Advanced Operations Research Techniques
IE316

Lecture 24

Dr. Ted Ralphs
Reading for This Lecture

- Bertsimas Sections 10.2, 10.3, 11.1, 11.2
Preprocessing

• Often, it is possible to simplify a model using logical arguments.

• Most commercial LP solvers do preprocessing automatically, but if you are developing a model that will be solved repeatedly, it may be worthwhile.

• The constraint \( a^1x \leq b_1 \) dominates the constraint \( a^2x \leq b_2 \) if

\[
\begin{align*}
a^1_i & \geq a^2_i \quad \forall i, \text{ and } \\
b_1 & \leq b_2
\end{align*}
\]

• In this case, the dominated inequality could be deleted.

• We can also derive implied bounds for variables from each constraint \( ax \leq b \). If \( a_0 > 0 \), then

\[
x_1 \leq (b - \sum_{j:a_j>0} a_j l_j - \sum_{j:a_j<0} a_j u_j)/a_0
\]
More Preprocessing

• The constraint $ax \leq b$ is redundant if

$$\sum_{j:a_j>0} a_j u_j + \sum_{j:a_j<0} a_j l_j \leq b.$$  

• The LP is infeasible if

$$\sum_{j:a_j>0} a_j l_j + \sum_{j:a_j<0} a_j u_j > b.$$  

• For an LP of the form $\min\{c^T x | Ax \geq b, l \leq x \leq b\}$,

  - If $a_{ij} \geq 0 \forall i \in [1..m]$ and $c_j < 0$, then $x_j = u_j$ in any optimal solution.
  - If $a_{ij} \leq 0 \forall i \in [1..m]$ and $c_j > 0$, then $x_j = l_j$ in any optimal solution.
More Preprocessing

• More sophisticated rules can also be applied.

• It is possible to take advantage of problem structure.

• Effect of preprocessing (example from the book, p. 540):

<table>
<thead>
<tr>
<th></th>
<th>Rows</th>
<th>Columns</th>
<th>Iterations</th>
<th>Seconds</th>
</tr>
</thead>
<tbody>
<tr>
<td>No preproc.</td>
<td>13,689</td>
<td>17,148</td>
<td>39,429</td>
<td>10,094</td>
</tr>
<tr>
<td>Preproc.</td>
<td>5,579</td>
<td>9,508</td>
<td>18,975</td>
<td>2,381</td>
</tr>
</tbody>
</table>
Round-off Error and Scaling

- One of the big issues faced in the practice of linear programming is **round-off error**.

- Round-off error occurs simply because computers do not perform **exact arithmetic**.

- Computers perform **floating point** arithmetic, which means that computations get rounded off.

- These round-off errors can accumulate until they become significant.

- To avoid round-off error, it is important that the matrix be **scaled** properly.

- The existence of variables and constraints that are on vastly different scales can cause **numerical problems**.

- Most LP solvers will scale the problem automatically, but it’s always better to start with a **well-scaled model**.
Error Tolerances

- Because of round-off error, LP solvers have specified *error tolerances*.

- Some have different error tolerances for different operations.

- These error tolerances are especially important in integer programming.

- Often, a variable will have a value that is very close to an integer value.

- If it is within the error tolerance, then it is considered integer.

- However, the rounded solution may not be feasible.

- If you are having numerical problems, you may have to adjust the error tolerances or consider *scaling*.

- Another possibility is to adjust the *basis refactorization frequency*.
Commercial LP Solvers

• Not all LP solvers are created equal.

• Some issues that set LP solvers apart from each other.
  – Preprocessing
  – Numerical stability
  – Handling of degeneracy
  – Methods of pricing
  – Methods for determining the pivot element
  – Methods other than simplex (barrier methods)

• Some commercial codes (see NEOS guide for full list)
  – CPLEX (ILOG)
  – OSL (IBM)
  – XPRESS-MP (Dash)

• There are also numerous free solvers available on the Web.
Improvements to Simplex: Case Study

• This table is for a large linear program (approximately 50K rows, 175K columns, and 400K nonzeros), taken from the *ILOG Optimization Times*.

• These tests were all run on the same machine (296 MHz Sun Ultrasparc).

• This shows the dramatic difference implementation can make.

<table>
<thead>
<tr>
<th>CPLEX Version</th>
<th>Year</th>
<th>Time (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>1988</td>
<td>57,840</td>
</tr>
<tr>
<td>3.0</td>
<td>1994</td>
<td>4,555</td>
</tr>
<tr>
<td>5.0</td>
<td>1996</td>
<td>3,835</td>
</tr>
<tr>
<td>6.5</td>
<td>1999</td>
<td>165</td>
</tr>
</tbody>
</table>
How to Choose an Algorithm

• The three basic choices for algorithm are
  – Primal simplex
  – Dual simplex
  – Barrier

• The choice must be made more or less empirically.

• As a rule of thumb, dual simplex seems to outperform primal simplex, especially with degenerate problems, which often occur in practice.

• The availability of an advanced basis could determine the method.

• For large, sparse problems, barrier methods can outperform simplex.

• However, barrier methods cannot be “warm started” with an advanced basis.

• Starting with an advanced basis can be a significant advantage.
Parameter Settings

- LP solvers have numerous parameter settings for both linear programming and integer programming.

- Examples of LP parameters
  - Algorithm type
  - Pricing method
  - Basis refactorization frequency

- Examples of IP Parameters
  - Search order
  - Method of branching
  - Methods for tightening relaxations

- For a summary of interesting parameters in CPLEX, see the CPLEX/AMPL guide.