Overview of Optimization Software

Heather Gaddy, Heather Hardeman, Jesse Patsolic, Rebecca Waldrip

April 11^{th}, 2013

1 Introduction

An optimization problem is a problem in which one attempts to maximize or minimize a function within a set of given constraints. The optimum solution can be found from a set of possible solutions by choosing input values in a systematic method, computing the output of the function to be optimized from each input value, and keeping a record of those input values that give the best solutions. Many real-world problems can be modeled as optimization problems. For example, an optimization problem could be used in the design of a motor by attempting to optimize the power output of the motor within a set of design constraints. A famous example of a real-world optimization application is the Stigler diet problem, in which Stigler, an economist, found the minimum cost of a diet within nutritional constraints [15].

Often optimization problems are solved using optimization software. There are several types of optimization software with various capabilities and benefits, including open source software, proprietary software, and freeware. This paper takes an in-depth look at one optimization program of each type [15].

2 Open Source: OpenSolver

There are several open source optimization programs. An open source program is a program that can be downloaded and used at no cost. The code for such programs is readily available. The benefit of open source software is that individuals are free to read, redistribute, and modify the code. The software evolves as people improve upon, adapt, and correct bugs in the program [12].

One important open source optimization program is OpenSolver, a solver designed to work with Microsoft Excel. It was developed and is maintained by Andrew Mason and students from the Engineering Science department at the University of Auckland in New Zealand. The program is a linear and integer optimizer available by free download [12]. Any published software that is made free should be released under a free software license, and OpenSolver is released under a GNU General Public License (GNU GPL) [13]. OpenSolver extends Excel's built-in solver and is compatible with regular Excel spreadsheets and applications [12].

OpenSolver relies upon the open source COIN-OR CBC optimization engine to solve problems. COIN-OR (COmputational INfrastructure for Operations Research) is an orginization with the goal of encouraging the development of open-source software to be used in operations research. COIN-OR assisted in the development and publishing of CBC (COIN-OR Branch and Cut), an open-source integer programming solver that is written in C++. CBC code was written by John J. Forrest, and it is maintained by Ted Ralphs. CBC can be used as a stand-alone program or as the library code that runs another program, as in the case with OpenSolver. OpenSolver analyzes the Excel spreadsheet to find the optimization model, which it then writes as a file and passes the file to the CBC optimization engine.

The CBC optimization engine then solves the problem, and the result is read back into OpenSolver and loaded into the spreadsheet [14].

To solve the optimization problem, OpenSolver iterates through the decision variables in the model, making small changes to each variable and keeping a record of how the objective function and constraint cells change with each of the small changes. OpenSolver only works with linear models. The program includes some checks to determine that the problem is in fact linear, but it remains the responsibility of the user to ensure that the model is truly linear (so it contains no square roots, absolute values, etc.) [12].

One useful feature of OpenSolver is a QuickSolve mode that can be used to solve an optimization problem after making only small changes to the model. Often building a model takes longer than actually finding the optimum solution, and the QuickSolve function allows the user to avoid having to rebuild a model every time a small change needs to be made. OpenSolver allows the user to define the cells that one will want to change as *parameter cells*. OpenSolver then analyzes the spreadsheet to build the model and determine how the constraints will change as changes are made to the parameter cells. The user can then change the parameters and select the QuickSolve button to quickly solve the slightly modified model. Another advantageous feature is that the solver has no set limit on the size of a problem [12].

There are a few potential improvements that could be made to OpenSolver that users have identified. For example, a way to check that the objective function value and the constraint values found by CBC match those calculated by the spreadsheet once the optimal solution is entered in would be useful. The fact that suggestions can be made and then individuals are free to work to try to improve the code in these ways is one of the benefits of open source software [12].

3 Proprietary Software: Maple

Proprietary software, also known as closed source software, is "licensed under exclusive legal right of the copyright holder with the intent that the licensee is given the right to use the software only under certain conditions, and restricted from other uses, such as modification, sharing, studying, redistribution, or reverse engineering" [10]. These types of software are not always free. If they are free, they are called freeware (this is discussed in the next section). "[W]hether or not proprietary software can be distributed, and what the fee would be, is at the proprietor's discretion" [10]. Furthermore,"vendors typically limit the number of computers on which software can be used, and prohibit the user from installing the software on extra computers" [10]. One example of proprietary software that is used by many students here at Wake Forest is Maple. Students learn how to use this software in Calculus and continue using it during other science (and possibly economic) courses. The first version of Maple appeared in 1980, and was first demontrated in 1982. The latest version of Maple, version 17, was released in March 2013. Maple is primarily

based in C, with some interfaces written in Java [9]. In fact Maple contains interfaces to many other computer languages other than C, such as Java, Matlab, C#, Visual Basic, and Excel [9]. It can also translate Mathematica code into Maple code. Its programming language "combines elements of functional, procedural, and object oriented programming" [11].

Maple has a number of wonderful features that make the program ideal for engineers, educators, researchers, and students alike. Maple is an especially great tool for students who may not have a great deal of programming background because the commands are not very complicated. It can be used to solve many types of problems, including those related to calculus, differential equations, factorization, number theory, optimization, and power series problems [7]. It can also be used to solve problems in other fields, including physics, financial modeling, control design, and research in computer algebra [5]. Further, it is often used as a teaching aid in calculus classes. Some interesting features that Maple offers and that helps to make it user friendly are its easy to use interface, side panel to assist in writing functions, commands which are easy to program, excellent graphing features, easily accessible help menu, handwritten symbol recognition, math equation editor, and ability to produce slideshows.

One important use of Maple is to solve optimization problems. Maple has two main packages for solving these types of problems. It has an optimization package for solving local optimization problems (both linear and nonlinear), and it also has a global optimization toolbox [8]. One of the features that is different from MATLAB is that there are a number of ways to enter an optimization problem in the command window. If the user needs Maple to solve an optimization problem quickly, it can be entered as a matrix problem, as is done in MATLAB. This is especially ideal for solving large-scale problems. However, to solve a simpler optimization problem, such as one with two or three variables, the problem can also be expressed in algebraic form [8], [3]. Problems can also be entered in procedure form, such as block-box procedure, which means that Maple can retrieve a problem from an external file.

Maple can solve optimization problems using a variety of methods. If the problem is simple, using the maximize and minimize commands requires the least amount of information [3], but since life is rarely that simple, Maple is also able to carry out large scale optimization problems using gradient search methods such as steepest descent, Newton's method, Quasi-Newton Method, and so on [4]. It can also model optimization problems easily using other symbolic tools, such as the visualization tool to plot results [8]. Maple is able to solve many types of optimization problems entered in a variety of ways.

4 Freeware

Freeware is a type of software that, just as the name suggests, is free for the user. It is often a more restrictive version of a program that is available for purchase. Many companies often employ freeware as a means to introduce individuals to their software, hoping to

incite them to invest in the less-restrictive, more powerful product.

A similar tactic is used with the algebraic modeling language AMPL, which stands for A Mathematical Programming Language. Its creators, Robert Fourer, David Gay, and Brian Kernighan of Bell Laboratories, offer a version of the software to students for free [2]. While developed in 1985, the creators have updated the program several times to include such areas as nonlinear programming and automatic differentiation as well as constraint programming. They designed AMPL to employ declarative and imperative programming styles in order to describe and solve optimization problems, mainly large-scale, as well as other problems. Some of these other types of problems include those of interest in non-linear optimizaiton such as linear and nonlinear programming, quadratic programming, global optimization, and constraint programming.

With regards to working in AMPL, the user employs the declarative language elements of AMPL (including sets, scalar and multidimensional parameters, decision variables, and objectives and constraints) to compose optimization models [2]. In comparison to other modeling languages such as GAMS, MGG, LINGO, MPL, and AIMMS, AMPL takes advantage of several features provided by these languages. While still being easy to use, the software augments the features of these other languages as well. When compared to the spreadsheet packages of Excel, 1-2-3, and Quattro, AMPL creates optimization models much more quickly. Also, the freeware can access more solvers than any of the spreadsheet packages. In addition, the versatility of the language provides a more organic approach to creating and maintaining more intricate optimization models.

As mentioned previously, the freeware version of a program is usually more restrictive. The Student Edition limits the problem size to 300 variables and 300 constraints and objectives [1]. Beyond the limitation to problem size, all the features of the full version of AMPL remain available for use in the Student Edition. In the professional version of AMPL, the only restriction to the number of variables stems from how much memory the computer being used possesses. With this in mind and on a final note for freeware, it is interesting to note that the NEOS server has several solvers which accept AMPL input [2]. In fact, the NEOS announced in 2011 that AMPL was the most used language for characterizing mathematical programming.

5 Examples

The Rosenbrock function was minimized using both Maple and AMPL to compare the two programs. OpenSolver cannot handle non-linear programming and was not considered as an example for minimizing the Rosenbrock function.

To minimize the Rosenbrock Function in Maple, one must call the Optimization package. Once initiated, the `Minimize` function may be used with the form
`Minimize(function,initialpoint=(`$\vec{x}$`))` to minimize the function in question from the

initial guess. This was done with initial points $(100, 100), (10, 10), (0, 0)$ and $(1.2, 1.2)$. Maple gives back the points $(0.3922, 0.13030)$, $(1.3714, 1.8633)$, $(1.0000, 1.0000)$ and $(1.0000, 1.0000)$ with the last two function values of 8.02×10^{-13} and 8.23×10^{-13} respectively. From previous experience with the Rosenbrock function, it is known that $(1, 1)$ is the true minimizer where the function value is 0.

Using AMPL, one must create a model file and instantiate the variables and functions to be used, then tell AMPL what exactly to minimize. Given the initial point $\vec{x} = (100, 100)$, AMPL found the minimizer $\vec{x} = (1, 1)$ on the first try [16].

6 Conclusion

Each optimization program has benefits and features that make it suitable for certain types of problems. The type of software and the particular optimization program chosen to solve a model depends upon the characteristics of the model being solved and the resources available to the user. Optimization software is a valuable tool used for many real-world applications on a regular basis.

References

[1] http://www.ampl.com/DOWNLOADS/details.html

[2] http://en.wikipedia.org/wiki/AMPL
RKW http://www.ampl.com/FAQ/index.html#WhatisAMPL

[3] http://www.adeptscience.co.uk/products/mathsim/ maple/Demos/optimization/ Optimization.html

[4] http://archives.math.utk.edu/ICTCM/VOL16/S018/paper.pdf

[5] http://www.maplesoft.com/products/Maple/features/

[6] http://www.mapleprimes.com/posts/43330-What-Do-People-Use-Maple-For-

[7] http://www.maplesoft.com/support/help/ Maple/view.aspx?path=Optimization

[8] http://www.maplesoft.com/products /maple/demo/player/Optimization.aspx

[9] http://en.wikipedia.org/wiki/Maple_(software)

[10] http://en.wikipedia.org/wiki/Proprietary_software

[11] www.wikivs.com/wiki/Maple_vs_Mathematica

[12] http://opensolver.org/

[13] http://www.gnu.org/licenses/

[14] https://projects.coin-or.org/Cbc

[15] http://en.wikipedia.org/wiki/List_of_optimization_software

[16] www.ampl.com/EXAMPLES