Outline

- Progress in computing machines
- Linear programming (LP)
  - Introduction
  - Computational history 1947 to late 1980s
  - The last decade
- Mixed-integer programming (MIP) – Closing the gap between theory and practice
  - Introduction
  - Computation history 1954 to late 1990s
  - Features in modern codes
  - The last year
- The future
Progress in Computing Machines

Late 1980s to 2000:
A Simple LP Example

Machine Comparison


<table>
<thead>
<tr>
<th>Machine</th>
<th>Seconds</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sun 3/150</td>
<td>44064.0</td>
</tr>
<tr>
<td>Pentium PC (60 MHz)</td>
<td>222.6</td>
</tr>
<tr>
<td>IBM 590 Powerstation</td>
<td>65.0</td>
</tr>
<tr>
<td>SGI R8000 Power Challenge</td>
<td>44.8</td>
</tr>
<tr>
<td>Athlon (650 MHz)</td>
<td>22.2</td>
</tr>
<tr>
<td>Compaq Alpha</td>
<td>6.9</td>
</tr>
</tbody>
</table>

~5000x improvement
A linear program (LP) is an optimization problem of the form

\[
\begin{align*}
\text{Minimize} & \quad c^T x \\
\text{Subject to} & \quad Ax = b \\
& \quad \ell \leq x \leq u
\end{align*}
\]
“A certain wide class of practical problems appears to be just beyond the range of modern computing machinery. These problems occur in everyday life; they run the gamut from some very simple situations that confront an individual to those connected with the national economy as a whole.”

George B. Dantzig, 1948

---

History: 1950 – 1990

- 1947 Dantzig invents simplex method
- 1951 SEAC (Standards Eastern Automatic Computer) 48 cons. & 72 vars.
  - “One could have started an iteration, gone to lunch and returned before it finished” William Orchard-Hays, 1990.
- 1963 IBM 7090, LP90: 1024 cons.
  - Oil companies used LP.
- 1973 IBM 360, MPSX & MPSIII, 32000 cons.
  - These codes lasted into the mid 80s with little fundamental change.
History: 1950 – 1990

• Mid 1980s – 1990
  – 1984 Karmarkar, Interior-point methods
  – 1991 Grötschel: “Some linear programs were hard to solve, even for highly praised commercial codes like IBM’s MPSX”

• Late 1980s:
  – OSL, XPRESS, CPLEX … started (see Padberg & Rinaldi, TSP)

1989 – 1998
An Example
A Fleet Assignment Model

• LAU2: A Fleet Assignment Model (1989)
  – 4420 constraints
  – 6711 variables
  – 101377 nonzeros
• Tests run on 500 MHz Alpha 21264 & Cray Y/MP

LAU2: 4420 x 6711, 101377 nz

• Idea 0: Run CPLEX
  – 7 hours on a Cray Y/MP, stalled in phase I.
• Idea 1: Handling degeneracy: Perturbation
  – 12332 secs, 1548304 itns
• Idea 2: Barrier, OB1 (Lustig, Marsten, Shanno)
  – 656 secs (Cray Y/MP)
• Idea 3: Better pricing: hybrid = partial pricing + DEVEX (1973)
  – Hybrid: 307 secs, 48301 itns
LAU2: 4420 x 6711, 101377 nz

- Idea 4: Use the dual
  - Explicit dual: 103 secs, 14914 itns
  - Dual simplex: 345 secs, 44695 itns
- Idea 5: Steepest edge
  - Dual steepest edge: 21 secs, 4906 itns
- Idea 6: Today’s parallel barrier
  - 50 secs, 1 processor
  - 26 secs, 4 processors
1999 – 2000

- Examine larger models: \( \geq 10000 \) rows
- Bottleneck:
  - Linear solves for systems with very sparse input and very sparse output
  - BTRAN, FTRAN operations
- Eliminating the Bottleneck:
  - These solves can be done in “linear time”
  - Linear-Algebra folklore: reachability

Bottleneck Removed: Exploiting It

- Dual:
  - Variables with two finite bounds usually do not need to be binding in the ratio test.
- Sparse pricing:
  - Fast updates for minimum when few reduced-costs change.
<table>
<thead>
<tr>
<th>Model</th>
<th>Rows</th>
<th>Cols</th>
<th>NZs</th>
<th>Model</th>
<th>Rows</th>
<th>Cols</th>
<th>NZs</th>
<th>Model</th>
<th>Rows</th>
<th>Cols</th>
<th>NZs</th>
</tr>
</thead>
<tbody>
<tr>
<td>M1</td>
<td>10295</td>
<td>50040</td>
<td>150110</td>
<td>M28</td>
<td>27839</td>
<td>26243</td>
<td>26198</td>
<td>M35</td>
<td>55463</td>
<td>191333</td>
<td>840986</td>
</tr>
<tr>
<td>M2</td>
<td>13005</td>
<td>71133</td>
<td>361567</td>
<td>M29</td>
<td>28240</td>
<td>55200</td>
<td>161640</td>
<td>M36</td>
<td>60394</td>
<td>100078</td>
<td>485414</td>
</tr>
<tr>
<td>M3</td>
<td>14738</td>
<td>33025</td>
<td>151383</td>
<td>M30</td>
<td>28420</td>
<td>164024</td>
<td>550253</td>
<td>M37</td>
<td>63856</td>
<td>144893</td>
<td>717229</td>
</tr>
<tr>
<td>M4</td>
<td>15014</td>
<td>37372</td>
<td>132066</td>
<td>M31</td>
<td>29002</td>
<td>111722</td>
<td>2632880</td>
<td>M38</td>
<td>66185</td>
<td>157496</td>
<td>418231</td>
</tr>
<tr>
<td>M5</td>
<td>15351</td>
<td>34553</td>
<td>132295</td>
<td>M32</td>
<td>29017</td>
<td>20074</td>
<td>2001102</td>
<td>M39</td>
<td>67745</td>
<td>111891</td>
<td>305135</td>
</tr>
<tr>
<td>M6</td>
<td>15349</td>
<td>35215</td>
<td>162709</td>
<td>M33</td>
<td>29147</td>
<td>9964</td>
<td>1013168</td>
<td>M40</td>
<td>69418</td>
<td>612068</td>
<td>1722112</td>
</tr>
<tr>
<td>M7</td>
<td>15455</td>
<td>59942</td>
<td>225514</td>
<td>M34</td>
<td>29272</td>
<td>98124</td>
<td>196264</td>
<td>M41</td>
<td>84469</td>
<td>318800</td>
<td>1893900</td>
</tr>
<tr>
<td>M8</td>
<td>15540</td>
<td>23752</td>
<td>86753</td>
<td>M35</td>
<td>30190</td>
<td>57000</td>
<td>623730</td>
<td>M42</td>
<td>92051</td>
<td>197469</td>
<td>749771</td>
</tr>
<tr>
<td>M9</td>
<td>16223</td>
<td>28568</td>
<td>88450</td>
<td>M36</td>
<td>31770</td>
<td>272372</td>
<td>829940</td>
<td>M43</td>
<td>95787</td>
<td>326504</td>
<td>2102723</td>
</tr>
<tr>
<td>M10</td>
<td>16768</td>
<td>39474</td>
<td>203112</td>
<td>M37</td>
<td>33440</td>
<td>56624</td>
<td>161831</td>
<td>M44</td>
<td>100393</td>
<td>112935</td>
<td>602948</td>
</tr>
<tr>
<td>M11</td>
<td>17081</td>
<td>161188</td>
<td>609273</td>
<td>M38</td>
<td>34994</td>
<td>87510</td>
<td>206713</td>
<td>M45</td>
<td>118158</td>
<td>487427</td>
<td>974684</td>
</tr>
<tr>
<td>M12</td>
<td>18262</td>
<td>23211</td>
<td>136304</td>
<td>M39</td>
<td>35519</td>
<td>43582</td>
<td>577463</td>
<td>M46</td>
<td>123894</td>
<td>332988</td>
<td>459680</td>
</tr>
<tr>
<td>M13</td>
<td>19103</td>
<td>33490</td>
<td>276995</td>
<td>M40</td>
<td>35645</td>
<td>34675</td>
<td>209705</td>
<td>M47</td>
<td>129311</td>
<td>105109</td>
<td>457138</td>
</tr>
<tr>
<td>M14</td>
<td>19374</td>
<td>18670</td>
<td>532558</td>
<td>M41</td>
<td>36460</td>
<td>92878</td>
<td>246006</td>
<td>M48</td>
<td>135211</td>
<td>83253</td>
<td>124788</td>
</tr>
<tr>
<td>M15</td>
<td>19519</td>
<td>45832</td>
<td>124200</td>
<td>M42</td>
<td>36762</td>
<td>261079</td>
<td>1508190</td>
<td>M49</td>
<td>142191</td>
<td>461702</td>
<td>1025706</td>
</tr>
<tr>
<td>M16</td>
<td>19644</td>
<td>55528</td>
<td>152852</td>
<td>M43</td>
<td>39067</td>
<td>125000</td>
<td>381259</td>
<td>M50</td>
<td>155255</td>
<td>377191</td>
<td>200165</td>
</tr>
<tr>
<td>M17</td>
<td>19999</td>
<td>85191</td>
<td>170269</td>
<td>M44</td>
<td>41340</td>
<td>64162</td>
<td>370839</td>
<td>M51</td>
<td>165325</td>
<td>377191</td>
<td>200165</td>
</tr>
<tr>
<td>M18</td>
<td>21019</td>
<td>115761</td>
<td>724342</td>
<td>M45</td>
<td>41344</td>
<td>183969</td>
<td>2258354</td>
<td>M52</td>
<td>175147</td>
<td>356239</td>
<td>1211488</td>
</tr>
<tr>
<td>M19</td>
<td>22513</td>
<td>92785</td>
<td>337746</td>
<td>M46</td>
<td>41365</td>
<td>79770</td>
<td>2711018</td>
<td>M53</td>
<td>185329</td>
<td>188887</td>
<td>2787708</td>
</tr>
<tr>
<td>M20</td>
<td>22797</td>
<td>63995</td>
<td>172018</td>
<td>M47</td>
<td>43387</td>
<td>107164</td>
<td>189894</td>
<td>M54</td>
<td>196441</td>
<td>237332</td>
<td>387080</td>
</tr>
<tr>
<td>M21</td>
<td>23610</td>
<td>44683</td>
<td>156402</td>
<td>M48</td>
<td>43437</td>
<td>164831</td>
<td>722056</td>
<td>M55</td>
<td>209760</td>
<td>363902</td>
<td>1951495</td>
</tr>
<tr>
<td>M22</td>
<td>23700</td>
<td>23025</td>
<td>190945</td>
<td>M49</td>
<td>44150</td>
<td>200777</td>
<td>4965017</td>
<td>M56</td>
<td>228940</td>
<td>1205454</td>
<td>6481640</td>
</tr>
<tr>
<td>M23</td>
<td>23712</td>
<td>31680</td>
<td>81241</td>
<td>M50</td>
<td>44211</td>
<td>37139</td>
<td>321683</td>
<td>M57</td>
<td>240756</td>
<td>920105</td>
<td>5335426</td>
</tr>
<tr>
<td>M24</td>
<td>24377</td>
<td>46592</td>
<td>2139096</td>
<td>M51</td>
<td>47423</td>
<td>81915</td>
<td>228655</td>
<td>M58</td>
<td>312956</td>
<td>838512</td>
<td>1231403</td>
</tr>
<tr>
<td>M25</td>
<td>26618</td>
<td>38094</td>
<td>1807713</td>
<td>M52</td>
<td>48548</td>
<td>163000</td>
<td>678582</td>
<td>M59</td>
<td>344297</td>
<td>559436</td>
<td>1905849</td>
</tr>
<tr>
<td>M26</td>
<td>27349</td>
<td>97710</td>
<td>28442</td>
<td>M53</td>
<td>54447</td>
<td>326504</td>
<td>1607146</td>
<td>M60</td>
<td>588350</td>
<td>1533590</td>
<td>5327318</td>
</tr>
<tr>
<td>M27</td>
<td>27441</td>
<td>15128</td>
<td>96118</td>
<td>M54</td>
<td>55020</td>
<td>117910</td>
<td>901081</td>
<td>M61</td>
<td>716772</td>
<td>1169910</td>
<td>2511088</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### BIG Performance Improvement

- **1999 – 2000**

<table>
<thead>
<tr>
<th>Segment</th>
<th>Primal</th>
<th>Dual</th>
<th>Best</th>
</tr>
</thead>
<tbody>
<tr>
<td>Biggest 10</td>
<td>8.5</td>
<td>22.3</td>
<td>18.0</td>
</tr>
<tr>
<td>Biggest 20</td>
<td>7.9</td>
<td>18.8</td>
<td>12.2</td>
</tr>
<tr>
<td>Biggest 30</td>
<td>7.4</td>
<td>20.2</td>
<td>11.3</td>
</tr>
<tr>
<td>Biggest 40</td>
<td>6.4</td>
<td>14.0</td>
<td>8.0</td>
</tr>
<tr>
<td>Biggest 50</td>
<td>5.5</td>
<td>11.9</td>
<td>6.7</td>
</tr>
<tr>
<td>Biggest 60</td>
<td>5.2</td>
<td>10.1</td>
<td>6.2</td>
</tr>
<tr>
<td>Biggest 70</td>
<td>5.1</td>
<td>9.1</td>
<td>5.6</td>
</tr>
<tr>
<td>All</td>
<td>4.4</td>
<td>8.0</td>
<td>5.2</td>
</tr>
</tbody>
</table>

- **Barrier, All: 3.6**
Algorithm Comparison

• Dual vs. Primal: 2.6x
  – (7 not solved by primal)
• Dual vs. Barrier: 1.2x
• Remarks:
  – PC has poor floating-point performance
    ⇒ bad for barrier
  – Parallel barrier changes these results
• And barrier is now faster anyway:
  – CPLEX 7.0 barrier 1.6x faster

Mixed Integer Programming (MIP)

Closing the Gap between Theory and Practice
A mixed-integer program (MIP) is an optimization problem of the form

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

Branch and Bound (Cut) Tree

Remarks:
(1) GAP = 0 \Rightarrow Proof of optimality
(2) Practice: Often good enough to have good Solution
MIP Really Is HARD

• A Customer Model:
  – 44 cons, 51 vars, 167 nzs, maximization
  – 51 general integer variables
• Branch-and-Cut:
  – Initial integer solution -2586.0
  – Initial upper bound -1379.4
  – After 120,000 seconds,
    • 370,000,000 B&C nodes, 45 Gig tree
    • Integer solution and bound: UNCHANGED

MIP Really Is HARD

• Electrical Power Industry, ERPI GS-6401, June 1989:
  – Mixed-integer programming is a powerful modeling tool, “They are, however, theoretically complicated and computationally cumbersome”
California Unit Commitment

- 7-Day Model:
  - UNITCAL_7
  - 48939 cons, 25755 vars (2856 binary)
- Previous attempts (by model formulator)
  - 2 Day model: 8 hours, no progress
  - 7 Day model: 1 hour to solve initial LP
- CPLEX 7.0 on Compaq Alpha
  - Running defaults ...

Model: UNITCAL_7

Problem 'unitcal_7.sav.gz' read.
Reduced MIP has 39359 rows, 20400 columns, and 106950 nonzeros.
Root relaxation solution time = 5.22 sec.

<table>
<thead>
<tr>
<th>Nodes</th>
<th>Left</th>
<th>Objective</th>
<th>Inf</th>
<th>Best Integer</th>
<th>Best Node</th>
<th>ItCnt</th>
<th>Gap</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1.9396e+07</td>
<td>797</td>
<td>1.9396e+07</td>
<td>13871</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>9487e-07</td>
<td>315</td>
<td></td>
<td>Cuts: 1992</td>
<td>17948</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.956e+07</td>
<td>377</td>
<td></td>
<td></td>
<td>Cuts: 1178</td>
<td>229351</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.956e+07</td>
<td>347</td>
<td></td>
<td></td>
<td>Cuts: 247</td>
<td>24162</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.956e+07</td>
<td>346</td>
<td></td>
<td></td>
<td>Flowcuts: 68</td>
<td>24508</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.956e+07</td>
<td>354</td>
<td></td>
<td></td>
<td>Cuts: 105</td>
<td>25024</td>
<td></td>
<td></td>
</tr>
<tr>
<td>100</td>
<td>87</td>
<td>1.9922e+07</td>
<td>110</td>
<td>1.9571e+07</td>
<td>34065</td>
<td></td>
<td></td>
</tr>
<tr>
<td>* 152</td>
<td>135</td>
<td>1.9958e+07</td>
<td></td>
<td>1.9958e+07</td>
<td>34973</td>
<td>1.94%</td>
<td></td>
</tr>
<tr>
<td>* 1157</td>
<td>396</td>
<td>1.9636e+07</td>
<td></td>
<td>1.9626e+07</td>
<td>178619</td>
<td>0.05%</td>
<td></td>
</tr>
</tbody>
</table>

GUB cover cuts applied: 2
Cover cuts applied: 7
Implied bound cuts applied: 1186
Flow cuts applied: 1120
Flow path cuts applied: 7
Gomory fractional cuts applied: 147

Integer optimal solution: Objective = 1.9635558244e+07
Solution time = 1234.05 sec. Iterations = 310699 Nodes = 6718
History: 1950 –1998

- 1954 Dantzig, Fulkerson, S. Johnson: 49 city TSP
  - Solved to optimality using cutting planes and solving LPs by hand
- 1957 Gomory
  - Cutting plane algorithm: A complete solution
- 1960 Land, Doig
  - B&B
- 1966 Beale, Small
  - B&B, Depth-first-search
- 1972 UMPIRE, Forrest, Hirst, Tomlin
  - SOS, pseudo-costs, best projection, …

• 1972 – 1998 Good B&B remained the state-of-the-art in commercial codes, in spite of
  - 1983 Crowder, Johnson, Padberg: PIPX, pure 0/1 MIP
  - 1987 Van Roy and Wolsey: MPSARX, mixed 0/1 MIP
  - Grötschel, Padberg, ... TSP
    (120, 666, 2392 city models solved)

1998…New Generation of MIP Codes

- Linear programming
  - Usable, stable, robust performance
- Node selection
  - Hybrid breadth- and depth-first-search.
- Variable selection
  - pseudo-costs, strong branching
- Primal heuristics
  - 5 different tried at root, one selected
- Node presolve
  - Fast, incremental bound strengthening
- Probing
  - Three levels
- Auto disaggregation
  - $\sum x_j \leq (\sum u_j) y, y = 0/1$ preferred
- Cutting planes
  - Gomory, knapsack covers, flow covers, mix-integer rounding, cliques, GUB covers, implied bounds, path cuts, disjunctive cuts
\[ x + y \geq 3.5, \ x \geq 0, \ y \text{ integral} \]
Gomory Mixed Cut

- Given \( y, x_j \in \mathbb{Z}^+ \), and \( y + \sum a_{ij}x_j = d = \lfloor d \rfloor + f, \ f > 0 \)
  - Rounding: Where \( a_{ij} = \lfloor a_{ij} \rfloor + f_j \), define
    \[ t = y + \sum \lfloor a_{ij} \rfloor x_j: f_j \leq f \] \[ + \sum \lceil a_{ij} \rceil x_j: f_j > f \] \( \in \mathbb{Z} \)
  - Then \( \sum (f_j x_j: f_j \leq f) + \sum (f_j-1)x_j: f_j > f) = d - t \)

- Disjunction:
  - \( t \leq \lfloor d \rfloor \Rightarrow \sum (f_j x_j: f_j \leq f) \geq f \)
  - \( t \geq \lceil d \rceil \Rightarrow \sum ((1-f_j)x_j: f_j > f) \geq 1-f \)

- Combining:
  - \( \sum ((f_j/f)x_j: f_j \leq f) + \sum ((1-f_j)/(1-f))x_j: f_j > f) \geq 1 \)

How Much Real Progress?

- Example: P2756,
  - 755 cons, 2756 vars (all binary)
- Before 1983:
  - Unsolved
- Crowder, Johnson, Padberg, 1983:
  - 54.4 mins, 2392 nodes
- CPLEX 7.0 (2000):
  - 1.3 secs, 25 nodes
  - (2500x faster, < machine improvement)
BIGTEST – 80 Models

- CPLEX 6.0 versus CPLEX 7.0
- Running defaults (7200 second limit)
  - CPLEX 6.0 FAILS on 31
  - CPLEX 7.0 FAILS on 2

- CPLEX 6.0 tuned versus CPLEX 7.0 defaults
  - CPLEX 7.0 5.1x faster

Impact: Individual Cuts

- Cliques - 3 %
- Implied - 1 %
- Path 0 %
- GUB covers 14 %
- Disjunctive 17 %
- Flow covers 41 %
- MIR 56 %
- Covers 56 %
- Gomory 114 %
Future?

- **Linear programming**
  - The improvements in the last 2 years were completely unanticipated!
  - Examine behavior of even larger and more-difficult models.

- **Mixed-integer programming**
  - More of the same, but we need a better pipeline …
  - Increasingly exploit special structure.