Selecting convex relaxations in a branch-and-bound algorithm using machine learning

Can Li*, Andrea Lodi†

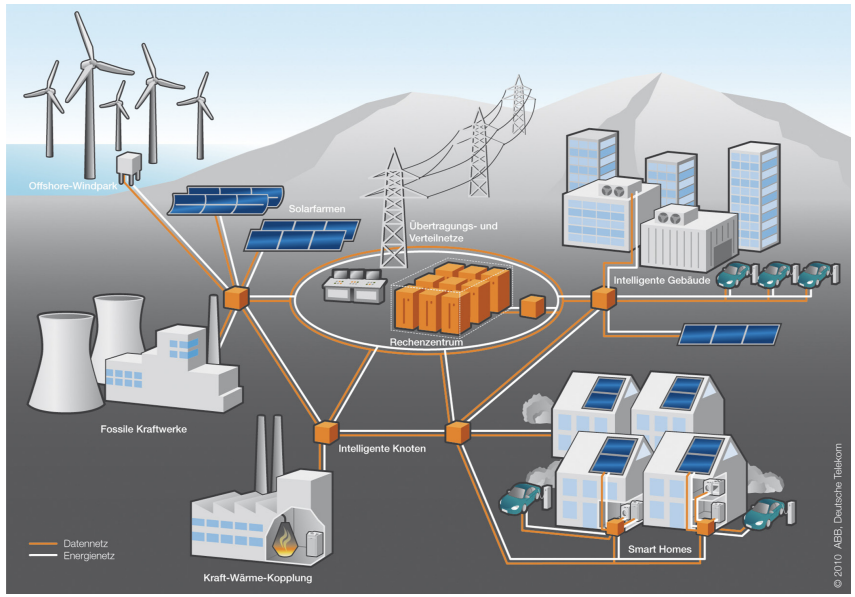
Purdue University *, USA
Cornell Tech and Technion- IIT†, USA

2023 Third Sevilla MINLP Workshop

Motivation

It is common to repeatedly solve similar optimization problems in practice.

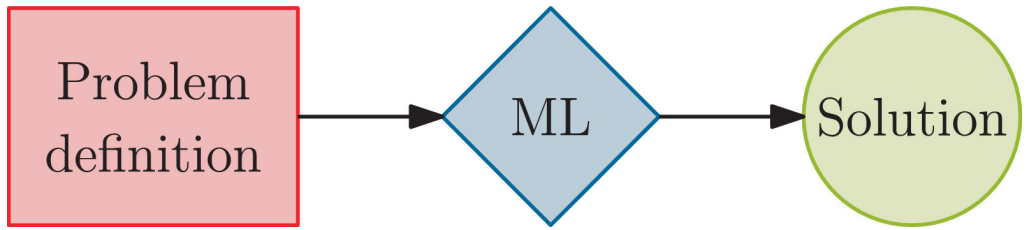
- Power producers repeatedly solve unit commitment (UC) problems to meet power demand at minimum market cost. Xavier et al., 2021



Use machine learning to make decisions when solving similar optimization problems.

End to end learning[1]

Train the machine learning model to output solutions directly from the input instance

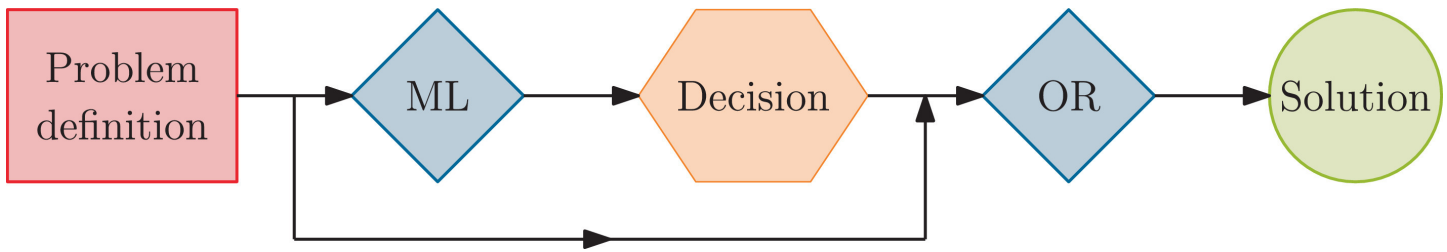


- W. Chen et al. (2022). Learning optimization proxies for large-scale security-constrained economic dispatch.
- E. Khalil et al. (2017). Learning combinatorial optimization algorithms over graphs.

[1] Y. Bengio et al. (2021). Machine learning for combinatorial optimization: a methodological tour d'horizon.

Learning to configure algorithms

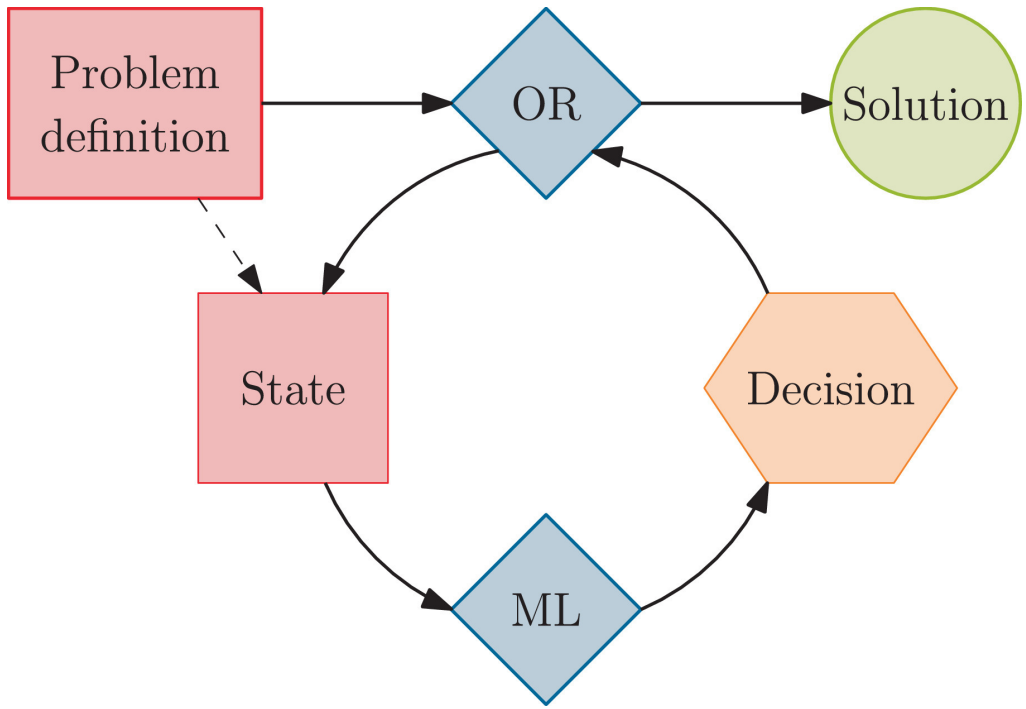
Machine learning can provide a parametrization of the algorithm



- P. Bonami et al. (2022). A classifier to decide on the linearization of mixed-integer quadratic problems in CPLEX.
- M. Kruber et al. (2017). Learning when to use a decomposition.

Machine learning alongside optimization algorithms

Calling an ML method to make decisions within an optimization algorithm



- S. Zeng et al. (2022). A reinforcement learning approach to parameter selection for distributed optimal power flow.
- M. Gasse et al. (2019b). Exact combinatorial optimization with graph convolutional neural networks.

ML for MILP: a growing field

Node selection

- [He et al., 2014]
- [Song, Lanka, Zhao, et al., 2018]

Variable selection

- [E. B. Khalil, Le Bodic, et al., 2016]
- [Hansknecht et al., 2018]
- [Balcan et al., 2018]
- [Gasse et al., 2019a]
- [Gupta et al., 2020]
- [Nair et al., 2020]

Cutting planes selection

- [Baltean-Lugojan et al., 2018]
- [Tang et al., 2019]

Primal heuristic selection

- [E. B. Khalil, Dilkina, et al., 2017]
- [Hendel et al., 2018]

Formulation selection

- [Bonami et al., 2018]

Neighborhood search heuristics

- [Ding et al., 2019]
- [Song, Lanka, Yue, et al., 2020]
- [Addanki et al., 2020]

Diving heuristics

- [Song, Lanka, Zhao, et al., 2018]
- [Yilmaz et al., 2020]
- [Nair et al., 2020]

Mixed-Integer Nonlinear Program (MINLP)

$$\begin{aligned} \arg\min_{x} \quad & f(x) \\ \text{subject to} \quad & g(x) \leq 0, \\ & l \leq x \leq u, \\ & x \in \mathbb{Z}^p \times \mathbb{R}^{n-p}. \end{aligned}$$

- $f : \mathbb{R}^n \to \mathbb{R}$ the objective function
- $g : \mathbb{R}^n \to \mathbb{R}^m$ the constraint coefficient matrix
- f, g can be nonconvex
- $l, u \in \mathbb{R}^n$ the lower and upper variable bounds
- $p \leq n$ integer variables

Wide applications of global optimization

Energy systems optimization

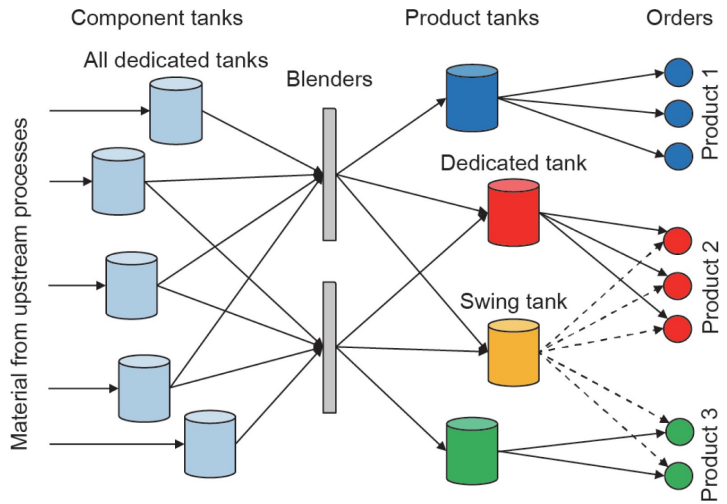
- AC optimal power flow
- Pooling problem

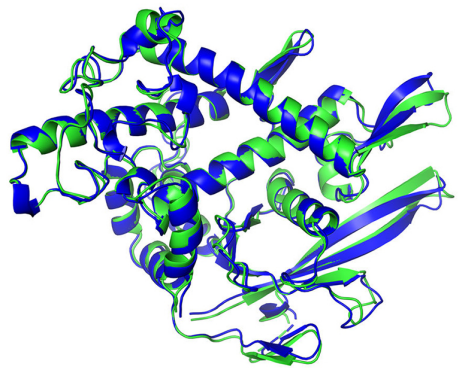
Chemical engineering

- Heat exchanger design
- Design of molecules

Systems biology

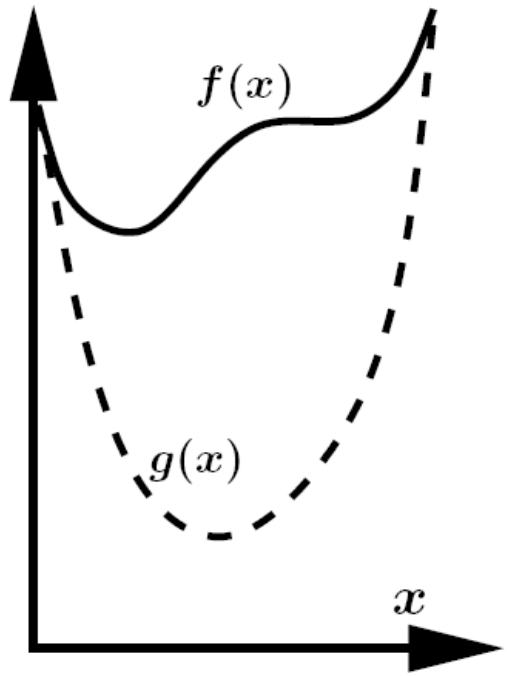
- Protein structure prediction
- Metabolic engineering





Convex relaxations for nonconvex functions

Convex function $g(x)$ underestimate nonconvex function $f(x)$

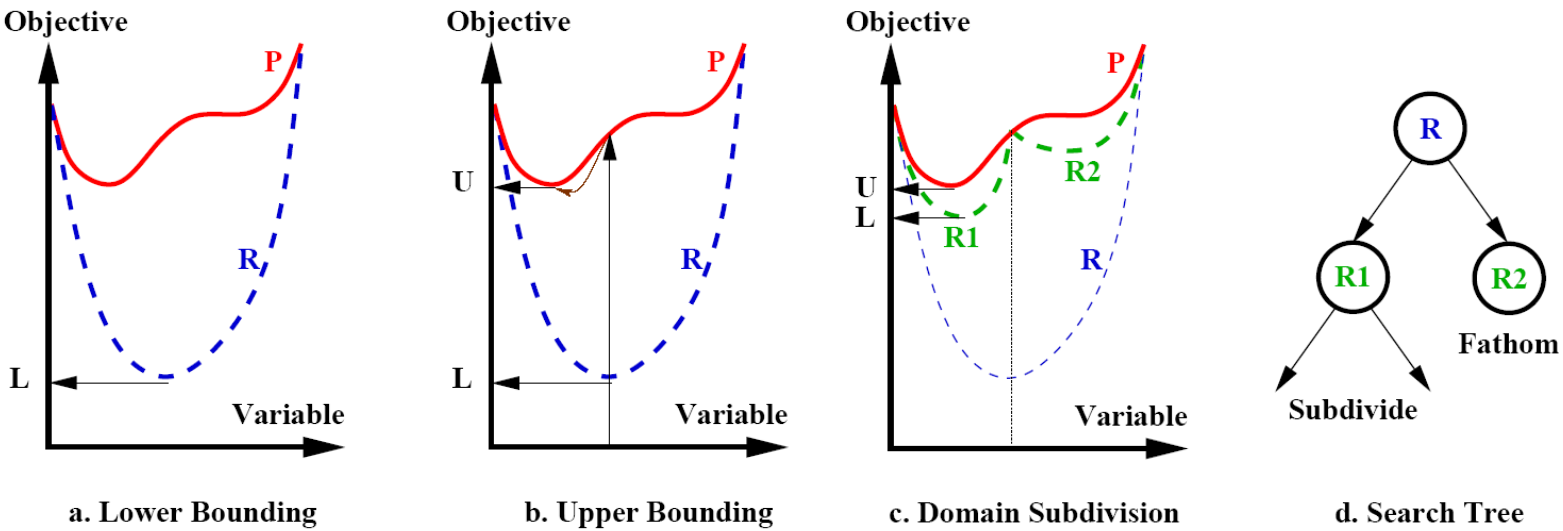


Example: quadratic functions $x^\top Qx$, Q not p.s.d

- McCormick inequalities
- Reformulation linearization technique (RLT)
- Semidefinite programming (SDP)

Spatial branch-and-bound

Need to perform spatial branching on the continuous variable to obtain global optimality



Machine learning for global optimization

Spatial branching

- improve the standard spatial branching rules [Ghaddar et al., 2022]

Cut selection

- select sparse SDP cuts [Baltean-Lugojan et al., 2019]

Initial formulation

- whether to linearize an MIQP or not [Bonami et al., 2022]

Learning to select relaxations in a branch and bound algorithm

- This talk.

Quadratic Unconstrained Binary Optimization(QUBO)

$$\min x^\top Q x$$

$$x \in \{0,1\}^n$$

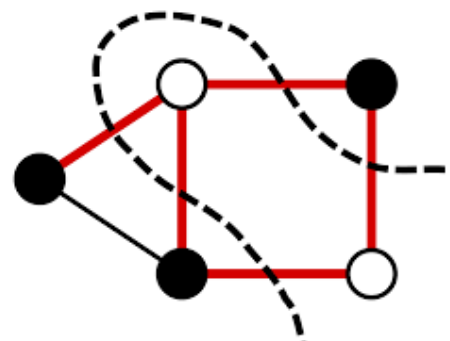
A number of applications can be represented as QUBOs.

- Maxcut
- Quadratic Stable Set Problem
- Graph Coloring
- Partition Problem

Maxcut formulation

Given graph $G = (V, E)$ & matrix $W \in S^n$ with $w_{i,j} \neq 0$ if $(i,j) \in E$ & 0 otherwise

$$f^* = \max_{x \in \{-1,+1\}^n} \sum_{(i,j) \in E} w_{i,j} \left(\frac{1 - x_i x_j}{2} \right)$$



Branch-and-bound Algorithm for QUBO

- In order to solve QUBO to global optimality, a branch-and-bound algorithm is needed.
- **Lower bound**
  Different relaxations can be used in the branch-and-bound algorithm
  1. Linear Programming (LP)
  2. Semidefinite programming (SDP)
  3. Hybrid LP & SDP
- **Upper bound**
  Heuristics like Goemans and Williamson (1995)
- We will focus on how to effectively use lower bound in this talk

LP relaxations

Let $X \in S^n$ represents $xx^\top$. The QUBO can be relaxed as

$$\min_{X, x \in [0,1]} \quad < Q, X >$$

$$\texttt{s.t.} \quad \texttt{valid linear inequalities}$$

- McCormick inequalities

$$X_{ij} \geq 0, \quad X_{ij} \geq x_i + x_j - 1, \quad X_{ij} \leq x_i \quad \forall i, j$$

- Odd cycle inequalities derived from the CUT polytope, e.g., triangle inequalities.

$$\left.\begin{aligned} \chi_{ij} + \chi_{ik} + \chi_{jk} &\leq 2 \\ \chi_{ij} - \chi_{ik} - \chi_{jk} &\leq 0 \\ -\chi_{ij} + \chi_{ik} - \chi_{jk} &\leq 0 \\ -\chi_{ij} - \chi_{ik} + \chi_{jk} &\leq 0 \end{aligned}\right\} \text{ for all distinct } i, j, k \in V$$

where $\chi_e \in \{0,1\}^{|E|}$ whether edge e is in the cut. $\chi_{ij} = x_i + x_j - X_{ij} - X_{ji}$

SDP relaxations

$$X = xx^\top \overset{\text{relax}}{\Longrightarrow} X \succeq xx^\top \overset{\text{Schur complement}}{\Longleftrightarrow} \begin{bmatrix} 1 & x^\top \\ x & X \end{bmatrix} \succeq 0$$

- Basic SDP relaxation of QUBO can be solved by interior point solvers efficiently
- For large problems, basic SDP relaxation bounds improve slowly in the branch and bound algorithm.

SDP + linear relaxations

- Stronger than SDP or linear relaxations alone. Reduce the number of nodes in branch-and-bound.
- State-of-the-art QUBO solvers, e.g., BiqMac, BiqCrunch, use SDP + polyhedral relaxations.
- Stronger but also more expensive to solve.

Motivations of our work

Trade-offs in applying different relaxations

- LP relaxation
  - easy to solve; weak bounds
  - Enumerate more nodes in the same amount of time
- SDP + LP relaxations
  - stronger bounds; expensive to solve
  - Higher chance of fathoming a node.

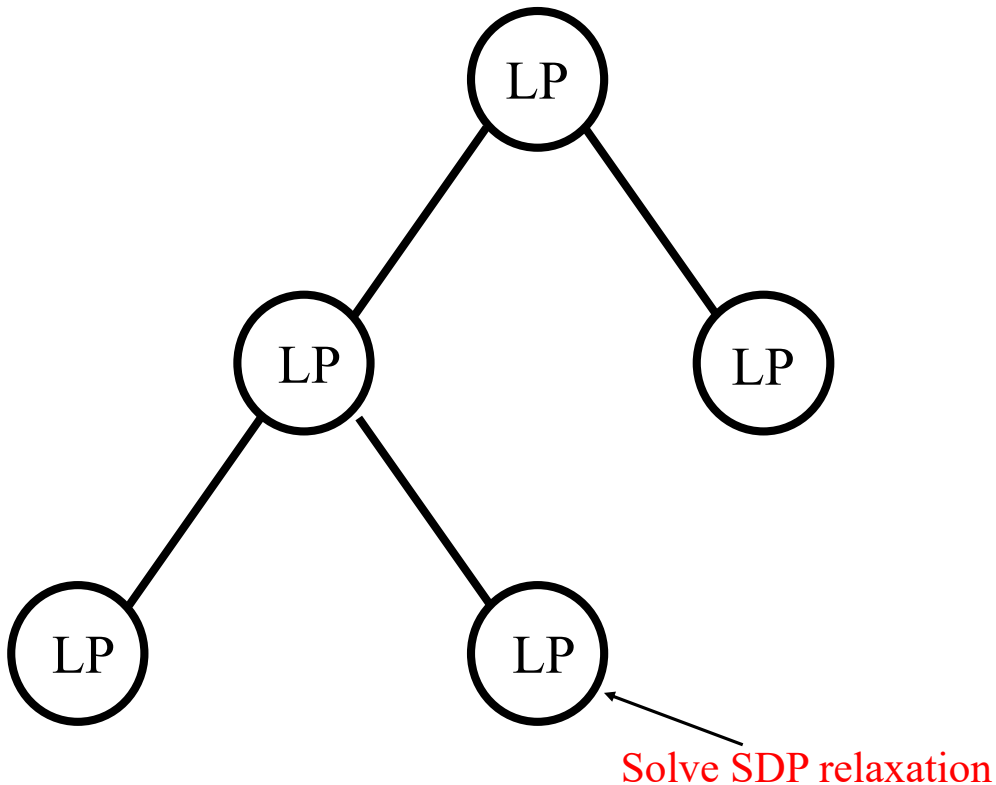
Questions we aim to address

- Which relaxation should we use at each node of the branch-and-bound algorithm?
- Is it worth solving a more expensive relaxation (SDP-based relaxation) at each node?

Hybrid bounding scheme

We adopt the framework of Furini and Traversi (2013).

- By default, we solve the LP relaxation of the problem at each node.
- The SDP relaxation is only solved if we believe a node is likely to be pruned by SDP bound.



Simple setting

McCormick relaxation v.s. McCormick relaxation + SDP

- At each node, we solve the LP relaxation with McCormick inequalities.

$$(LP) \quad \min_{X, x \in [0,1]} \quad < Q, X >$$
$$\texttt{s.t.} \quad X_{ij} \geq 0, \quad X_{ij} \geq x_i + x_j - 1, \quad X_{ij} \leq x_i \quad \forall i, j$$

- The SDP relaxation is solved conditionally.

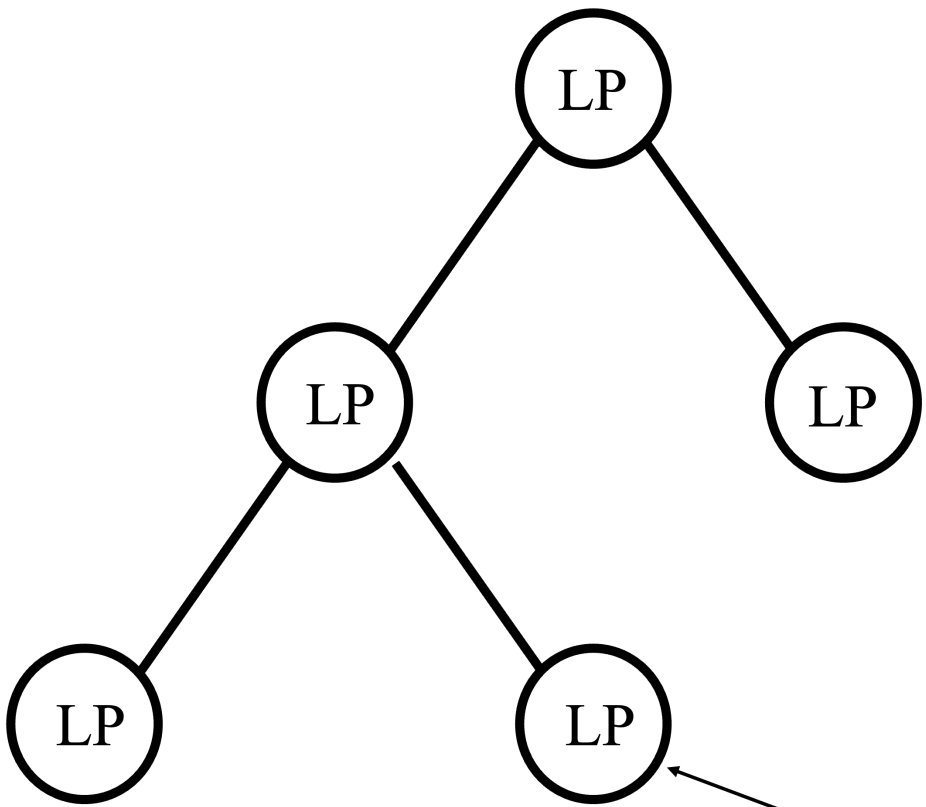
$$(SDP) \quad \min_{X, x \in [0,1]} \quad < Q, X >$$
$$\texttt{s.t.} \quad X_{ij} \geq 0, \quad X_{ij} \geq x_i + x_j - 1, \quad X_{ij} \leq x_i \quad \forall i, j$$
$$\begin{bmatrix} 1 & x^\top \\ x & X \end{bmatrix} \succeq 0$$

Predicting SDP bound using Heuristic/Machine learning

- If we can predict whether the SDP bound can fathom a node without solving an SDP using a heuristic/machine learning, the computational time can be saved for the nodes that cannot be fathomed.
- This work: *learning to predict bounds*

A simple observation

- If a feasible solution of the SDP relaxation can be found that is smaller than the incumbent solution, the SDP bound cannot fathom and node. We can skip solving the SDP.
- How to quickly find a near optimal solution to the SDP?

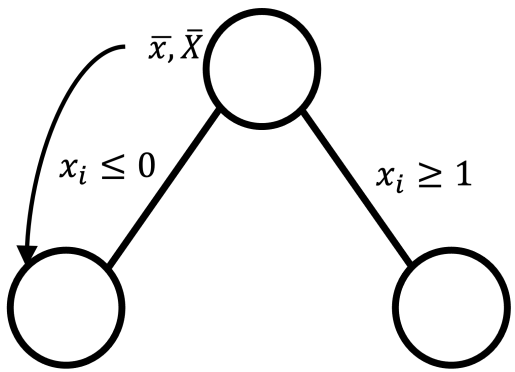


$\overline{LB} \geq LB(SDP), \overline{LB} \leq UB$

The SDP bound cannot fathom the node

QCP-based approach

- Suppose SDP is solved at a node with optimal solution $\bar{x}, \bar{X}$. We solve the child nodes with x, X fixed to $\bar{x}, \bar{X}$ except the ith row and column.



$$\min < Q, X >$$

$$X_{ij} \geq 0, \quad X_{ij} \geq x_i + x_j - 1, \quad X_{ij} \leq x_j \quad \forall j$$

$$0 \leq x_i \leq 1$$

$$\begin{bmatrix} X_{i,i} & X_{[i,1:i-1]} & x_i & X_{[i,i+1:]} \\ X_{[1:i-1,i]} & \bar{X}_{1:i-1,1:i-1} & \bar{x}_{1:i-1} & \bar{X}_{[1:i-1,i+1:]} \\ x_i & \bar{x}_{1:i-1}^\top & 1 & \bar{x}_{i+1:}^\top \\ X_{[i+1:,i]} & \bar{X}_{[i+1:,1:i-1]} & \bar{x}_{i+1:} & \bar{X}_{[i+1:,i+1:]} \end{bmatrix} \succeq 0$$

- The SDP can be reformulated as a convex QCP by taking the Schur complement.

QCP-based approach

$$\min < Q, X >$$

$$X_{ij} \geq 0, \quad X_{ij} \geq x_i + x_j - 1, \quad X_{ij} \leq x_j \quad \forall j$$

$$0 \leq x_i \leq 1$$

$$\begin{bmatrix} X_{i,i} & X_{[i,1:i-1]} & x_i & X_{[i,i+1:]} \\ X_{[1:i-1,i]} & \bar{X}_{1:i-1,1:i-1} & \bar{x}_{1:i-1} & \bar{X}_{[1:i-1,i+1:]} \\ x_i & \bar{x}_{1:i-1}^\top & 1 & \bar{x}_{i+1:}^\top \\ X_{[i+1:,i]} & \bar{X}_{[i+1:,1:i-1]} & \bar{x}_{i+1:} & \bar{X}_{[i+1:,i+1:]} \end{bmatrix} \succeq 0$$

Equivalent to a convex QCP with $\mathcal{O}(n)$ variables and constraints

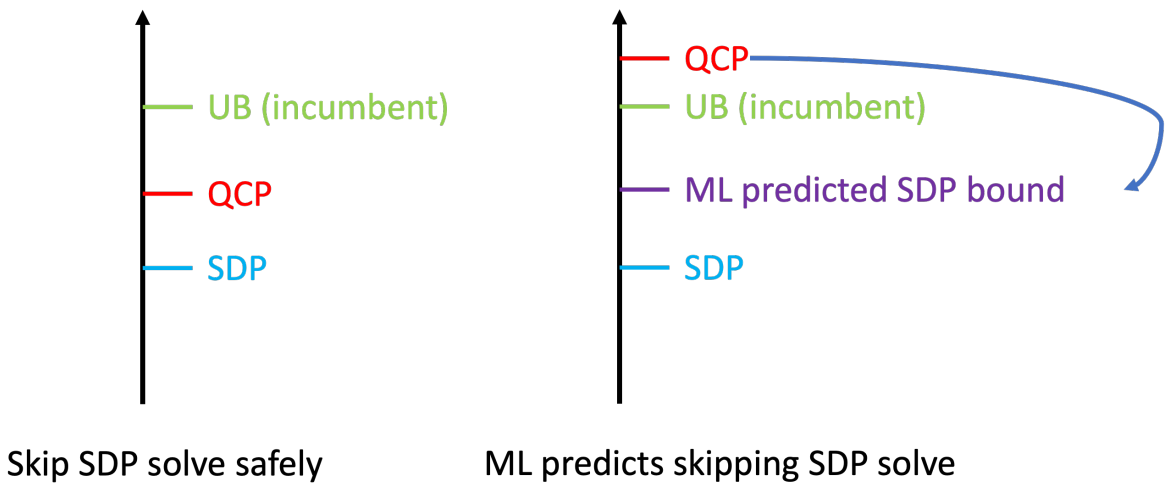
$$X_{i,i} - \begin{bmatrix} X_{[1:i-1,i]} \\ x_i \\ X_{[i+1:,i]} \end{bmatrix}^\top \bar{X}^{-1} \begin{bmatrix} X_{[1:i-1,i]} \\ x_i \\ X_{[i+1:,i]} \end{bmatrix} \geq 0$$

Combine QCP with Machine learning

- If the QCP value is smaller than the incumbent solution, we can safely skip the SDP solve.
- Can we be more aggressive in skipping the nodes?

Estimating the "true" SDP bounds using ML

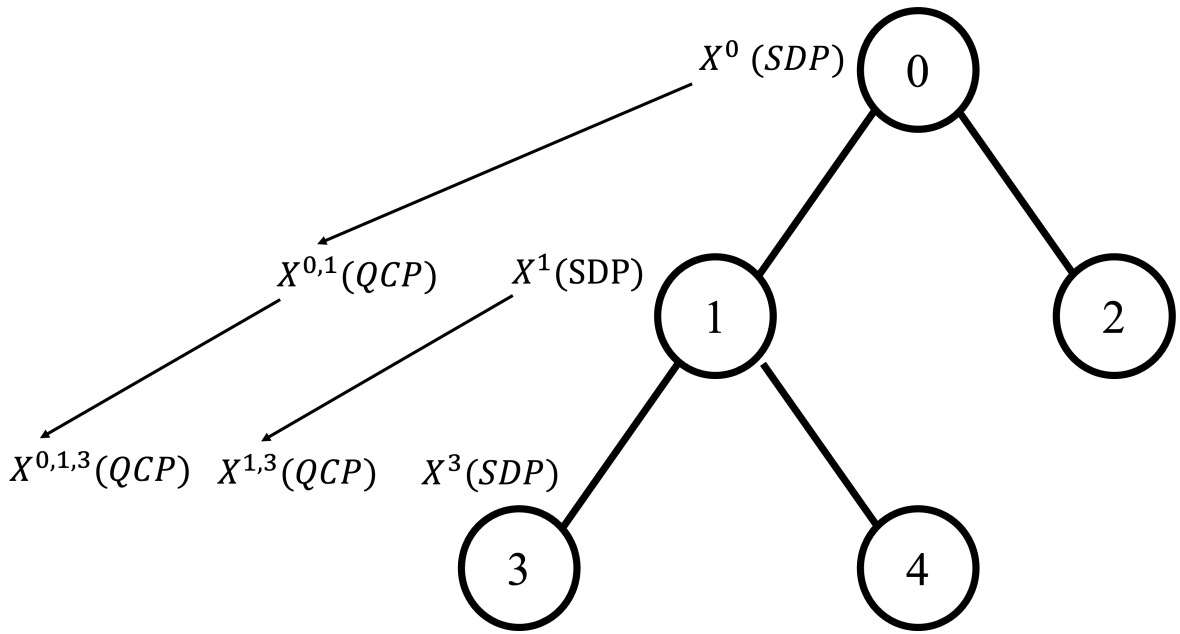
- Idea: we can estimate the "suboptimality" of QCP solve from data
- If we have an accurate estimation of the suboptimality, we might skip the SDP solve even if the QCP value is greater than the incumbent.



Skip SDP solve safely

ML predicts skipping SDP solve

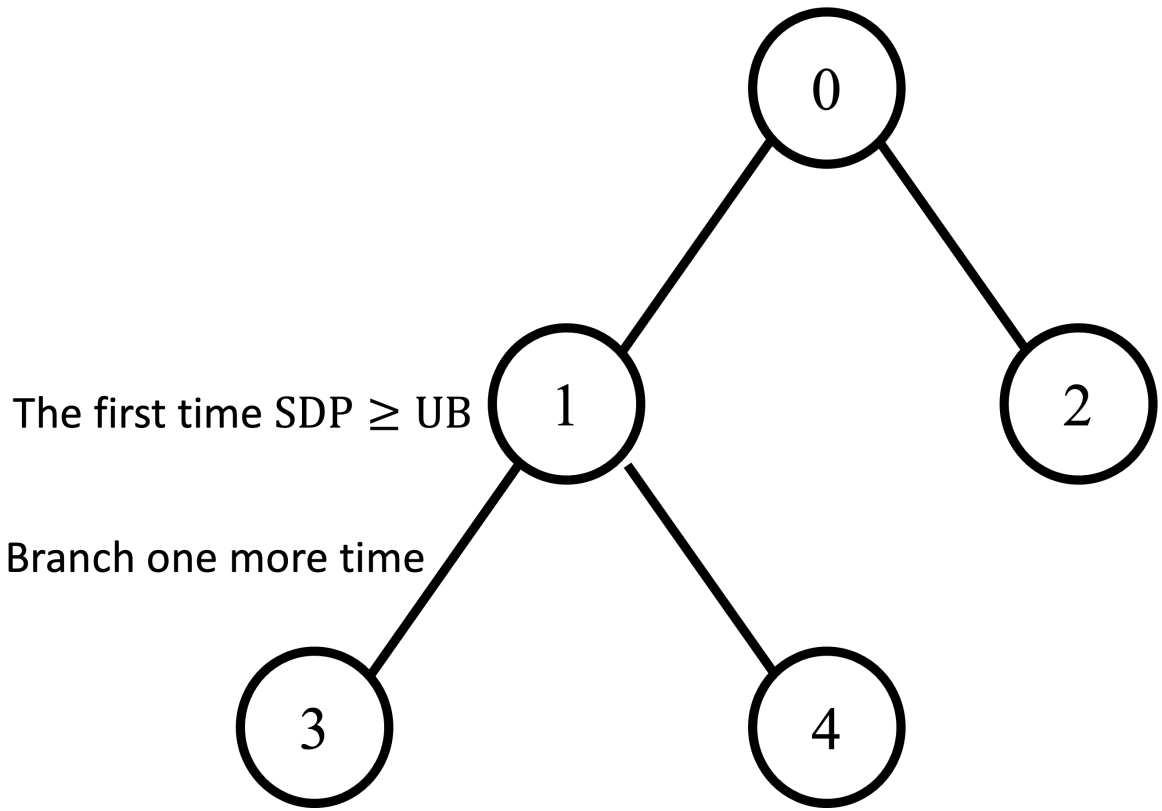
Data collection



- SDP is solved at each node
- QCP is solved based on the SDP/QCP solves of its parent node
- `napprox` denotes the number of QCP approximations made
- For $X^{0,1,3}$, `napprox = 2`. For $X^{1,3}$ and $X^{0,1}$, `napprox = 1`
- we only consider `napprox`≤ 3

Data collection continued

- Since we can make false predictions, We branch one more time after the node can be fathomed by SDP bound during data collection.
- The training data is collected on 36 maxcut instances of size 60 with densities ranging from 10% to 90%.



Feature engineering

Table: Features for each QCP solved

obj_qcp - obj_fea	QCP and incumbent solution difference
obj_fea - obj_sdp_root	incumbent solution and root node SDP difference
depth	node depth
napprox	number of QCP approximations
geo_mean	geometric mean of number of branches on 1 and 0
density	density of the maxcut adjacency matrix

SVM classifier

- With the features, we predict whether a node can be fathomed by SDP bound or not. (0/1 labels)
- We use SVC from scikit-learn that implements a Support Vector Machine (SVM) classifier with radial basis function (rbf) kernel.

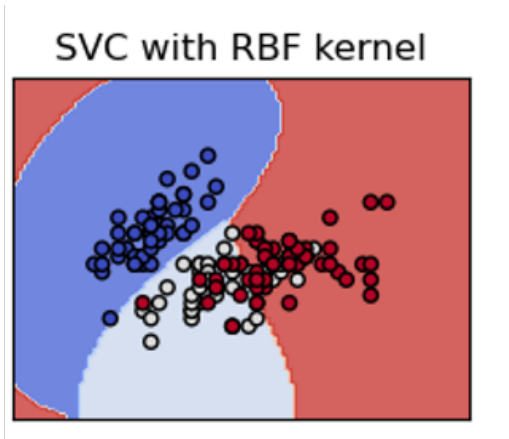
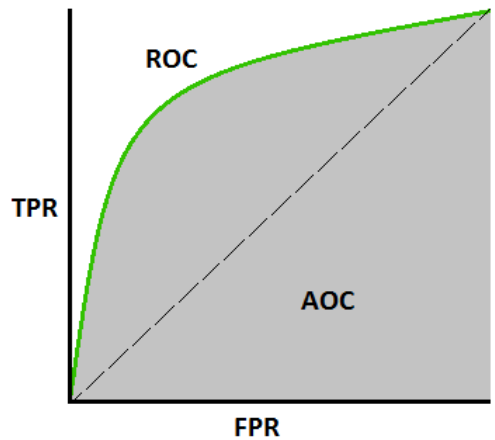


Table: Prediction statistics

	Accuracy	Precision	Recall
SVM	93.58%	94.93%	95.15%

TPR v.s. FPR

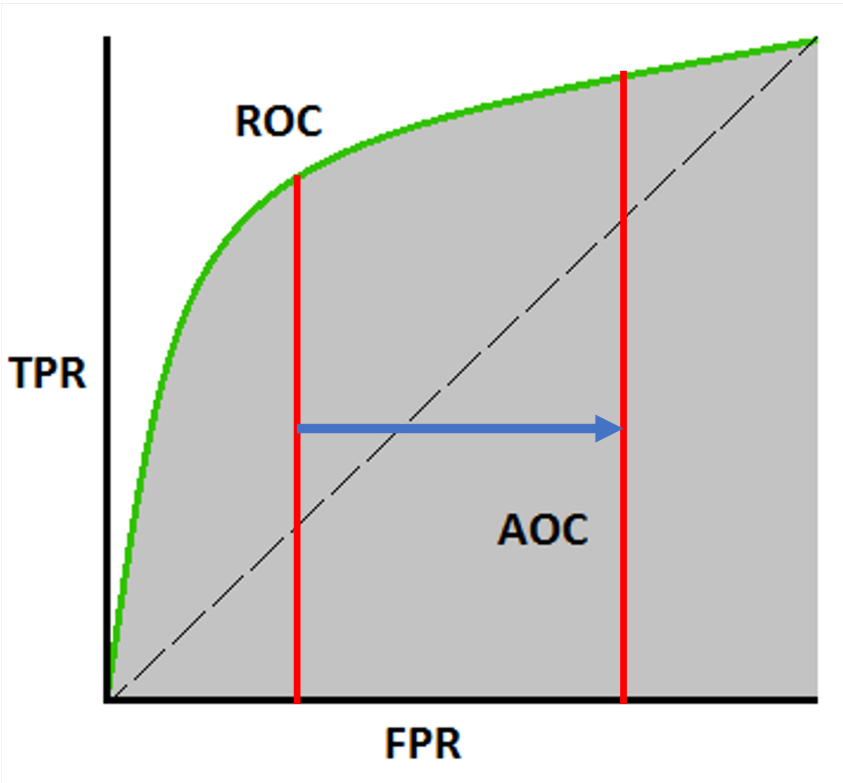
- TPR (recall): true positive rate
  $$\frac{\#\text{ nodes predicted to be fathomed by SDP that is true}}{\#\text{node that can be fathomed by SDP}}$$
- FPR: false positive rate
- We want to increase recall such that we do not miss many nodes that can be fathomed by SDP bounds. Otherwise, more nodes will be created and have to be solved.



TPR v.s. FPR

move the decision boundary to have higher recall

- Set the decision boundary threshold such that $8TPR - FPR$ is maximized



Testing results

- Increasing the recall also decreases accuracy and precision

Table: Prediction statistics

	Accuracy	Precision	Recall
SVM	93.58%	94.93%	95.15%
SVM with high recall	92.40%	90.30%	98.86%

Numerical results

- Average over 9 maxcut instances with densities 10%-90%
- The reduction in nodes and time can be different
- QCP SVM with high recall performs the best

Table: Node and time reduction compared with solving SDP at each node

	QCP	QCP SVM	QCP SVM high recall
# SDP solve reduction	19%	26%	28%
walltime reduction	16%	8%	20%

Conclusions & Future work

- We have proposed a ML-based framework to predict SDP bounds in solving QUBO
- Future work can be focused on implementation in state-of-the-art solvers like BiqMac, BiqCrunch, BiqBin, MADAM.