Machine Learning for Discrete Optimization: an Intro

Yifeng Xiao

September 20, 2023

Content

- Motivation and Compromise
- Algorithm Acceleration
  - Continuous Case
  - Discrete Case
- Objective Prediction

Motivation and Compromise

End-to-end learning

Let a learnable large model act as a solver (for different instances within one problem / different problems), which receives problem contexts and outputs optimal solutions.

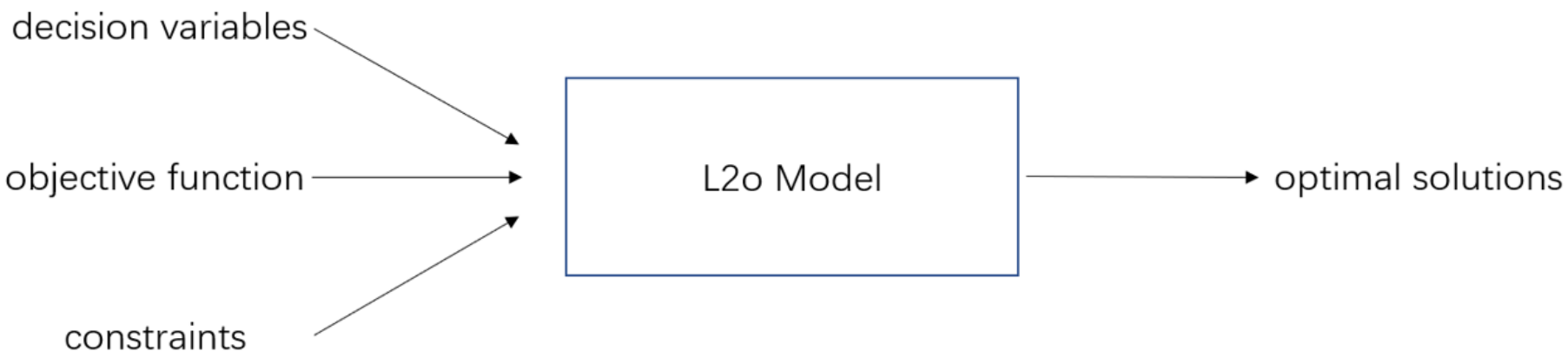

Figure: An end-to-end solver

Motivation and Compromise

Bipartite graph

Bipartite graphs can represent linear programming problems without information loss.

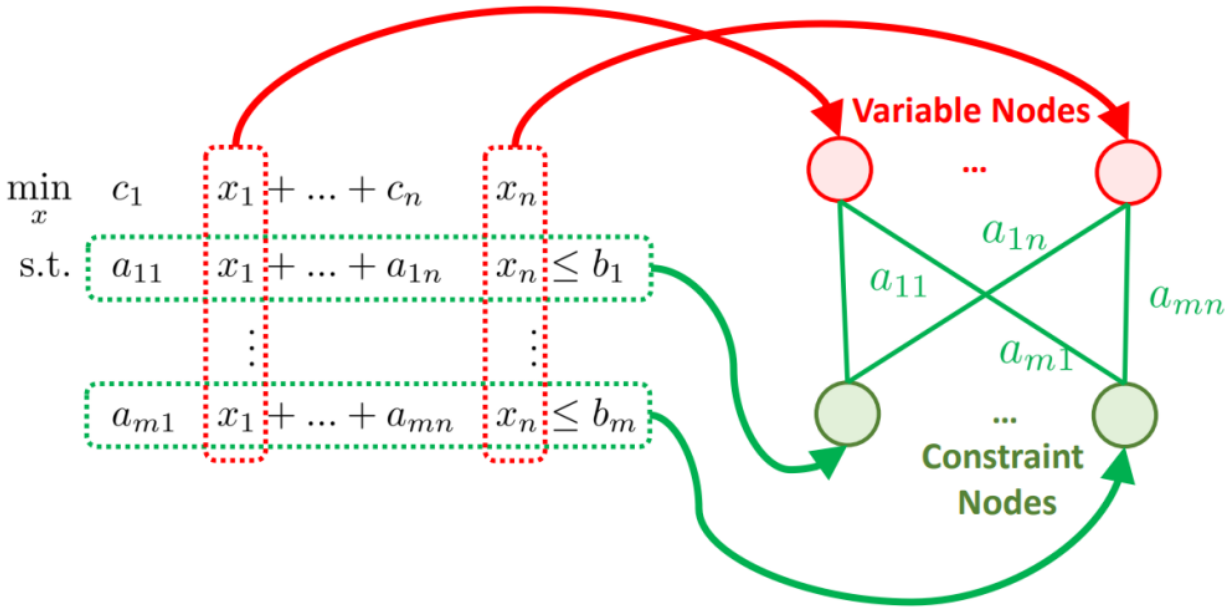

Figure: Bipartite graph

Motivation and Compromise

Obstacles

- Dimensionality
- Feasibility
- Ill-posed mapping

Motivation and Compromise

Solution distribution learning [NBG+20]

To avoid the problems feasibility, we use GNNs to predict $P(x_{i,j} = 1)$.

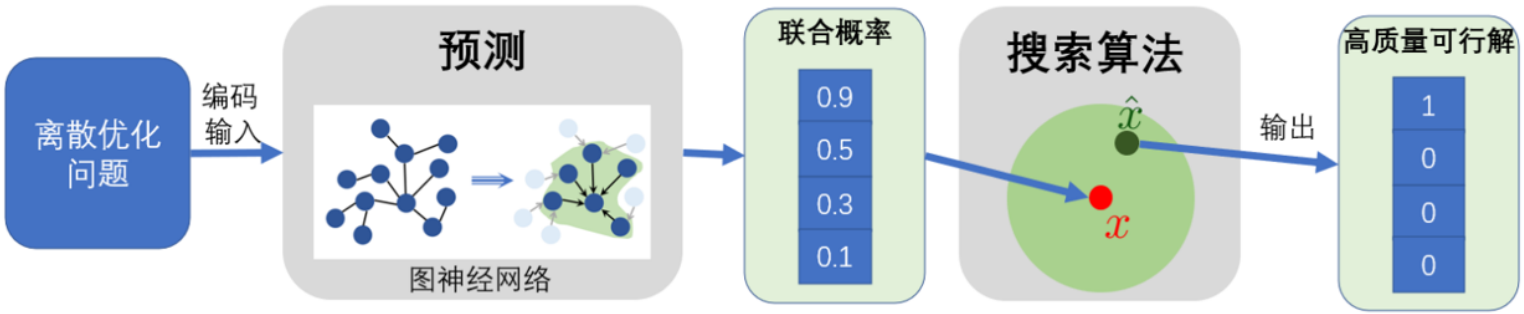

Figure: Neural diving

Algorithm Acceleration

Using network to make pivital and hard decision in traditional algorithm:

- branch and bound
- local neighbor search
- cutting plane

Algorithm Acceleration

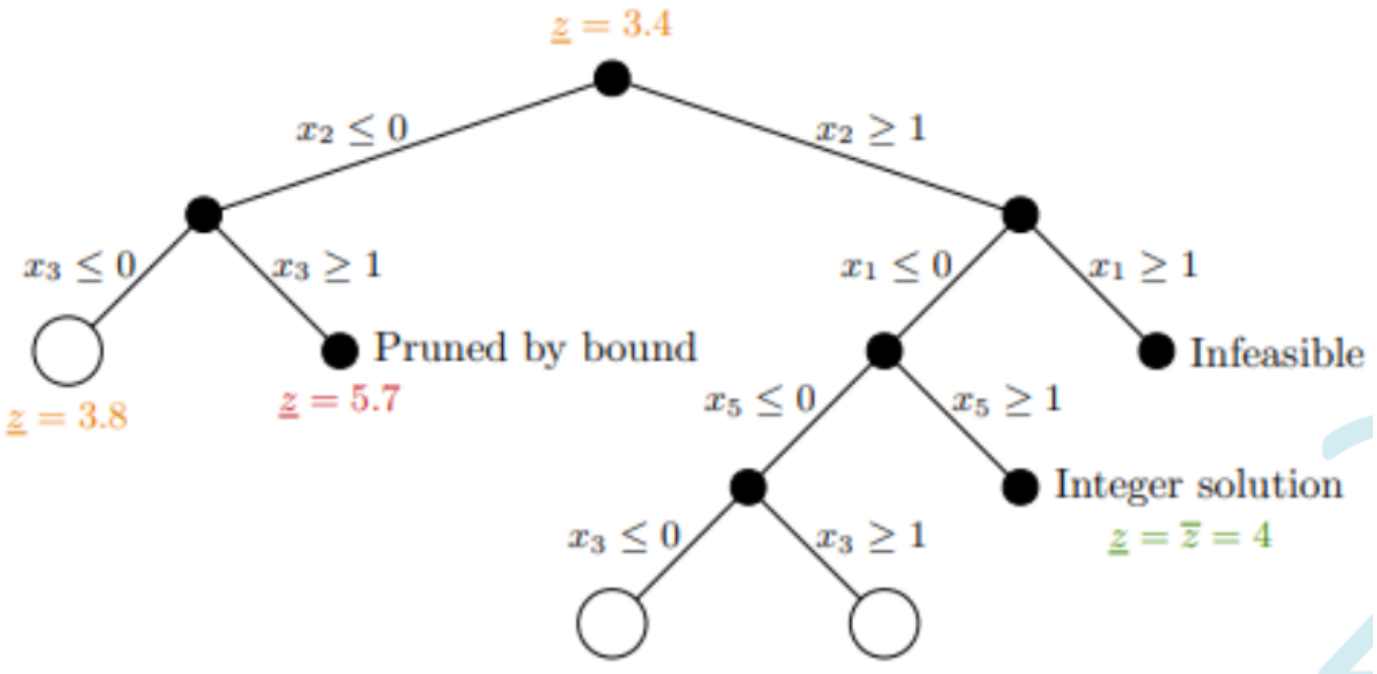

Figure: Branch and bound

node + bound $\longrightarrow$ Network $\longrightarrow$ which variable to destroy

Algorithm Acceleration

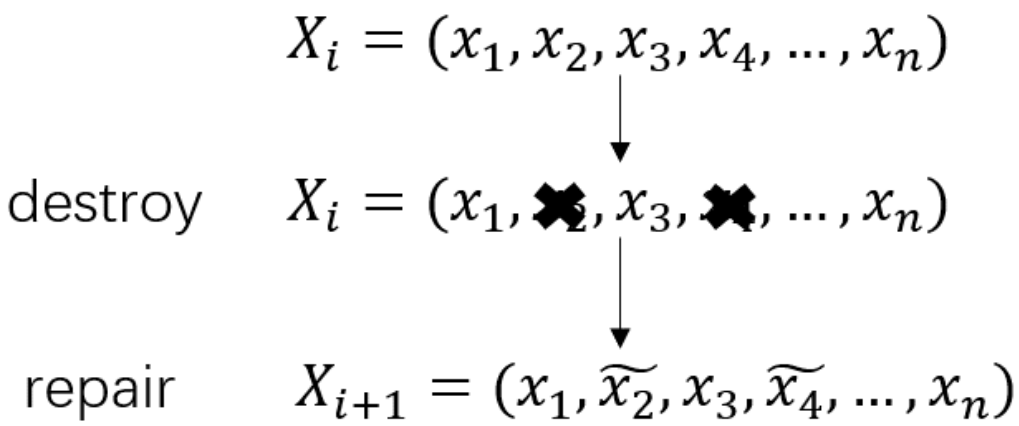

Figure: Local neighbor search

solution + current problem $\longrightarrow$ Network $\longrightarrow$ which variable to branch

Algorithm Acceleration

Cutting plane

The branch-and-bound can be regarded as a special cutting plane method (cutting plane in B&B is $x_i \leq n$, $x_i \geq n$). Actually, we can add more efficient cutting planes (more variables involved).

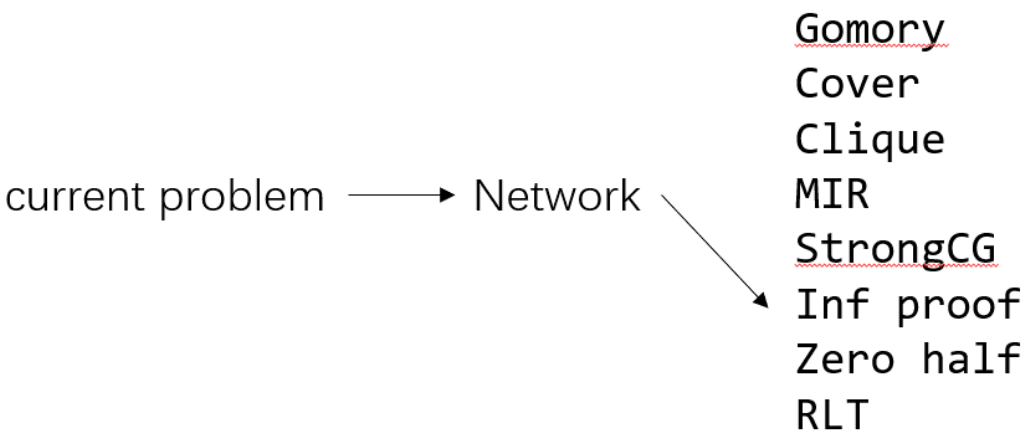

Figure: Learning to cut

Objective Prediction

Solutions
- high-dimensionality
- feasibility issue
- ill-posed nature

Objective
- single value
- no constraint
- continuous mapping

problem context $\longrightarrow$ Network $\longrightarrow$ objective value

Objective Prediction

Problem formulation:

$$\begin{aligned} \min_{x,y} \quad & f_1(x) + f_2(y) \\ \text{s.t.} \quad & g(x) \leq 0 \\ & g_1(x) + g_2(y) \leq 0 \end{aligned} \tag{1}$$

Objective prediction network:

$$\Phi(x) = \min_{y: g_1(x)+g_2(y)\leq 0} f_2(y) \tag{2}$$

Surrogate problem:

$$\begin{aligned} \min_{x,y} \quad & f_1(x) + \Phi(x) \\ \text{s.t.} \quad & g(x) \leq 0 \end{aligned} \tag{3}$$

Objective Prediction

Problem formulation:

$$\begin{aligned} \min_{x,y} \quad & f_1(x) + f_2(y) \\ \text{s.t.} \quad & g(x) \leq 0 \\ & g_1(x) + g_2(y) \leq 0 \end{aligned} \tag{4}$$

Objective prediction network:

$$\Phi(x) = \min_{y: g_1(x) + g_2(y) \leq 0} f_2(y) \tag{5}$$

Surrogate problem:

$$\begin{aligned} \min_{x,y} \quad & f_1(x) + \Phi(x) \\ \text{s.t.} \quad & g(x) \leq 0 \end{aligned} \tag{6}$$

References I

[NBG+20] Vinod Nair, Sergey Bartunov, Felix Gimeno, Ingrid von Glehn, Pawel Lichocki, Ivan Lobov, Brendan O'Donoghue, Nicolas Sonnerat, Christian Tjandraatmadja, Pengming Wang, Ravichandra Addanki, Tharindi Hapuarachchi, Thomas Keck, James Keeling, Pushmeet Kohli, Ira Ktena, Yujia Li, Oriol Vinyals, and Yori Zwols, *Solving mixed integer programs using neural networks*, 2020.

Codes:

Neural diving:
https://github.com/sribdcn/Predict-and-Search$_M$*ILP*$_m$*ethod*/*tree*/*main*
B&B: https://github.com/piquepq/ML4CO$_b$*ranching*
Objective prediction:
https://github.com/yifengxiao1/approximate-and-optimize