u}))^T \boldsymbol{x} \geq b_j - M(1 - z_i) + \epsilon, \quad \forall \boldsymbol{u} : ||\boldsymbol{u}||_p \leq \rho, \quad \forall i \in \mathcal{L}, j \in R(L_i)$$

- Robust Counterpart:

$$\bar{\boldsymbol{a}}_j^T \boldsymbol{x} + \rho ||\bar{\boldsymbol{a}}_j \odot \boldsymbol{x}||_q \leq b_j + M(1 - z_i), \quad \forall i \in \mathcal{L}, j \in L(L_i)$$
$$\bar{\boldsymbol{a}}_j^T \boldsymbol{x} - \rho ||\bar{\boldsymbol{a}}_j \odot \boldsymbol{x}||_q \geq b_j - M(1 - z_i) + \epsilon, \quad \forall i \in \mathcal{L}, j \in R(L_i)$$

Step 6: Solve MIO and Improve

$$
\begin{aligned}
\min \quad & y_{\text{ML\_REGR}}(\boldsymbol{x}) + \lambda \sum_{i \in I} u_i \\
\text{s.t.} \quad & y^i_{\text{ML\_CLS}}(\boldsymbol{x}) + u_i \geq t_i, \; i \in I \\
& \boldsymbol{A}\boldsymbol{x} \geq \boldsymbol{b}, \; \boldsymbol{C}\boldsymbol{x} = \boldsymbol{d} \\
& x_k \in [\underline{x}_k, \overline{x}_k], \; k \in [n] \\
& u_i \geq 0
\end{aligned}
$$

- Solve the resulting MIO and obtain approximate solution $\boldsymbol{x}^*$.
- Try to ensure local optimality with Projected Gradient Descent (PGD) steps

Recap: Solution Process

Solution process:

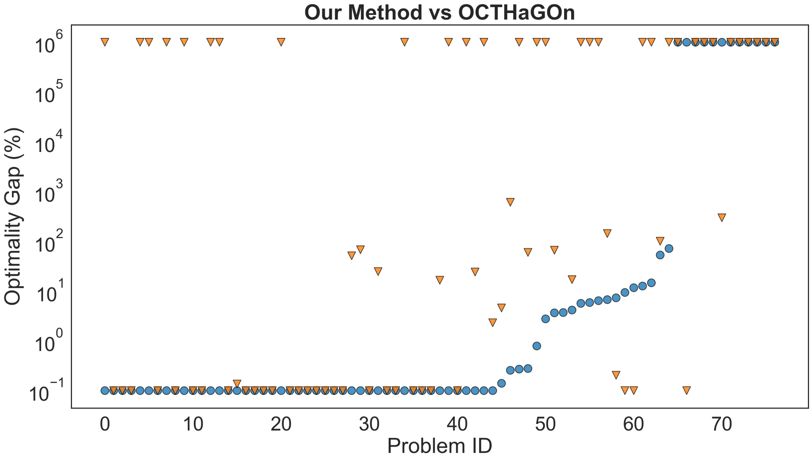
1. Sample non-linear constraints:
   - Additionally use OCT-based sampling
2. Train ML models to approximate nonlinear constraints/objective:
   - Use GBMs, MLPs, SVMs besides OCTs
3. Generate MI approximation:
   - Relax the constraints to account for infeasibilities
4. Robustify final approximation
5. Solve the MI approximation and repair using PGD

Outline

Results

- Run the framework on bounded & continuous non-convex problems
- Use 77 problems from MINLP library
- Try multiple hyperparameters & choose best solution:
  - Relaxation penalty λ: $\lambda = 10^2, \lambda = 10^4, \lambda = \infty$ (no relax.)
  - Robustness radius $\rho = 0, 0.01, 0.1, 1$
  - **Only need to sample & train models once!**
- Compare against:
  - **OCHaGOn**: Work we are extending.
  - **Gurobi 10**: General Constraints allow for approximations of some non-convex functions.
  - **BARON**: Best commercial optimizer for Global Optimization.

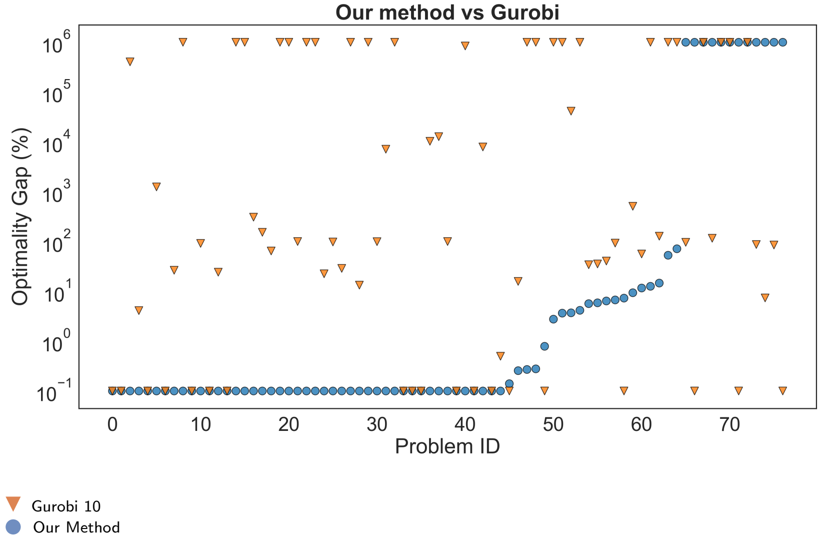
Results: Comparing with OCTHaGOn



OCTHaGOn

Our Method

Results: Our method VS Gurobi with General Constraints



Results: Our method VS BARON

- BARON is very good at the MINLPLib problems:
  - Can solve the vast majority to optimality (73 out of 77).
  - MINLPLib & BARON have been around for 25+ years.
  - Development of BARON has long been influenced by MINLPLib.
- Still, our method:
  - Better solutions than BARON in **3** out of the **77** instances.
  - Better time than BARON in **4** out of the **77** instances.
- All these problems are compatible with BARON:
  - Our method can work on more general problems.

Outline

Conclusion

- Enhanced framework:
  - Significant improvement over OCTHaGOn
  - Handles nonconvexities much better than Gurobi 10.0 General Constraints
  - Better solutions than BARON in 3 instances
- Benefits: It is compatible:
  - Constraints with very general primitives
  - Black-box constraints
  - Data-driven constraints
- Disadvantages:
  - Approximate method: No guarantees of optimality
  - Solutions returned may not be optimal or feasible