

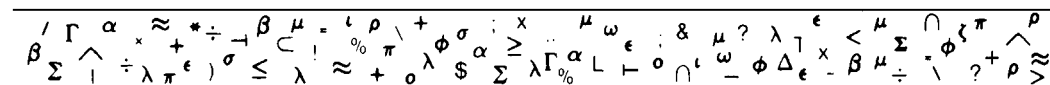
The Teachers' Forum: Teaching Nonlinear Programming Using Cooperative Active Learning

Leon Lasdon

*MSIS Department
College of Business Administration
The University of Texas at Austin
Austin, Texas 78712*

Judith S. Liebman

*Department of Mechanical and Industrial
Engineering
University of Illinois at Urbana-Champaign
Urbana, Illinois 61801*



In teaching nonlinear programming (NLP) to undergraduate and graduate students majoring in engineering or operations research, we have changed our instructional objectives, course content, and approaches to teaching because of recent dramatic improvements in optimization software and computer hardware. Our primary instructional innovation is to use cooperative active learning methods rather than a traditional lecture format.

Lasdon has taught a master's- and doctoral-level nonlinear programming course once a year since 1964, first in the operations research department of the management school at Case Western Reserve University and since 1977 in the management science and information systems (MSIS) department of the College of Business Administration at the University of Texas at Austin. The MSIS course

originally had 15 to 20 students, mostly operations research majors, with a few engineering students. Now, the size of the MSIS doctoral program in operations research has decreased, so there are about 10 to 20 students in the course, with engineering students outnumbering the OR majors.

Liebman has taught in the industrial engineering program at the University of Illinois at Urbana-Champaign since 1972. In the earlier years, she taught a nonlinear programming course at the master's- and doctoral-level primarily for engineering students. Recently, she has been teaching an introductory course to undergraduate engineering students that includes some coverage of nonlinear programming. For the last four years she has been developing and using active learning methods for teaching a variety of operations research

topics at both the undergraduate and graduate levels.

Over time, our instructional objectives for NLP courses have changed. In the 1970s and 1980s, commercially available optimization software was scarce, expensive, and much less effective than it is today. Computer hardware was more expensive and far less powerful. Hence students studying nonlinear optimization needed to learn how to implement algorithms in addition to selecting them. Now the need to design or implement has been greatly reduced. We believe that the instructional emphasis should shift away from theory and algorithms and more toward modeling, evaluating, and using existing software, interpreting solutions, and understanding decision support requirements. Liebman [1998] has stressed the need to relate course content to instructional goals.

We highly recommend the use of active learning in the classroom. Active learning can be cooperative when students work in teams. During the process of cooperative active learning, student teams solve problems and discuss theoretical issues during the class, with instructors acting as facilitators. Instructors lecture briefly, if at all, only to clarify issues raised by the teams or to explain material inadequately covered in a preparatory reading assignment. Educational research [West, Farmer, and Wolffe 1991] indicates that students learn more thoroughly and retain ideas longer using this approach. The use of an active-learning approach in operations research education is described by Liebman [1994, 1996a, 1998] and Fellers [1996]. Johnson, Johnson, and Smith [1991] and Meyers and Jones [1993] have written books pro-

moting the use of active learning in college classrooms.

Changes in the Optimization Environment

Today, optimization software is readily available, powerful, and easy to use. In a recent publication, Lasdon, Plummer, and Waren [1996] describe NLP algorithms, software, and applications. Most NLP applications today use an algebraic modeling language, a spreadsheet optimizer, or a stand-alone optimizer whose model is coded in Fortran or C. The algebraic languages with NLP capabilities include AMPL [Fourer, Gay, and Kernighan 1993], GAMS [Brooke, Kendrick, and Meeraus 1992], and LINGO [Schrage 1991]. All have interfaces to at least two NLP solvers, including CONOPT [Drud 1994], LSGRG2 [Smith and Lasdon 1992], and MINOS [Murtagh and Saunders 1982]. All are capable of solving large sparse problems and are reliable and efficient.

Because these tools are now available, students will rarely need to design or implement an NLP algorithm. Hence the educational focus should shift to learning how to build, solve, and interpret results of models using this software. Teaching the derivation and properties of algorithms is still important for future practitioners, because no class of algorithms dominates. Practitioners must be prepared to choose an appropriate solver, tune that solver for best performance, and take appropriate action if the solver should fail. We also believe that truly educated students, particularly those majoring in operations research at the graduate level, should understand the fundamental tools of their profession in depth.

The most widely available spreadsheet optimizer is the Excel Solver, bundled with each copy of Microsoft Excel. Between 20 and 25 million copies are on desktops today, and many students have a copy on their own PCs. The Excel Solver uses GRG2 [Lasdon et al. 1978] to solve nonlinear models represented by Excel formulas and data. It includes a dense-matrix simplex code for linear problems and a branch-and-bound capability for mixed-integer linear or nonlinear problems. It is widely used in teaching operations research to MBAs, and several recent texts [Ragsdale 1995; Winston and Albright 1996] use it exclusively in their LP and NLP chapters. Maximum problem size is 200 variables and 100 constraints, not including simple bounds on the variables. This is sufficient for virtually all examples used in a course, as well as many practical applications. For a detailed discussion of spreadsheet solvers see Fylstra et al. [forthcoming].

Many efficient stand-alone NLP codes are now available, including those mentioned above. See also More and Wright [1993] and Lasdon, Plummer, and Waren [1996, section 4.5]. Many are available at low prices for academic use, either through the authors or as a part of such software libraries as National Algorithms Group (NAG), Harwell, IMSL, and Optima. Most of these codes are written in Fortran and implement SQP, the successive quadratic programming algorithm [Powell 1986].

The Graduate Course

In establishing course goals, Lasdon starts from an understanding of how students may use the course material in the

future. Undergraduate- or master's-level engineering and operations research majors may become practitioners using optimization or may study for higher degrees. With today's shrinking academic job market, most operations research and engineering doctoral students take industrial positions. Hence we need to help students become informed users of optimization. This means they should be able to recognize profitable optimization applications, formulate and implement an appropriate model, choose and tune an effective solver, and analyze, interpret, and communicate results to clients. Either before or after a graduate NLP course, many students take other courses that stress optimization modeling and applications within a particular problem class, for example, water resources, chemical processes, electric power, or mechanical design.

A secondary goal of a graduate level NLP course is to provide those students who will do research involving NLP applications or (less often) research on NLP itself with an appropriate background. These students need to understand basic NLP theory and the origins and properties of important NLP algorithms, and they should be introduced to the broad spectrum of optimization modeling systems now available.

Learning Objectives for Lasdon's Graduate Course

In Lasdon's one-semester introductory course for graduate students, he has the following learning objectives for the students:

Theory: Understand the derivation and uses of the first-order necessary conditions for optimality, second-order optimality

conditions, saddle points, and the Lagrangian dual problem. Also, understand convergence and basic convexity results, including definitions and properties of convex sets and functions, and theorems showing when a local optimum is global.

Algorithms: Understand the derivation, convergence rates, and comparative advantages of the following classes of algorithms; use given implementations of these algorithms; and observe and analyze the results:

- One-dimensional search: inexact line searches using polynomial interpolation;
- Unconstrained multidimensional: steepest descent, Newton methods, quasi-Newton methods, and conjugate gradient methods;
- Constrained multidimensional: generalized reduced gradient (GRG), successive quadratic programming (SQP), successive linear programming (SLP), penalty and barrier methods (exact and inexact), and interior point methods.

Modeling Systems: Through the following activities, begin to understand the tools practitioners use to build NLP models and how they build models using them:

- Learn to use stand-alone Fortran or C NLP solvers for solving problems coded in those languages, the Excel Solver, and algebraic modeling languages (GAMS or AMPL or both); understand their advantages and limitations;
- Improve modeling skills by formulating a variety of problems in several modeling languages, solving, then analyzing and understanding the solution;
- Learn principles of good modeling prac-

tice and what to avoid.

Applications: Learn about several important NLP application areas, for example,

- Process industries (gasoline blending, process unit models, refinery models),
- Electric power (hydroelectric planning, optimal load flows),
- Financial (Markowitz asset allocation, multi-period models, robust optimization),
- Optimal control, and
- Water resources.

Since there are many NLP algorithms, the instructor must decide which to teach. They should include the major classes of algorithms that are now in software libraries, that are copied coupled to modeling systems, that are widely used otherwise, or that are promising for the future.

Lasdon includes SLP because it is widely used in the process industries (several of his students are chemical engineers), and it can solve very large problems. Interior point algorithms are not yet widely used in solving nonlinear problems, but they are good for quadratic programming, promising for general NLP, familiar to students who have taken LP, and are closely related to SQP. He includes barrier methods because they provide a basis for understanding interior point methods. Finally, exact penalty functions are used in SQP and SLP.

Table 1 lists the topics covered in Lasdon's graduate course in the fall semester of 1996 in the order covered. Each class session is 75 minutes long. He conducted the sessions on spreadsheet optimization and GAMS in a computer lab. The text used was Luenberger's *Linear and Nonlinear Programming* [1984]. For further

TEACHERS' FORUM

Topic	Sessions
Basic definitions, convexity, introduction to unconstrained optimization	1
Steepest descent, Newton methods, convergence and rate of convergence	2-3
Trust-region Newton methods, quasi-Newton, and conjugate gradient methods	4-5
Inexact line search, convergence theorems, finite difference derivatives	6
Spreadsheet optimization	7
The GAMS modeling language	8
Using callable Fortran solvers	9
Necessary and sufficient conditions for constrained problems	10-11
Saddle points and Lagrangian duality	12-13
Reduced gradient algorithms	14-15
Successive quadratic programming algorithms	16-17
Exact and inexact penalty and barrier algorithms	18
Successive linear programming, algorithm comparison	19
Interior point methods	20
MILP and MINLP modeling with binary variables	21-22
NLP applications	23-27

Table 1. Topics for Lasdon's graduate course in fall 1996.

information on the time allotted to these topics and the contents of the related readings packet, see Lasdon's courses web page: www.utexas.edu/courses/lasdon.

Seventeen of the sessions concerned algorithms and theory, and 10 concerned modeling systems and applications. Since most students will become NLP users

rather than researchers, the time devoted to modeling and applications might be increased. This issue is far from being resolved. Lasdon arrived at the current allocation because engineering faculty send their doctoral students to this course primarily for the coverage of theory, algorithms, and modeling systems as a basis for research in which NLP will play a role. Further, NLP modeling often requires a great deal of domain-specific knowledge, and courses in each engineering area include optimization modeling, for example in chemical engineering, water resources, transportation, power systems, and aeronautical engineering. Allocating 60 percent of class sessions to algorithms and theory also seems appropriate for operations research majors when no follow-on NLP course is available.

Lasdon chose Luenberger's text despite its 1984 publication date because it has excellent coverage of NLP theory and algorithms. It is accessible to students without a course in analysis, requiring only basic matrix calculus and linear algebra. The derivations of the first- and second-order optimality conditions in Chapter 10 are especially recommended because they are geometrically motivated and intuitive yet rigorous. Luenberger discusses the algorithms mentioned in Table 1 except SLP and interior-point methods. Some other texts to consider are those of Bazaraa, Sherali, and Shetty [1993] and Bertsekas [1995]; Lasdon switched to the Bazaraa text in fall 1997. Nonetheless, for a modern course with good coverage of computational issues, modeling systems, and applications, other materials are needed. Lasdon provides a readings packet. It in-

cludes handwritten notes, computer programs and output, sections of books, and journal articles.

To illustrate Lasdon's use of active learning ideas in the graduate course, we describe some class session plans. These contain the schedule for each class: group activities, class discussions, brief lectures, and homework and readings for the next class. They are distributed at the start of each session. Exercises to be done collaboratively by groups in class should not be distributed earlier, because the more eager students will do the work alone ahead of time instead of developing and sharing insights cooperatively.

The First Session for Lasdon's Graduate Course

The plan for the course lists the following examples of group activities:

- (1) The group reviews problem(s) assigned for homework. Each group member brings in his or her proposed solution. The group compares and discusses these solutions, and writes the group's solution to the problem on the board. Individual solutions are not modified and are handed in to be graded.
- (2) The group discusses a reading assignment, answers questions within the group, and prepares one or more questions for class discussion.
- (3) The group reads and comments on draft project reports from group members.
- (4) The group formulates and solves an assigned NLP problem using the Excel solver, GAMS, or a Fortran NLP code and presents the model and solution in class.
- (5) The group solves a problem given during class.
- (6) The group creates and solves a nonlin-

ear programming problem with specific characteristics.

- (7) The group prepares and delivers a class presentation on an NLP application area. There are several NLP application papers in the readings packet.

The first activity in session 1 is to form learning groups of three students each and discuss the course format. The members of each group are asked to assign the roles of discussion leader, note-taker, and spokesperson. These roles are rotated at the start of each class (15 minutes).

For their second activity, the groups discuss the following terms and write their mathematical definitions on the board. A class discussion of the results follows (25 minutes):

- General optimization problem,
- Global solution,
- Strict global solution,
- Local solution,
- Nonlinear program,
- Unconstrained optimization problem,
- Nonsmooth optimization problem, and
- Mixed integer program.

For the third activity, the groups meet to discuss and write on the board the first-order necessary conditions, the second-order necessary conditions, and the second-order sufficient conditions for an unconstrained minimum of a twice differentiable nonlinear function $f(x)$, followed by a class discussion of the results (20 minutes).

In this small class, groups of three are ideal. Initially, students form their own groups. This usually results in groups of friends, often of the same ethnic or disciplinary background. After about 10 classes, Lasdon reorganizes the groups so

that each group contains both vocal and quiet students. In general, groups of three to four students appear to work well. They are small enough to encourage all students to participate, yet large enough to bring in diverse points of view.

Lasdon has used activities of types 1, 2, 4, and 5 extensively. Homework is assigned for most classes and is reviewed as the first group activity. This provides immediate feedback, allows students to function as teachers, and is far superior to our previous practice of assigning weekly homework, which was handed back with solutions one week later. Now, we provide solution sheets after the class discussion and tell students not to modify their written solutions. We grade homework individually. Midterm and final examinations also provide individual accountability.

In activities 2 and 3, the students use group problem solving. Instead of telling the students what these basic terms mean, we ask them to define them. A lively discussion occurs within each group, and most of the resulting definitions are approximately correct. The ensuing class discussion is also lively, and the instructor can elaborate or correct where needed. Similar comments apply to activities 4 and 5. Most graduate students have seen this material before; and here they are forced to recall it with help from other group members.

The detailed session plans add structure and organization for both the students and the instructor. The instructor is also forced to estimate the time required for each activity. Lasdon found that he consistently underestimated these times, often by a factor of two, because learning groups take

longer to discuss a topic than an instructor would take to lecture on it. As a consequence, fewer topics (or less depth in some topics) may be covered in an active learning course than in a traditional lecture course, but educational research [West, Farmer, and Wolffe 1991] indicates that students learn more thoroughly and retain the material longer. The group discussions stimulate a deeper level of mental processing than passive listening.

Generic questions, such as "discuss the derivation of the SQP algorithm on page 27," are not as effective as specific and detailed questions designed to highlight important points. Some examples follow, taken from the plan for the class session on interior point algorithms, using Lasdon and Plummer [1995] as the assigned reading:

- Why is the barrier function introduced in equation 4. What is gained?
- In the Kuhn-Tucker conditions on page 323 of the paper, derive equations 11–12.
- The group leader explains the derivation of the Newton equations (15–19) and the modified system (20).
- In step 4 of algorithm PD (primal-dual) on page 323, why is it necessary to maintain both the primal and dual variables strictly within bounds?
- What are special properties of the coefficient matrix in the linear system (20)?
- The group leader explains the logic behind step 5 of the PD algorithm.
- The group discusses the problems involved in evaluating second derivatives of problem functions when the problem is coded as a Fortran GCOMP.
- From the results on pages 328–330, what do you think are the advantages of the PD

algorithm? What are the disadvantages?
—What features need to be added to this PD algorithm to make it provably convergent?

Instructors lecture briefly, if at all.

As learning groups undertake the scheduled activities, the instructor and any teaching assistants move from group to group listening. Occasionally the instructor or assistant joins a discussion if it is clear that no one in the group has a good answer to the question under discussion. As long as the energy level of the discussion remains high, a group is generally making progress. After 15 or 20 minutes, discussion usually wanes. Some open-ended questions might stump some groups, but they still introduce important issues and force their active consideration.

When instructors ask a class as a whole for answers, only a small subset of students may respond. To ensure more even participation, it is better to call upon groups at random, with the group spokesperson answering for the group. For questions with complex answers, instructors can call on several groups to provide parts of the answer and then provide missing information themselves if necessary. Instructors can also use these opportunities to provide minilectures (a few minutes long) on points that need elaboration.

The Undergraduate Course

In Liebman's one-semester introductory operations research course for engineering undergraduates, only two or three two-hour classroom sessions can be devoted to nonlinear programming. The text for the

course is *Introduction to Operations Research* [Hillier and Lieberman 1995]. The students are expected to do the following:

- Know the mathematical form of a general nonlinear optimization problem;
- Understand the following examples of nonlinear programming models: the product-mix problem with price elasticity, transportation problem with volume discounts, and the portfolio selection problem with nonlinear function for risk as constraint or objective;
- Be able to develop a mathematical model for a nonlinear optimization problem described in words;
- Know the definitions of local maximum, global maximum, concave function, convex function, and gradient;
- Know how to use GAMS, GINO, or Excel to solve nonlinear programming problems;
- Know how to apply the following calculus-based optimization methods: one-dimensional (setting first derivatives to zero, checking second derivatives) and multi-dimensional (Lagrange multiplier method for equality constrained problems and Karush-Kuhn-Tucker conditions for general NLP); and
- Know how to carry out gradient search (steepest descent).

Liebman first introduces students to the general form of nonlinear programming problems, some simple applications, and some basic definitions. They extend their previous knowledge about using commercial software to solve linear problems to the solution of nonlinear problems. Then they learn the calculus-based criteria for optimal solutions and, finally, how to undertake a gradient-based steepest-descent

search. In learning steepest descent, students further their understanding of how commercial codes work.

They usually need to spend about 10 to 16 hours (four to six in class, two to four in reading, and four to six in homework) to learn this material. A major reason they can accomplish so much so quickly is that the hours they spend in the classroom are spent actively learning, not passively listening.

The overall approach Liebman takes in the undergraduate course is to design activities to build students' understanding by using small problems and providing graphical illustrations in two and three dimensions whenever possible.

Undergraduate Activities for Two or Three Two-Hour Sessions on Nonlinear Programming

In the first activity, on applications of nonlinear programming, Liebman asks students to do the following:

—Describe an application in which the objective function and constraint functions are algebraic and have continuous first and second partial derivatives [Hint: there are many examples in engineering optimization];

—Describe an application in which the functions might not be expressible in algebraic form [Hint: sometimes computer simulations are used to model the behavior of an engineering system]; and

—Develop a nonlinear optimization model for a selected word problem from the text.

In the second activity, on reviewing and understanding basic definitions, she asks them to do the following:

—Find the gradient $\nabla f(\mathbf{x})$ for $f(\mathbf{x}) = (x_1 - 2)^2 + x_1 x_2 + (x_2 - 3)^2$,

—For the $f(\mathbf{x})$ used in part A, compute its Hessian $H(\mathbf{x})$;

—Draw a picture of a function $f(x)$ over the range from $x = a$ to $x = b$ that has at least one local max, at least one local min, a global max, an inflection point, and an unbounded minimum and describe what all the finite maxima, minima, and inflection points have in common;

—Draw a picture of a convex function of one variable;

—Draw a picture of a convex function of two variables;

—Explain whether a straight-line function is concave or convex or both;

—For the Hessian computed earlier today, explain whether its $f(\mathbf{x})$ is a convex function;

—Draw a picture of a convex set and explain whether the set is defined by $\{x \mid x_1 + x_2 \leq 4, x_1, x_2 \geq 0\}$ is convex and whether the feasible region for a linear programming problem is convex.

For the third activity, Liebman asks students to use the necessary and sufficient conditions for optimality in unconstrained problems and do the following:

—Consider the function $f(x) = (x - 3)^2$. Use the necessary conditions to solve for all stationary points. For each point found, use the sufficient conditions to indicate whether it is a local max, a local min, or an inflection point.

—If a differentiable function is convex, a point at which the first derivative is zero yields a global min. If the function is concave, what do you conclude about a point at which the derivative is zero?

—Use the necessary and sufficient conditions to find the minimum point for $f(\mathbf{x}) = x_1^2 + 3x_1x_2 + 4x_2^2 - x_1 - 2x_2$.

For the fourth activity, on performing a gradient search, Liebman assigns the following activities:

—Attached is a contour plot of the function $f(x_1, x_2) = (x_1 - 2)^2 + (x_2 - 3)^2$ which is to be minimized using steepest descent. Mark your starting point and all subsequent points on the attached graph.

—Also on the attached sheet is a partly worked example of using gradient search to find the minimum of this function. Fill in the blanks.

—Most commercial optimization software uses methods related to steepest descent. A major complication is the presence of constraints. How would you suggest using steepest descent in the presence of constraints?

For the fifth activity, on applying the Lagrange multiplier method, Liebman asks the students to answer the following questions:

—Use the Lagrange multiplier method to find the point on the plane $A x_1 + B x_2 + C x_3 = D$ that is closest to the origin ($x_1 = x_2 = x_3 = 0$). [Hint: the problem can be modeled with the following

$$\begin{aligned} \text{NLP: } \min & (x_1 - 0)^2 + (x_2 - 0)^2 + (x_3 - 0)^2 \\ \text{s.t. } & A x_1 + B x_2 + C x_3 = D. \end{aligned}$$

Note that the distance from the origin is actually the square root of $(x_1 - 0)^2 + (x_2 - 0)^2 + (x_3 - 0)^2$, but the optimal values for the x s will be the same no matter whether we minimize this or its square root. The objective function is convex. Thus is your solution a maximum or a minimum point?

—Note that each constraint has a Lagrange multiplier associated with it. At the optimal solution, the Lagrange multi-

plier is the shadow price (marginal cost) associated with that constraint. That is, a multiplier shows the rate of change of the objective function per unit increase in the right-hand side of its corresponding constraint. For this problem, as D increases, is the plane moving closer to or away from the origin?

For activity 6, she assigns another Lagrange multiplier problem:

—Using Lagrange multipliers, solve the following problem:

$$\begin{aligned} \max & 3 x_1^2 + x_2^2 + 2 x_1 x_2 + 6 x_1 + 2 x_2 \\ \text{s.t. } & 2 x_1 - x_2 = 4. \end{aligned}$$

[Hint: first set up the simultaneous equations and call me over. If your equations are correct, I will give you the solution.

Then you must check to see if you have a max or a min. Do this by figuring out whether the objective function is convex or concave by checking the Hessian. If your solution is a min, use the attached plot of the function and the constraint to see where the max might lie.]

For activity 7, on using the Karush-Kuhn-Tucker conditions, she asks students to work on the following problem:

—Set up the Karush-Kuhn-Tucker conditions for

$$\begin{aligned} \max f(\mathbf{x}) &= 3 x_1 + x_2 \\ \text{s.t. } g_1(\mathbf{x}) &= x_1^2 + x_2^2 \leq 5 \\ g_2(\mathbf{x}) &= x_1 - x_2 \leq 1 \\ x_1, x_2 &\geq 0. \end{aligned}$$

—Is the feasible region convex? Is the objective function concave?

—Use the Karush-Kuhn-Tucker conditions and your answer to A and B to show that the point $x_1^* = 2, x_2^* = 1$ is an optimal solution.

—What do the values of the Lagrange multipliers u_1 and u_2 predict if the right-hand side of their corresponding constraints is increased?

In activity 1, Liebman asks students to think about examples in which NLP can be applied, in addition to the examples provided by the text. She also asks them to develop a model for a word problem. Having students work in a group to develop a model is extremely effective. Students with natural modeling skills have much to share with students for whom modeling is a black art. The purpose of activity 1 is to help students develop a mental framework in which to think about nonlinear optimization and to motivate students by the use of relevant applications.

In activity 2, having students use the basic definitions significantly increases their understanding. The review of unconstrained optimization in activity 3 is essential; students barely remember the topic from their calculus courses.

Students practice gradient search in activity 4 by filling in the blanks in a partially worked example. To convey the

We recommend the use of active learning.

metaphor of a search process taking place on hilly terrain, Liebman also asks students to plot the progress of their numerical search on a contour plot provided to them. The students think about the relationship between gradient search and the approaches used by common commercial computer software as they consider how they might handle bumping into con-

straints while using steepest descent. In a related homework assignment, the students solve several nonlinear optimization problems using commercial software (usually GAMS, Excel, or GINO).

In activity 5, students practice the Lagrange multiplier method and the interpretation of Lagrange multipliers. In activity 6, the students deal with a problem in which the Lagrange multiplier method yields a minimum instead of the sought-for maximum. They are given a plot of the problem's objective function contours and constraints to help them identify what is going on. To save class time on problems involving the solution of simultaneous equations, students can be asked to set up the equations for which the instructor provides the solutions.

In activity 7, students set up the Karush-Kuhn-Tucker conditions for a two-variable, two-constraint problem and observe whether or not the feasible region is convex and the objective function concave. To save class time, the instructor gives the students a solution and asks them to use the Karush-Kuhn-Tucker conditions to show if the given solution is optimal. Finally the instructor asks students to interpret the meanings of the two Lagrange multiplier values with respect to how the objective function value might change if the right-hand side of the constraint changes.

In all of these activities, students are figuring out for themselves what the instructors used to put on the board during lectures. Although it takes more time for them to do it themselves, cognitive psychologists predict that the resulting long-term learning is better. In contrast to

Lasdon's experiences at the graduate level, Liebman found that time limits for the group discussions did not work well in the undergraduate course. Groups varied with respect to the time they needed to complete activities, probably because undergraduate students came to class less prepared. The undergraduate course activities were such that prepared groups could move through them fairly rapidly, but unprepared groups needed more time. Answers were shared between the groups by having the first group to complete an activity put their answers to the activity on the board. Later-finishing groups record on the board only answers that differ.

Evaluations and Conclusions

What do students think about this way of teaching? In the graduate course, Lasdon gave the class of eight students a midsemester feedback questionnaire, administered by the UT Center for Teaching Effectiveness. There were 15 questions dealing with student satisfaction with the course and the instructor, graded on a five-point scale (5 = excellent). Averages for all but one question ranged from 3.75 to 4.625. The administrator had observed a previous class and had provided some useful advice. She noted that none of the students had encountered the active-learning approach before, and they were still getting used to it. This reflects the fact that courses in engineering and most PhD-level business courses are taught mainly by lecture. Two or three students expressed strong reservations, asking for more lectures. Final average scores on five questions, including overall course and instructor evaluations, were in the range 3.6–4.0. This is not markedly different

from the two previous years, except for one measure, "helped to think for myself." The average scores for three classes taught between 1994 and 1996 were 3.0, 2.8, and 3.9, the first two taught by lecture, the last using cooperative active learning. Despite the small numbers of responses (5, 8, and 8), we believe this indicates a real improvement, and it agrees with Liebman's experience.

In the undergraduate course, Liebman first used active learning techniques when she taught two sections of the course simultaneously. One section used learning groups and had no lectures, and the other section had lectures with cooperative active learning only through turn-to-your-neighbor discussions [Liebman 1996a] every 10 to 12 minutes. Students were allowed to choose which section to enroll in, that is, to choose the learning environment that suited them best. At the end of the semester, students gave both sections of the course high ratings on the standard campus instructional evaluation form and in focus-group evaluations. The focus-group evaluations were carried out by a professional member of the university's instructional resource staff who asked focus groups in each section to list the strengths of the course, the ways in which the course could be improved, and so forth. The most interesting difference between the learning-group section and the lecture section was that the learning-group section listed as strengths the development of their own learning skills and interests and the lecture section listed as strengths the interest of the professor in students, the professor's knowledge of the material, the professor's presentation skills, and so

forth. Thus students in the learning groups recognized that they had assumed the responsibility for learning and believed that their own learning skills had improved. Students in the lecture section, on the other hand, still gave the instructor the major responsibility for their learning accomplishments.

When the undergraduate course (required for IE majors and scheduled in their curriculum for the fall semester) is taught in the spring semester, there are only enough students for one section. Liebman teaches that section using learning groups, without lectures, to the comfort of some students and the discomfort of others. Students who are willing to do the assigned reading ahead of time come to class prepared and generally prefer the use of learning groups. Students who would rather come to lectures without preparation or who are less disciplined in their time management are not as well served. For situations in which a majority of students fall into this second category, instructors may wish to lecture and generate active learning by using turn-to-your-neighbor discussions every 10 or 12 minutes.

Conclusions

In summary, we both believe that the use of active cooperative learning greatly enhances students' understanding and mastery of material. Instructors can use active cooperative learning to augment or completely replace lectures, both in undergraduate and graduate courses. The energy levels in the classroom rise visibly when active-learning techniques are employed, and students display greater enthusiasm and enjoyment. To obtain these benefits,

students must prepare for each class. We strongly recommended this approach for teaching other areas of operations research.

References

- Bazaraa, M.; Sherali, H.; and Shetty, C. M. 1993, *Nonlinear Programming, Theory and Algorithms*, John Wiley and Sons, New York.
- Bertsekas, D. P. 1995, *Nonlinear Programming*, Athena Scientific, Belmont, Massachusetts.
- Brooke, A.; Kendrick, D.; and Meeraus, A. 1992, *GAMS, A User's Guide*, Boyd and Fraser, Danvers, Massachusetts.
- Drud, A. 1994, "CONOPT—A large-scale GRG code," *ORSA Journal on Computing*, Vol. 6, No. 2, pp. 207–216.
- Fellers, J. W. 1996, "Using the cooperative learning model to teach students 'people skills,'" *Interfaces*, Vol. 26, No. 5, pp. 42–49.
- Fourer, R.; Gay, G.; and Kernighan, B. 1993, *AMPL: A Modeling Language for Mathematical Programming*, Boyd and Fraser, Danvers, Massachusetts.
- Fylstra, D.; Lasdon, L.; Watson, J.; and Waren, A. forthcoming, "Design and use of the Microsoft Excel Solver," *Interfaces*.
- Hillier, Frederick S. and Lieberman, Gerald J. 1995, *Introduction to Operations Research*, McGraw Hill, New York.
- Johnson, David W.; Johnson, Roger T.; and Smith, Karl A. 1991, *Active Learning: Cooperation in the College Classroom*, Interaction Book Company, Edina, Minnesota.
- Lasdon, L.; Waren, A.; Jain, A.; and Ratner, M. 1978, "Design and testing of a generalized reduced gradient code for nonlinear programming," *ACM Transactions on Mathematical Software*, Vol. 4, No. 1 (March), pp. 34–50.
- Lasdon, L. and Plummer, J. 1995, "Primal-dual and primal interior point algorithms for general nonlinear programs," *ORSA Journal on Computing*, Vol. 7, No. 3, pp. 321–332.
- Lasdon, L.; Plummer, J.; and Waren, A. 1996, "Nonlinear programming," in *Mathematical Programming for Industrial Engineers*, eds. M. Avriel and B. Golany, Marcel Dekker, New York, pp. 385–485.
- Liebman, Judith; Lasdon, Leon; Schrage, Linus; and Waren, Allan 1986, *Modeling and Optimization with GINO*, Boyd and Fraser, Danvers, Massachusetts.
- Liebman, Judith S. 1994, "New approaches in

- operations research education," *International Transactions of Operational Research*, Vol. 1, No. 2, pp. 189–196.
- Liebman, Judith S. 1996, "Promoting active learning during lectures," *OR/MS Today*, Vol. 23, No. 6 (December), p. 8.
- Liebman, Judith S. 1998, "Teaching operations research: Lessons from cognitive psychology," working paper, University of Illinois.
- Liebman, Judith S. forthcoming, "Professor Liebman, will this be on the test?" *OR/MS Today*.
- Luenberger, D. G. 1984, *Linear and Nonlinear Programming*, second edition, Addison-Wesley, Menlo Park, California.
- More, J. J. and Wright, S. J. 1993, *Optimization Software Guide*, SIAM, Philadelphia, Pennsylvania.
- Meyers, Chet and Jones, Thomas B. 1993, *Promoting Active Learning: Strategies for the College Classroom*, Jossey-Bass Publishers, San Francisco, California.
- Murtagh, B. A. and Saunders, M. A. 1982, "A projected Lagrangian algorithm and its implementation for sparse nonlinear constraints," *Mathematical Programming Study Sixteen*, pp. 84–117.
- Powell, M. J. D. 1986, "Convergence properties of algorithms for nonlinear optimization," *SIAM Review*, Vol. 28, No. 4, pp. 487–500.
- Ragsdale, C. T. 1995, *Spreadsheet Modeling and Decision Analysis*, Course Technology Inc., Cambridge, Massachusetts.
- Rosenthal, R. 1992, "A GAMS Tutorial," in *GAMS, A User's Guide*, Boyd and Fraser, Danvers, Massachusetts.
- Schrage, L. 1991, *Lindo, an Optimization Modeling System*, Boyd and Fraser, Danvers, Massachusetts.
- Smith, S. and Lasdon, L. 1992, "Solving large sparse nonlinear programs using GRG," *ORSA Journal on Computing*, Vol. 4, No. 1, pp. 3–15.
- West, Charles K.; Farmer, James A.; and Wolff, Phillip M. 1991, *Instructional Design: Implications from Cognitive Science*, Allyn and Bacon, Needham Heights, Massachusetts.
- Winston, Wayne and Albright, S. Christian 1996, *Practical Management Science*, Duxbury Press, Belmont, California.