Introduction to Mathematical Programming
IE496

Final Review

Dr. Ted Ralphs

Course Wrap-up: Chapter 2

- In the introduction, we discussed the general framework of mathematical modeling and defined the concept of a linear program.
- In Chapter 2, we discussed the geometry of linear programming.
  - We showed how to solve linear programs graphically and showed that the feasible region for an LP is a polyhedron.
  - A (bounded) polyhedron is described by its vertices, or extreme points.
  - Every linear program has an optimal solution that is an extreme point of its feasible region.
  - In order to solve a linear program, we derived an algebraic analogue of an extreme point, called a basic feasible solution.
  - We showed how to construct basic feasible solutions by choosing a basis matrix and solving an associated system of equations.
  - We showed that the set of extreme points and the set of basic feasible solution are the same and that therefore, we need only consider basic feasible solutions when optimizing.

Course Wrap-up: Chapter 3

- In Chapter 3, we first discussed optimality conditions.
  - From a given starting point, we showed how to construct a feasible, improving direction.
  - We derived that the reduced cost of variable j is the cost reduction that occurs from moving in the jth basic direction.
  - We therefore derived that if no variable has negative reduced cost, then the current solution is optimal.
- We then derived the simplex algorithm.
  - Moving as far as possible in the jth basic direction from a given basic feasible solution takes us to an adjacent basic feasible solution.
  - The idea of the simplex algorithm is to move from the current BFS, along an improving basic direction, to an improved basic feasible solution.

Course Wrap-up: The Basis Matrix

- The basis matrix was a central concept in the course. You should understand
  - How to tell if a given solution is basic.
  - How to form the basis matrix corresponding to a given basic solution.
  - How to compute the primal and dual solutions corresponding to a given basis.
  - How to compute the tableau corresponding to a particular basis matrix.
  - How to interpret all the elements of the tableau.
  - How to tell if a given basis matrix is feasible and/or optimal.
  - How a given basis matrix can be used to prove infeasibility or unboundedness of an LP.
  - The role the basis matrix plays in the simplex algorithm.
  - How to find an initial feasible basis matrix.

Course Wrap-up: Chapter 4

- In Chapter 4, we first derived duality theory.
- We showed how to derive the dual problem by using Lagrangian relaxation.
- We showed that the dual problem provides a lower bound on the optimal value of the primal problem (weak duality).
- In fact, we showed that the optimal solution of the dual is the same as that of the primal (strong duality).
- Using duality, we derived the concept of complementary slackness and derived optimality conditions based on this concept.
- We interpreted these optimality conditions geometrically and related this to the Farkas Lemma.

Course Wrap-up: Chapter 4 (cont.)

- After duality theory, we derived the dual simplex method based on the idea of maintaining dual feasibility instead of primal feasibility.
- Note that both versions of simplex always maintain complementary slackness.
- Finally, we introduced the concept of extreme rays and the recession cone.
- We showed that every polyhedron has a unique description in terms of extreme rays and extreme points.
- We also showed how to characterize unbounded LPs.

Course Wrap-up: Chapter 5

- In Chapter 5, we saw how to perform basic sensitivity analysis.
- We determined how to tell whether the basis remains feasible after changing problem data (locally).
- We determined how to tell if the basis remains feasible after adding variables and constraints.
- Note that all of this analysis depends only on knowing how to compute the entries of the tableau for a given basis.
- Finally, we looked at global dependence on the cost and right-hand side vectors and derived the parametric simplex algorithm.

Course Wrap-up: Chapter 6

- In Chapter 6, we looked at methods for dealing with LPs that have large numbers of variables and constraints.
- This led to the idea of delayed column generation.
- Cutting plane methods, on the other hand, generate violated constraints "on the fly".
- These methods are critical to solving large LPs, especially in the context of integer programming.

Course Wrap-up: Chapter 7

- In Chapter 7, we discussed network flow problems.
- We defined the concept of a graph and a network.
- We defined what a minimum cost network flow problem is and derived an analogue of the simplex algorithm for solving such problems.
- We showed how to improve on this algorithm with a combinatorial counterpart.
- We discussed some related problem and derived algorithms for solving them
  - Assignment Problem
  - Shortest Path Problem
  - Minimum Spanning Tree Problem

Course Wrap-up: The Primal-Dual Algorithm

- The primal-dual algorithm can be used to solve general linear programs.
- Suppose we have an LP in standard form and assume without loss of generality that $b \geq 0$.
- The idea is to start with a feasible dual solution and try to construct a primal solution that obeys complementary slackness.
- This is done by attempting to solve $Ax = b$ with only the variables having zero reduced cost allowed to enter the basis.
- If we succeed, then the primal solution is optimal.
- Otherwise, we change the dual prices and continue.
- When applied to certain problem classes, this algorithm can be implemented very efficiently.

Course Wrap-up: Chapters 10 and 11

- To wrap up the course, we showed how the tools we have developed can be used to solve integer programs.
- Certain integer programs are "easy," but these are rare.
- Most integer programs are difficult to solve.
- Rounding doesn't really help.
- The main algorithmic tool for solving integer programs is branch and bound.
- This method depends critically on the computation of lower bounds.
- This can be done with linear programming.
- The idea is to approximate the convex hull of feasible solutions to the integer program as closely as possible.
- This is why formulation is so important in integer programming.
- The formulation can be augmented by generating additional valid inequalities.