Carnegie Mellon University

Neural Mixed Integer Linear Programming Solvers

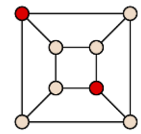
Invited Lecture (Graph 12) at CMU 11441/11741
Speaker: Shengyu Feng

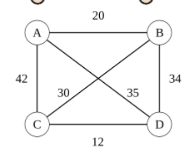
Roadmap

- Introduction to Mixed Integer Linear Programming (MILP)
- Branch and Bound (BnB)
- Neural Heuristics for BnB
  - Neural Branching
  - Neural Diving

Representation of Combinatorial Optimization Problems

- **Maximal Independent Set (MIS)**
  - Nodes and edges are straightforward
- **Traveling Salesman Problem (TSP)**
  - Nodes: cities
  - Edges: connections between cities
- **Job Shop Scheduling Problem (JSSP)**
  - Nodes?
  - Edges?





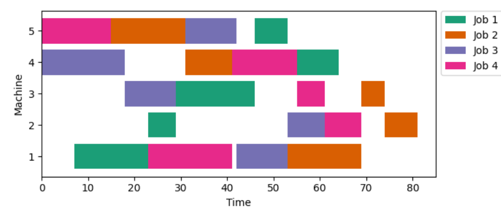


Image source: Wikipedia; Bruno Scalia C.F. Leite.

Mixed Integer Linear Programming (MILP)

Standard form:

Objective: $\max_{x_1,\cdots,x_n} c_1x_1 + \cdots + c_nx_n$

Subject to: $\begin{cases} A_{11}x_1 + \cdots + A_{1n}x_n \le b_1 \\ \vdots \\ A_{m1}x_1 + \cdots + A_{mn}x_n \le b_m \end{cases}$

$x_i \ge 0, i = 1, \cdots, n$

$x_i \in \mathbb{Z}, i \in \mathcal{I}$

Example:

- n types of products, m types of resources
- x_i: the amount of product i to produce
- c_i: the profit by producing a unit of product i
- Maximize the total profit $c_1x_1 + \cdots + c_nx_n$
- b_j: the amount of resource j
- A_{ji}: the consumption of resource j by producing a unit of product i

Mixed Integer Linear Programming (Cont'd)

Standard form:

Objective: $\max_{x_1,\cdots,x_n} c_1x_1 + \cdots + c_nx_n$

Subject to: $\begin{cases} A_{11}x_1 + \cdots + A_{1n}x_n \le b_1 \\ \vdots \\ A_{m1}x_1 + \cdots + A_{mn}x_n \le b_m \end{cases}$

$x_i \ge 0, i = 1, \cdots, n$

$x_i \in \mathbb{Z}, i \in \mathcal{I}$

Matrix form →

$\max_{\boldsymbol{x}} \boldsymbol{c}^T\boldsymbol{x}$

s.t. $\boldsymbol{Ax} \le \boldsymbol{b}$

$\boldsymbol{x} \ge \boldsymbol{0}$

$x_i \in \mathbb{Z}, i \in \mathcal{I}$

Graph Representation of a MILP Problem

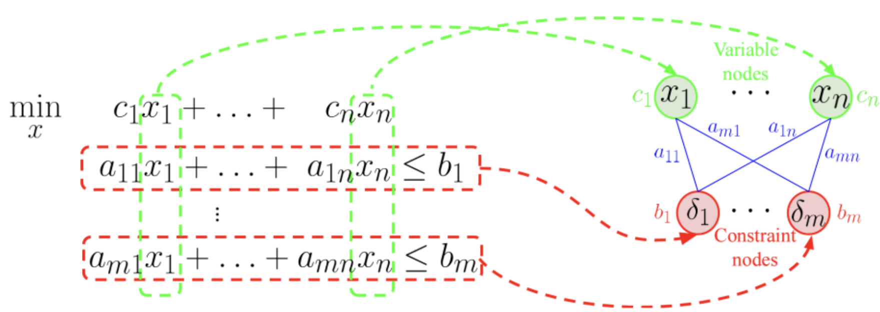
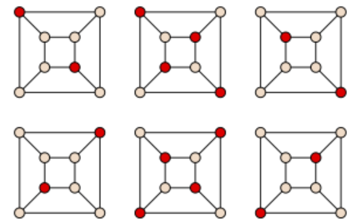


Figure 3 Bipartite graph representation of a MIP used as the input to a neural network. The set of n variables $\{x_1, \ldots, x_n\}$ and the set of m constraints $\{\delta_1, \ldots, \delta_m\}$ form the two sets of nodes of the bipartite graph. The coefficients are encoded as features of the nodes and edges.

Image source: Nair et al., 2021.

MILP Subsumes MIS



- The maximal subset of nodes without including adjacent pairs
- Whether node i in the set or not $x_i \in \{0,1\}$

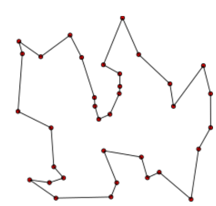
$$\max \sum_{i \in \mathcal{V}} x_i$$

s.t. $x_i + x_j \le 1, \ (i,j) \in \mathcal{E}$ — No adjacent pairs

$x_i \in \{0,1\}, \quad i \in \mathcal{V}$

Image source: Wikipedia.

MILP Subsumes TSP



- The shortest path to visit each city exactly once and return to the original city
- Whether traveling from city i to j: $x_{ij} \in \{0,1\}$
- Visiting order of city i: $u_i \in \{2, \cdots, n\}$ (smaller one being visited first)

$$\min \sum_{i=1}^{n} \sum_{j \ne i, j=1}^{n} c_{ij} x_{ij}:$$

$$x_{ij} \in \{0,1\} \qquad i,j = 1, \ldots, n;$$

$$\sum_{i=1, i \ne j}^{n} x_{ij} = 1 \qquad j = 1, \ldots, n;$$

Enter each city once

$$\sum_{j=1, j \ne i}^{n} x_{ij} = 1 \qquad i = 1, \ldots, n;$$

Leave each city once

$$u_i - u_j + 1 \le (n-1)(1 - x_{ij}) \qquad 2 \le i \ne j \le n;$$

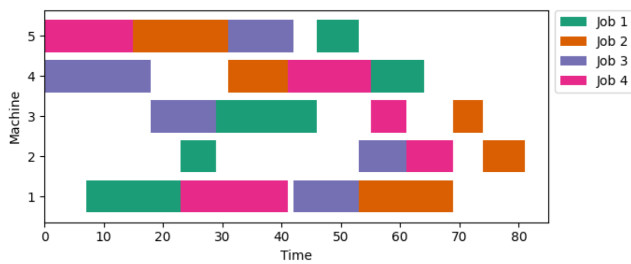
$$2 \le u_i \le n \qquad 2 \le i \le n.$$

If $x_{ij} = 1$, $u_j \ge u_i + 1$

Image source: Wikipedia.

MILP Subsumes JSSP

- Process J jobs on M machines in parallel, minimize the total processing time (makespan)
  - Each job j needs to follow some processing order $(\sigma_1^j, \cdots, \sigma_k^j)$ of machines
  - Each machine can only process one job at a time



Slide credit to: Bruno Scalia C.F. Leite.

MILP Subsumes JSSP (Cont'd)

- Process J jobs on M machines in parallel, minimize the total processing time (makespan)
  - Each job j needs to follow some processing order $(\sigma_1^j, \cdots, \sigma_k^j)$ of machines
  - Each machine can only process one job at a time
  - Starting time of job j: $x_{m,j}$; precedence between job j and k: $z_{m,j,k} \in \{0,1\}$ (on machine m)

$$
\begin{array}{lllr}
\min & C & & \\
\text{s.t.} & x_{\sigma_{h-1}^j,j} + p_{\sigma_{h-1}^j,j} \leq x_{\sigma_h^j,j} & \forall\ j \in J; h \in (2, ..., |M|) & \text{Follow processing order} \\
& x_{m,j} + p_{m,j} \leq x_{m,k} + V(1 - z_{m,j,k}) & \forall\ j,k \in J, j \neq k; m \in M & \text{Process one job at a time} \\
& z_{m,j,k} + z_{m,k,j} = 1 & \forall\ j,k \in J, j \neq k; m \in M & \text{Unique precedence} \\
& x_{\sigma_{|M|}^j,j} + p_{\sigma_{|M|}^j,j} \leq C & \forall\ j \in J & \\
& x_{m,j} \geq 0 & \forall\ j \in J; m \in M & \\
& z_{m,j,k} \in \{0,1\} & \forall\ j,k \in J; m \in M &
\end{array}
$$

Slide credit to: Bruno Scalia C.F. Leite.

Other Examples

- Knapsack problem (pack items into the knapsack)
- Combinatorial auction (distribute the items over bids)
- Crew scheduling (assign crew members to shifts)
- Treatment planning (treat tumor with radiation while minimize damage to the surrounding healthy tissue)
- CPU Resource Allocation
- … (ask GPT4)

Roadmap

✓ Introduction to Mixed Integer Linear Programming (MILP)

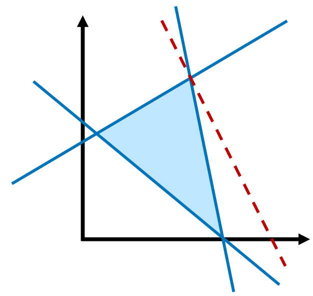
➢ Branch and Bound (BnB)

- Neural Heuristics for BnB
  - Neural Branching
  - Neural Diving

LP (P) vs. MILP (NP-Hard)

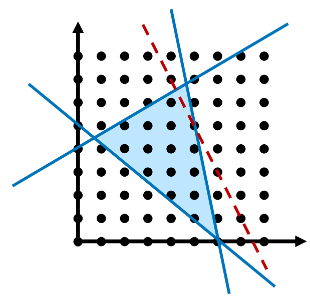
LP

$$\max_{\boldsymbol{x}} \boldsymbol{c}^T \boldsymbol{x}$$
$$\text{s.t. } \boldsymbol{A}\boldsymbol{x} \le \boldsymbol{b}$$
$$\boldsymbol{x} \ge \boldsymbol{0}$$



MILP

$$\max_{\boldsymbol{x}} \boldsymbol{c}^T \boldsymbol{x}$$
$$\text{s.t. } \boldsymbol{A}\boldsymbol{x} \le \boldsymbol{b}$$
$$\boldsymbol{x} \ge \boldsymbol{0}$$
$$x_i \in \mathbb{Z}, i \in \mathcal{I}$$



Branch and Bound (BnB)

- **Branch:** divide and conquer
- **Bound:** track the primal and dual bounds to reduce the searching space

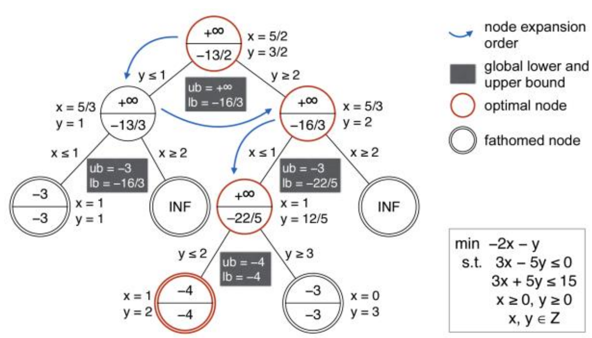


Image source: He et al., NIPS 2014.

Example

$$\text{maximize } Z = \$100x_1 + 150x_2$$

subject to

$$8{,}000x_1 + 4{,}000x_2 \le \$40{,}000$$
$$15x_1 + 30x_2 \le 200 \text{ ft}^2$$
$$x_1, x_2 \ge 0 \text{ and integer}$$

Image source: Bernald W. Taylor. Introduction to Management Science.

Dual Bound

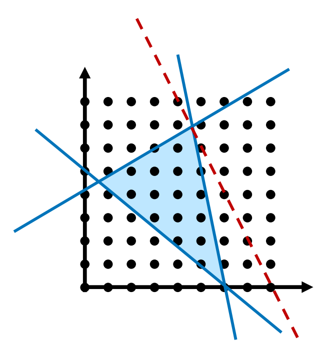
- **Linear relaxation**: remove/relax the integer constraints
- Expand the feasible region → a higher objective value (in maximization)
- It forms an upper bound ("dual bound")

$$\begin{aligned} \max_{\boldsymbol{x}}\ & \boldsymbol{c}^T\boldsymbol{x} \\ \text{s.t.}\ & \boldsymbol{A}\boldsymbol{x} \le \boldsymbol{b} \\ & \boldsymbol{x} \ge \boldsymbol{0} \\ & x_i \in \mathbb{Z}, i \in \mathcal{I} \end{aligned}$$

Linear relaxation →

$\le$

$$\begin{aligned} \max_{\boldsymbol{x}}\ & \boldsymbol{c}^T\boldsymbol{x} \\ \text{s.t.}\ & \boldsymbol{A}\boldsymbol{x} \le \boldsymbol{b} \\ & \boldsymbol{x} \ge \boldsymbol{0} \\ & \cancel{x_i \in \mathbb{Z}, i \in \mathcal{I}} \end{aligned}$$



Primal Bound

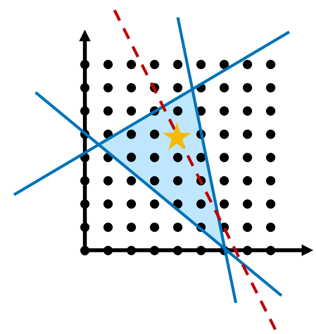
- Any feasible solution generates a lower bound ("primal bound")
- The heuristic used to find a feasible solution is known as the **primal heuristic**
- Example: rounding ($[1.2, 0.4, 0.8] \rightarrow [1, 0, 1]$)

$$\begin{aligned} \max_{\boldsymbol{x}}\ & \boldsymbol{c}^T\boldsymbol{x} \\ \text{s.t.}\ & \boldsymbol{A}\boldsymbol{x} \le \boldsymbol{b} \\ & \boldsymbol{x} \ge \boldsymbol{0} \\ & x_i \in \mathbb{Z}, i \in \mathcal{I} \end{aligned}$$

Primal Heuristic $\rightarrow$

$\geq$

$$\begin{aligned} \cancel{\max_{\boldsymbol{x}}}\ & \boldsymbol{c}^T\boldsymbol{x} \\ \text{s.t.}\ & \boldsymbol{A}\boldsymbol{x} \le \boldsymbol{b} \\ & \boldsymbol{x} \ge \boldsymbol{0} \\ & x_i \in \mathbb{Z}, i \in \mathcal{I} \end{aligned}$$



Example

$$\text{maximize } Z = \$100x_1 + 150x_2$$

subject to

$$\begin{aligned} 8{,}000x_1 + 4{,}000x_2 &\le \$40{,}000 \\ 15x_1 + 30x_2 &\le 200 \text{ ft}^2 \\ x_1, x_2 &\ge 0 \text{ \sout{and integer}} \end{aligned}$$

UB = 1,055.56 (x_1 = 2.22, x_2 = 5.56)
LB = 950 (x_1 = 2, x_2 = 5)

1
1,055.56

Linear relaxation result: $x_1 = 2.22, x_2 = 5.56, Z = 1055.56$

Feasible solution via rounding: $x_1 = 2, x_2 = 5, Z = 950$

Image source: Bernald W. Taylor. Introduction to Management Science.

Optimality: Primal Bound = Dual Bound

- If the relaxed solution happens to satisfy the integer constraint, then the solving is over
- Otherwise, we branch on the variable not satisfying the integer constraint
- Key idea: exclude the relaxed solution while keep all feasible solutions in the searching space

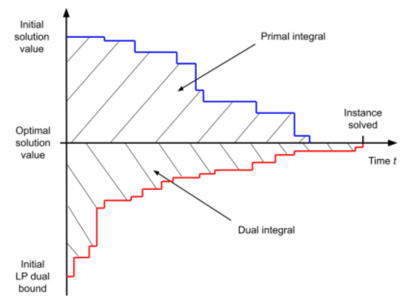
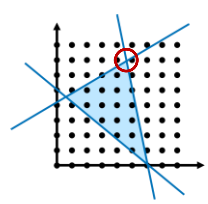


Image source: ML4CO competition.

Branching

$$\max_{\boldsymbol{x}} \boldsymbol{c}^T\boldsymbol{x}$$
s.t. $\boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b}$
$\boldsymbol{x} \geq \boldsymbol{0}$
$x_i \in \mathbb{Z}, i \in \mathcal{I}$



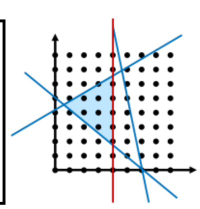
Relaxed solution $\boldsymbol{x}'$ violates the integer constraint

Divide into sub-problems

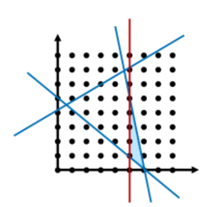
$x_1 \leq \lfloor x_1' \rfloor$

$x_1 \geq \lceil x_1' \rceil$

$$\max_{\boldsymbol{x}} \boldsymbol{c}^T\boldsymbol{x}$$
s.t. $\boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b}$
$\boldsymbol{x} \geq \boldsymbol{0}$
$x_1 \leq \lfloor x_1' \rfloor$
$x_i \in \mathbb{Z}, i \in \mathcal{I}$

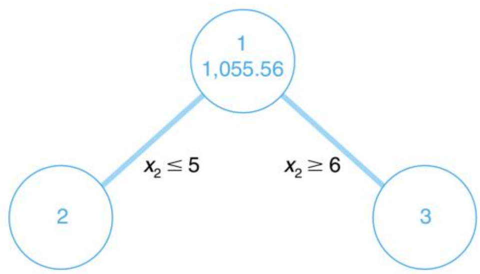


$$\max_{\boldsymbol{x}} \boldsymbol{c}^T\boldsymbol{x}$$
s.t. $\boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b}$
$\boldsymbol{x} \geq \boldsymbol{0}$
$x_1 \geq \lceil x_1' \rceil$
$x_i \in \mathbb{Z}, i \in \mathcal{I}$



Example

UB = 1,055.56 (x_1 = 2.22, x_2 = 5.56)
LB = 950 (x_1 = 2, x_2 = 5)

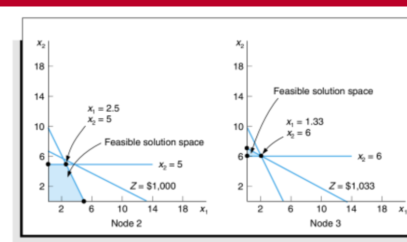


$$\begin{aligned} &\text{maximize } Z = \$100x_1 + 150x_2 \\ &\text{subject to} \\ &8{,}000x_1 + 4{,}000x_2 \le 40{,}000 \\ &15x_1 + 30x_2 \le 200 \\ &x_2 \le 5 \\ &x_1, x_2 \ge 0 \end{aligned}$$

$$\begin{aligned} &\text{maximize } Z = \$100x_1 + 150x_2 \\ &\text{subject to} \\ &8{,}000x_1 + 4{,}000x_2 \le 40{,}000 \\ &15x_1 + 30x_2 \le 200 \\ &x_2 \ge 6 \\ &x_1, x_2 \ge 0 \end{aligned}$$

Image source: Bernald W. Taylor. Introduction to Management Science.

Example



UB = 1,055.56 (x_1 = 2.22, x_2 = 5.56)
LB = 950 (x_1 = 2, x_2 = 5)

UB = 1,000 (x_1 = 2.5, x_2 = 5)
LB = 950 (x_1 = 2, x_2 = 5)

UB = 1,033 (x_1 = 1.33, x_2 = 6)
LB = 950 (x_1 = 2, x_2 = 5)

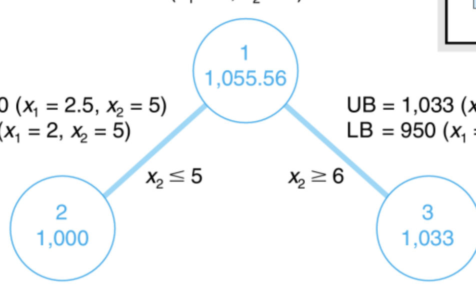


Image source: Bernald W. Taylor. Introduction to Management Science.

Carnegie Mellon University

Example

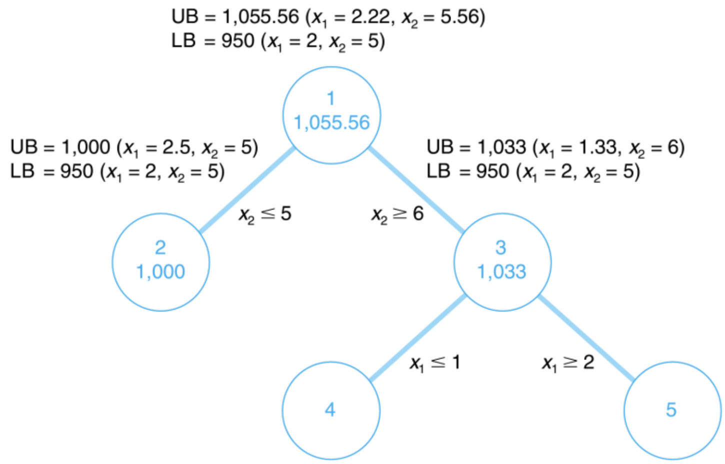


Image source: Bernald W. Taylor. Introduction to Management Science.

Carnegie Mellon University

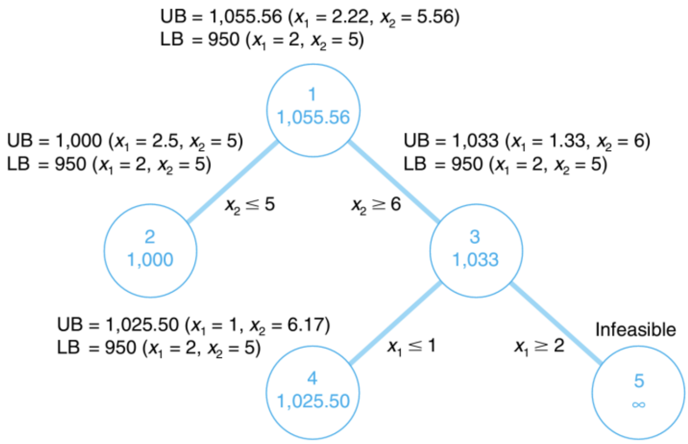
Fathomed (Leaf) Nodes

We **fathom** a node if it is fully solved or no longer worth exploration

- **Infeasibility**: the sub-problem is infeasible
- **Integrality**: the solution to the linear relaxation of the sub-problem satisfies the integer constraint (fully solved)
- **Bound**: the local upper bound (local dual bound) is not better than the global lower bound (global primal bound)

Slide credit to: Gianni A. Di Caro.

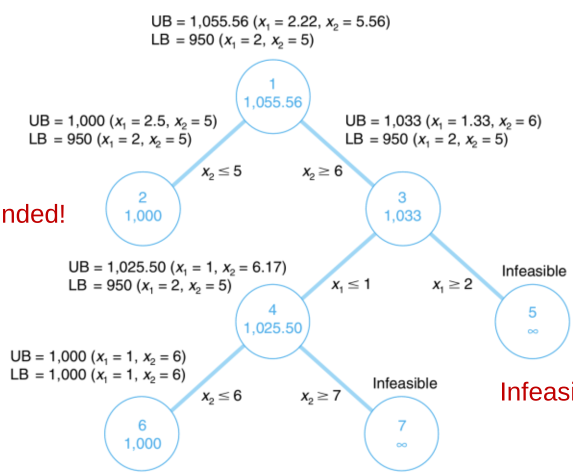
Example



Infeasible!

Image source: Bernald W. Taylor. Introduction to Management Science.

Example



$1000 \leq 1000$, bounded!

Infeasible!

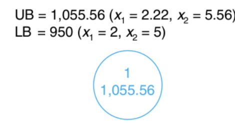
Upper bound=lower bound: solved!

Infeasible!

Image source: Bernald W. Taylor. Introduction to Management Science.

Efficiency of BnB is Decided by Its Heuristics

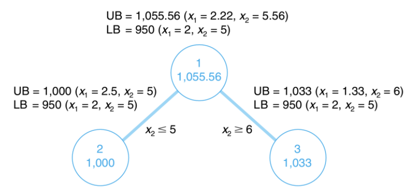
- **Variable selection**: which variable to branch



Branch on x_1 or x_2?

- **Node Selection**: which node to explore

Explore node 2 or 3?



- **Primal Heuristic**: how to efficiently find a high-quality feasible solution

Slide credit to: Gianni A. Di Caro.

Roadmap

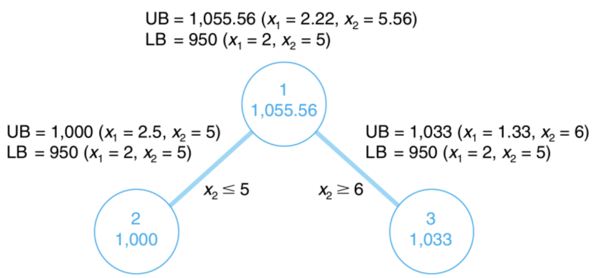
✓ Introduction to Mixed Integer Linear Programming (MILP)

✓ Branch and Bound (BnB)

➢ Neural Heuristics for BnB
- Neural Branching
- Neural Diving

Neural Branching: Imitation Learning

- Expert heuristic: full strong branching (FSB)
  - For each variable, compute the **dual bound improvement** after branching on it (high cost)
  - Then branch on the variable leading to the largest improvement



Dual bound improvement (x_2)
$1055.56 - \max\{1000, 1033\} = 22.56$

Image source: Bernald W. Taylor. Introduction to Management Science.

Neural Branching: Imitation Learning (Cont'd)

- Expert heuristic: full strong branching (FSB)
  - For each variable, compute the **dual bound improvement** after branching on it (high cost)
  - Then branch on the variable leading to the largest improvement
- Collect training pairs (node/subproblem, branching decision) offline via FSB

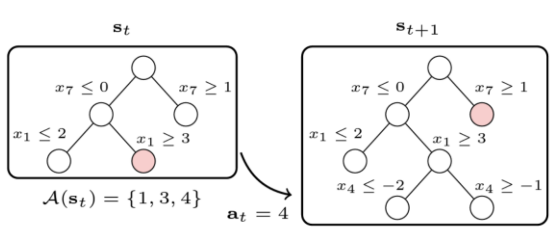


Image source: Gasse et al., NeuIPS 2019.

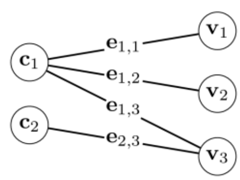
Carnegie Mellon University

Neural Branching: Imitation Learning (Cont'd)

- Expert heuristic: full strong branching (FSB)
  - For each variable, compute the **dual bound improvement** after branching on it (high cost)
  - Then branch on the variable leading to the largest improvement
- Collect training pairs (node/subproblem, branching decision) offline via FSB
- Train a GNN to imitate the choice of FSB

$$\text{maximize } z = 3x_1 + 5x_2 + 2x_3$$
$$\begin{aligned} 2x_1 + 4x_2 - x_3 &\le 10, \\ x_3 &\ge 3, \\ x_1 \ge 0,\ x_2 &\ge 0, \\ x_2 &\in \mathbb{Z}, \\ x_3 &\in \mathbb{Z}. \end{aligned}$$

⇨



GNN ⇨ $a_t = 2$ (branch on x_2)

Image source: Gasse et al., NeuIPS 2019.

Carnegie Mellon University

Experimental Results on MILPs

- Metric: dual gap (normalized dual bound, lower is better)

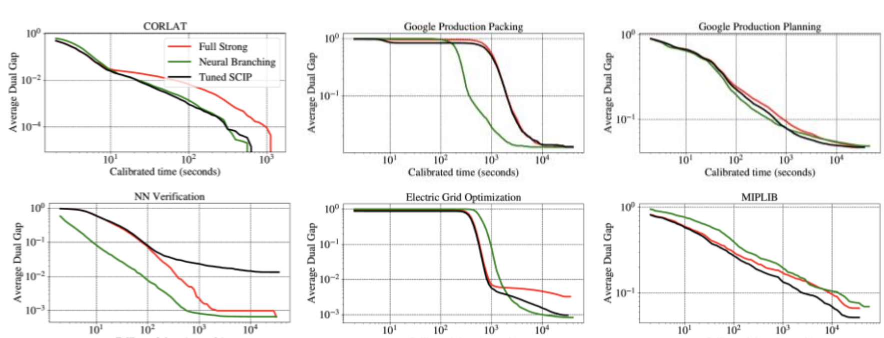


Image source: Nair et al., 2021.

Tuned SCIP: a solver based on human heuristics

Diving

- Rounding all variables does not necessarily lead to a good solution

$$\max_{x_1,\cdots,x_n} \sum_{i=1}^{n} x_i$$
s.t. $x_i + x_j + x_k \leq 1,\ (i,j,k) \in \mathcal{E}$
$x_i \geq 0,\ i = 1,\cdots,n$
$x_i \in \mathbb{Z},\ i = 1,\cdots,n$

Linear relaxation

$$\boldsymbol{x}' = \begin{bmatrix} 1/3 \\ 1/3 \\ \vdots \\ 1/3 \end{bmatrix}$$

Round

$$\boldsymbol{x} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

Diving (Cont'd)

- Rounding all variables does not necessarily lead to a good solution
- Diving: depth-first-search (rounding one variable at a time)
- Score the variables to decide the rounding order (heuristic-based)
- Example: $s_i = -|x_i' - round(x_i')|$

Use neural network!

New constraint
$x_i = round(x_i')$

$$\max_{x_1,\cdots,x_n} \sum_{i=1}^{n} x_i$$
s.t. $x_i + x_j + x_k \leq 1,\ (i,j,k) \in \mathcal{E}$
$x_i \geq 0,\ i = 1,\cdots,n$
$x_i \in \mathbb{Z},\ i = 1,\cdots,n$

Linear relaxation

$$\boldsymbol{x}' = \begin{bmatrix} 1/3 \\ 1/3 \\ \vdots \\ 1/3 \end{bmatrix}$$

Round x_i

$$\boldsymbol{x} = \begin{bmatrix} 0 \\ - \\ \vdots \\ - \end{bmatrix}$$

Neural Diving: Generative Modeling

- **Samples**: high-quality feasible solutions collected offline (run BnB for enough time)
- **Training**: non-autoregressive generation

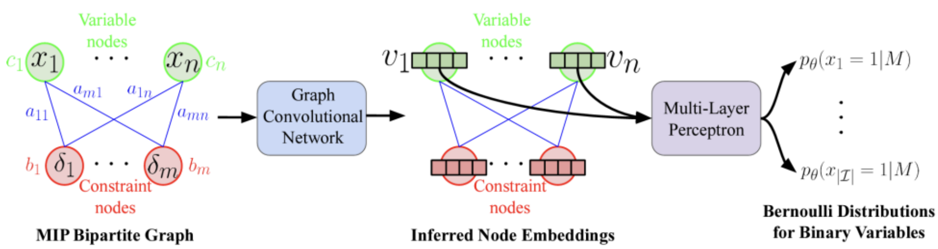
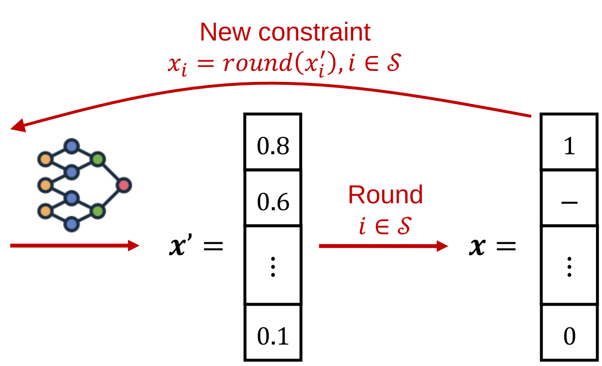


Image source: Nair et al., 2021.

Neural Diving: Generative Modeling (Cont'd)

- **Samples**: high-quality feasible solutions collected offline (run BnB for enough time)
- **Training**: non-autoregressive generation
- **Inference**: round the variables with high prediction confidence, dive the remainders

$$\begin{aligned} \max_{x_1,\cdots,x_n} \; & \sum_{i=1}^{n} x_i \\ \text{s.t.} \; & x_i + x_j + x_k \le 1,\ (i,j,k) \in \mathcal{E} \\ & x_i \ge 0,\ i = 1,\cdots,n \\ & x_i \in \mathbb{Z},\ i = 1,\cdots,n \end{aligned}$$



Experimental Results on MILPs

- Metric: primal gap (normalized primal bound, lower is better)

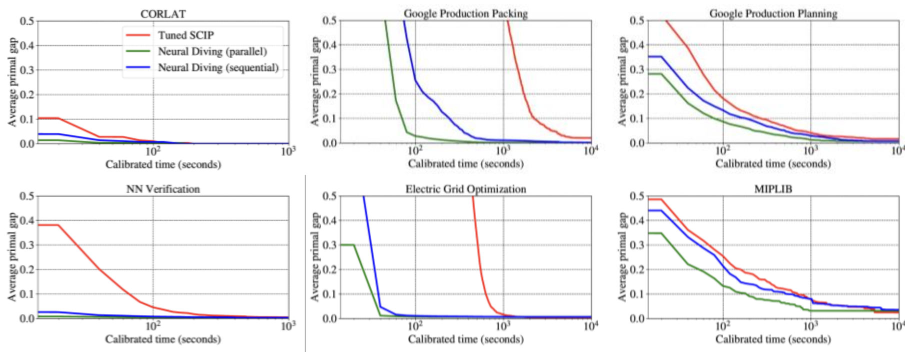


Image source: Nair et al., 2021.

Neural Branching + Neural Diving

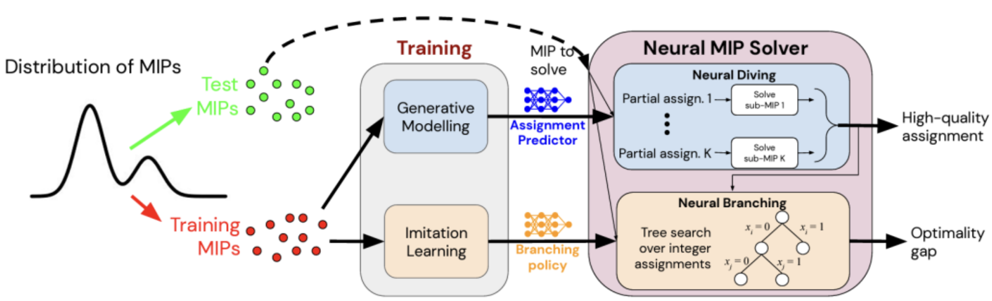


Image source: Nair et al., 2021.

Experimental Results on MILPs

- Metric: primal-dual gap (normalized difference between primal and bounds, lower is better)

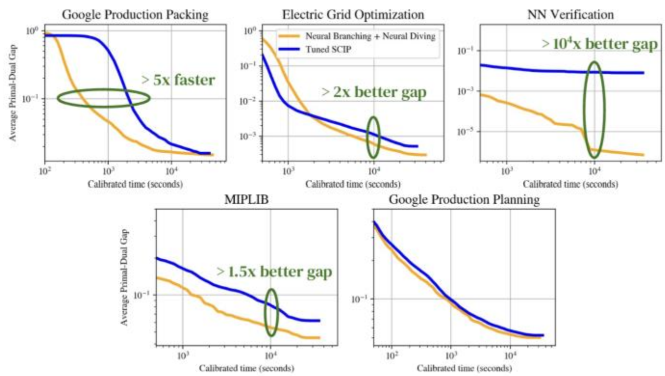


Image source: Nair et al., 2021.

Takeaways

- MILP formulation provides a more general representation for combinatorial optimization problems (the variable-constraint bipartite graph)

- MILP method: branch and bound

- Neural MILP solver: accelerate/improve the heuristics in existing MILP solvers