Optimize Me! – J. Warfel

Purpose

Many important problems in industrial engineering and operations research can be expressed as mathematical programs. IE/OR professionals put significant effort into finding algebraic formulations that reflect the limitations of reality (through constraints) and their aspirations in solving the problem (through the objective function). For small problems, or those with a special structure, there may be a clever mathematical way to find a solution; however, for almost all interesting problems, the assistance of a computer is required. In this activity, the students follow the steps of an IE/OR professional in solving a problem in this fashion. They start with the algebraic formulation. Then, they transfer this into a format that can be solved with a computer-based optimization solver. Finally, they use the computer-generated solutions to improve their model's representation of reality, hence improving their solutions.

Overview

This activity consists of three lessons. It builds on previous lessons about formulating linear programs. All of linear programs in this activity are based on data sets about the prices and nutritional quality of food from McDonald's, as well as the nutritional requirements of teenage boys and girls.
In the first lesson, students formulate small linear programs (with four decision variables). After formulating the problem algebraically, they create a model in AMPL (a modeling language for mathematical programs) and solve it with an online solver.
In the second lesson, students formulate linear programs with arbitrarily many decision variables by using summation notation. They modify their previous AMPL model to reflect this change, and then use it to solve a problem with 99 decision variables.
In the third lesson, students add constraints to the linear program they formulated in the second lesson. They solve their new model, then use their results to determine whether it is possible to eat a healthful diet only by consuming food from McDonald's.

Student Outcomes

SWBAT:
Formulate a linear program in AMPL.
Solve a linear program with an online solver.
Interpret the solution of an optimization problem.
Evaluate how well a mathematical model represents reality.
Improve a mathematical model by adding constraints to it.

Time

This lesson will require at least three full class periods. It may require more, depending on the students' basic computer skills and their ability to formulate linear programs.

Reach for the Stars is a GK-12 program supported by the National Science Foundation under grant DGE-0948017. However, any opinions, findings, conclusions, and/or recommendations are those of the investigators and do not necessarily reflect the views of the Foundation.

Level
This lesson was taught in a ninth grade algebra class, although it may be more appropriate for precalculus.

Materials and Tools
The students need to use computers to complete this lesson. No special software is required beyond a program that can modify text documents; the optimization solver is accessed through the internet.
This activity requires three handouts, one for each lesson. Each lesson also requires a set of files that will be used as examples by the students.

Preparation
Each lesson requires a set of files to be available to the students.
Lesson 1: McDonalds-model.txt, McDonalds-data.txt, McDonalds-commands.txt
Lesson 2: McDonalds2-model.txt, McDonalds2-data.txt, McDonalds2-commands.txt
Lesson 3: McDonalds2-model-additionalConstraints.txt
The teacher should also modify the handouts (Lesson 1, Lesson 2, and Lesson 3 Worksheets) to reflect the email addresses to which students should send their results, as well as any changes in the NEOS website. An answer key is included for the homework assignment at the end of the Lesson 1 Worksheet.

Prerequisites
Students should know how to formulate small linear programs.
Students should have some previous exposure to summation notation, although they need not be very familiar with it, since the set-based summation notation used in this lesson differs significantly from the index-based notation more commonly used on the high school level.

Background
In order to succeed in this lesson, you should be able to formulate a small linear program. (That is, a program with four decision variables, an objective function, and two or three constraints.)

Teaching Notes
This lesson builds on previous lessons about formulation of linear programs. Students must be able to formulate linear programs algebraically in order to do any of the parts of this activity.
In addition, students need to have some basic computer skills in order to complete this lesson. In particular, they need to know how to download a text file from a web page and how to save modified versions of the file without overwriting the original file.
Lesson 1:
In the first lesson, the students write very simple linear programs algebraically. Then they transfer those programs into AMPL (a modeling language, used for representing mathematical programs as text files) and solve them with an online solver.
Begin the lesson by solving some simple mathematical programs (Lesson 1 Worksheet).
The next several pages of the lesson introduce the example that will be used in every lesson of this activity. It comes from the film *Supersize Me*, in which a man attempts to eat a diet consisting only of McDonald's food. The mathematical program formulated here seeks the minimum-cost McDonald's diet using four foods that includes at least 2000 calories. Ensure that the students understand the

algebraic formulation of the problem, including the meaning of every variable and expression. Then have them, in groups, discuss how the algebraic formulation on page 2 is related to the AMPL file on page 3. The goal is for them to see a one-to-one correspondence between the parts of the algebraic description and the file McDonalds-model.txt. If necessary, have them discuss it in groups multiple times, or point out analogous parts of the algebraic and AMPL representations of the model to help them make the connections.

After they have been convinced that the AMPL file is equivalent to the algebraic formulation, demonstrate the use of the online solver and the interpretation of the results. A screenshot of the online solver interface and an example of the solver output is included on pages 4-5, and directions for using the solver are on page 6. The only part of the solver output that will be used in this and succeeding lessons is the solution, the error output (if any), and the sums of the various nutrients, which means that the students should be taught (at least for this activity) to ignore the majority of the solver output.

NOTE: This part of the lesson should be modified in the week before it is presented. You should ensure that the solver named in the lesson materials is still working. If the solver mentioned in the lesson (Gurobi) is not available, any other mixed-integer linear programming solver will work (such as MINTO). Furthermore, you should remove any references to the school where this was originally taught (such as the email address in the screenshot and the methods for downloading the files) and replace them with appropriate language for your school.

Next, have the students download the example files and solve the problem with NEOS. They should get almost exactly the same output as that included in the lesson materials. *The amount of time necessary to complete this part of the lesson will vary greatly depending on the computer skills of the students. For some groups, this may be a lesson in and of itself.*

The final in-class work is to formulate two more linear programs and to solve them with the online solver (page 7). Students will inevitably make coding errors, as with any programming language; debugging advice for common problems is included on page 6.

For homework, students will formulate a linear program based on dietary guidelines for teenage boys and girls, and use the solver output to answer the question "Is it possible to have a healthful diet eating only food from McDonald's?" This problem, stated on page 8, also includes references to the solver being used, and should be edited to reflect the solver chosen for the examples.

Lesson 2:

The second lesson starts with a discussion of the output from the homework. The first page of the lesson structures the discussion around three questions. Depending on the level of the students, this discussion could be done in student groups, could be teacher-led, or could be a mixture of the two. Some essential points that should come up in the discussion are that the homework indicates that it is not possible to eat a healthful diet from McDonald's because the solver is not able to find a feasible solution. However, the information provided to the solver has some shortcomings: most significantly, it only included four food items (McDonald's has many more), and it only modeled a few of the nutrition requirements. In this lesson, students will modify their model in light of these considerations.

Again, the lesson starts with the *Supersize Me* example. In order to accommodate a larger number of foods, the linear program is presented with summation notation. This allows a large set of decision variables to be represented compactly. As before, ensure students understand the algebraic notation of the problem, then have them connect it to the AMPL formulation on the facing page. Also, take this opportunity to point out the comments in the AMPL code. In AMPL, any line that starts with # is a comment. They will be expected, from here on, to include comments in their models.

The last page of this lesson presents a two-part homework assignment. Part 1, which students can finish based on this lesson, consists of modifying their model from last week to include summation notation. They also need to include comments for every line of the model file. Students will be prepared for Part 2 after the next lesson. Students should download the new versions of the data, model, and commands

files for this homework. Also, they need to have a successful model for Part 1 of this homework in order to complete Part 2.
NOTE: Lessons 2 and 3 must be completed with the McDonalds2-data.txt and McDonalds2-commands.txt files.
Lesson 3:
The third lesson begins by demonstrating the absurdity of the solutions given by the basic model from *Supersize Me* on the larger set of foods. Since the objective function of the problem seeks to minimize cost, the optimal solution consists entirely of "free" foods. This happens because the optimization model is an imperfect reflection of reality. Although no one pays for coffee cream or ketchup at McDonald's, it is also not true that a person can eat an unlimited amount of these foods – and any student that has eaten at McDonald's can recognize this violation of reality. Therefore, in this lesson, students are shown how to look at a solution from the solver, recognize violations of "reality," and add constraints to improve the model. The lesson includes two new constraints, and the file McDonalds2-model-additionalConstraints.txt includes a few more (so that the optimal solution does not consist entirely of "free" foods).
The lesson then shows the solutions that come from the problems solved in Lesson 2. Although these diets are not impossible to follow, they have shortcomings. The students' homework is to improve their models by using the same procedure demonstrated in class (finding shortcomings in the solution returned by the solver). This is probably best done in pairs, although groups or individuals could also work.
Students should be encouraged to be creative with their additional constraints. The constraints should express their understanding of how people behave at McDonald's; that is, they should bring the model closer to an expression of how people really behave when choosing food at McDonald's. Some suggestions:

- Limit the number of times foods in a certain group (such as desserts or fries) can be eaten.
- Allow the person to eat just part of a food. (This is accomplished by changing the decision variables to continuous instead of integer variables.)
- Create constraints that link certain foods together. For example, the number of orders of fries could be related to the number of sandwiches ordered.

The students submit a written response with their final model and solver output, describing the constraints they added to the model in order make it better represent reality.

Assessment
Each lesson includes homework, and the final lesson includes a written response. There are also abundant opportunities to observe students working in class.

Additional Information
The "linear programs" solved by the students in this lesson are integral linear programs because all of the solution variables are integers. Therefore, it is *not* possible to solve them to optimality using the simplex method.
The linear program used here is similar to those in introductory mathematical programming courses at the undergraduate level, where it is called "the diet problem."
As an introduction to programming, students will benefit much more from this assignment if they are allowed to struggle with the debugging on their own. AMPL has a relatively simple syntax (compared to other languages), so students should be able to solve almost all problems on their own by referring to the debugging advice included in Lesson 1. The solver output can assist in debugging because most solvers print out the part of the model which they could not understand.
An exception to this: perhaps because students prefer to use USB drives to store files, sometimes their files are corrupted, and they fail to solve when uploaded to the NEOS solver. If a student appears to

have perfect syntax in their model but still does not get a solution from the solver, he or she should delete the original versions of the data, model, and commands files; download them again from the original source; and then copy and paste the apparently correct syntax into the new file.