Introduction to Mathematical Programming
IE406

Lecture 22

Dr. Ted Ralphs

Reading for This Lecture

- Bertsimas Sections 10.2, 10.3, 11.1, 11.2

Solving Linear Programs in Practice

- The practice of linear and integer programming is as much art as science.
- There are many tradeoffs and considerations in developing and solving a model for a complex system.
- Developing a Model
  - For large, complex systems, there may be a large number of possible models.
  - Real systems have many, many constraints.
  - Timing considerations will determine how many constraints can realistically be modeled.
  - It is important to limit the number of constraints and variables as much as possible.

Developing a Set of Variables

- It is easy to develop models with far too many variables.
- The variables should represent independent decisions that need to be made.
- If the value of a variable can be inferred from the values of other variables, it can sometimes be eliminated.
- Exception: complex cost structures.
- Examples
  - Inventory Models
  - Fixed Charge Network Flow Models
- Additional variables can mean additional *linking constraints*.
- If the number of variables in the model is still large, consider column generation or try alternative pricing rules.

Developing a Set of Constraints

- The number of constraints determines the size of the basis, which in turn affects the efficiency of the simplex algorithm (and others).
- It is therefore crucial to eliminate *redundant constraints*.
- Example: The subtour elimination constraints.
  - Are all the subtour constraints needed?
  - Which ones might be eliminated?
- "Soft" constraints can sometimes be incorporated into the objective function with a penalty (Lagrangian relaxation).
- In real problems, constraints may have to be left out to simplify the model.

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 $a^1 x \le b_1$ *dominates* the constraint $a^2 x \le b_2$ if

$$\begin{aligned} a_i^1 &\ge a_i^2 \ \forall i, \ \text{and} \\ b_1 &\le b_2 \end{aligned}$$

- In this case, the dominated inequality could be deleted.
- We can also derive *implied bounds* for variables from each constraint $ax \le b$. If $a_0 > 0$, then

$$x_1 \le (b - \sum_{j:a_j>0} a_j l_j - \sum_{j:a_j<0} a_j u_j)/a_0$$

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 $\min\{c^\top x | Ax \geq b, l \leq x \leq b\}$,
  - 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):

	Rows	Columns	Iterations	Seconds
No preproc.	13,689	17,148	39, 429	10, 094
Preproc.	5,579	9,508	18,975	2,381

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.

CPLEX Version	Year	Time (seconds)
1.0	1988	57,840
3.0	1994	4,555
5.0	1996	3,835
6.5	1999	165

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.