Mixed Integer Programming
The State of the Art

Robert E. Bixby
A Definition

A *mixed-integer program* (MIP) is an optimization problem of the form

\[
\begin{align*}
\text{Minimize} \quad & c^T x \\
\text{Subject to} \quad & Ax = b \\
& l \leq x \leq u \\
& \text{some or all } x_j \text{ integer}
\end{align*}
\]
Unit–Commitment Models

Electrical Power Industry, ERPI GS-6401, June 1989: Mixed-integer programming (MIP) is a powerful modeling tool, “They are, however, theoretically complicated and computationally cumbersome”

In Other Words: MIP is an interesting modeling “toy”, but it just doesn’t work in practice.

This perception began to change in 1999.
From the Rutger’s DIMACS Meeting 1999: California 7-Day Model

UNITCAL_7: 48939 constraints, 25755 variables (2856 binary)

Reported Results 1999 – machine unknown
2 Day model: 8 hours, no progress
7 Day model: 1 hour to solve initial LP

Desktop PC -- ran full 7-day model
CPLEX 6.5 (1999): 22 minutes, optimal
TODAY (2015): 15 seconds, optimal
Computational History: 1950 – 1998

- **1954** Dantzig, Fulkerson, S. Johnson: 42 city TSP
  - Solved to optimality using LP and cutting planes
- **1957** Gomory
  - Cutting plane algorithms
- **1960** Land, Doig; 1965 Dakin
  - B&B
- **1964–68** LP/90/94
  - First commercial application
- **IBM 360 computer**
  - 1974 MPSX/370
  - 1976 Sciconic
    - LP-based B&B
    - MIP became commercially viable
- **1974** – **1998** Good B&B remained the state-of-the-art in commercial codes, in spite of ….
  - Edmonds, polyhedral combinatorics
  - 1973 Padberg, cutting planes
  - 1973 Chvátal, revisited Gomory
  - 1974 Balas, disjunctive programming
  - 1983 Crowder, Johnson, Padberg: PIPX, pure 0/1 MIP
  - 1987 Van Roy and Wolsey: MPSARX, mixed 0/1 MIP
  - TSP, Grötschel, Padberg, …
1998 ... A New Generation of MIP Codes

- Linear programming
  - Stable, robust dual simplex
- Variable/node selection
  - Influenced by traveling salesman problem
- Primal heuristics
  - 12 different tried at root
  - Retried based upon success
- Node presolve
  - Fast, incremental bound strengthening (very similar to Constraint Programming)

- Presolve – numerous small ideas
  - Probing in constraints:
    \[ \sum x_j \leq (\sum u_j) y, \ y = 0/1 \]
    \[ \Rightarrow x_j \leq u_j y \text{ (for all } j) \]
- Cutting planes
  - Gomory, mixed-integer rounding (MIR), knapsack covers, flow covers, cliques, GUB covers, implied bounds, zero-half cuts, path cuts
MIP Speedups
Some Test Results

- **Test set:** 1852 real-world MIPs
  - Full library
    - 2791 MIPs
  - Removed:
    - 559 “Easy” MIPs
    - 348 “Duplicates”
    - 22 “Hard” LPs (0.8%)

- **Parameter settings**
  - Pure defaults
  - 30000 second time limit

- **Versions Run**

**Cumulative Speedup**

- **Mature Dual Simplex: 1994**
- **Mined Theoretical Backlog: 1998**
- **29530x improvement**
Progress: 2009 – Present
Gurobi MIP Library

(3550 models)
MIP Speedup 2009–Present

- **Starting point**
  - Gurobi 1.0 & CPLEX 11.0 ~equivalent on 4-core machine

- **Gurobi Version–to–version improvements**
  - Gurobi 1.0 -> 2.0: 2.4X
  - Gurobi 2.0 -> 3.0: 2.2X (5.1X)
  - Gurobi 3.0 -> 4.0: 1.3X (6.6X)
  - Gurobi 4.0 -> 5.0: 2.0X (12.8X)
  - Gurobi 5.0 -> 6.0: 2.2X (27.6X)
  - Gurobi 6.0 -> (6.5): 1.7X (46.0X)

- **Machine–independent IMPROVEMENT since 1991**
  - Over 1.3 million X -- 1.8X/year
Suppose you were given the following choices:

- **Option 1**: Solve a MIP with today’s solution technology on a machine from 1991
- **Option 2**: Solve a MIP with 1991 solution technology on a machine from today

*Which option should you choose?*

*Answer*: Option 1 would be faster by a factor of approximately 300.
Thank you