Specialized Knowledge Area Module 5

Deterministic Operations Research Techniques

Student: Terri Bittner

Faculty Mentor: Dr. Kimberly L. Ross

Faculty Assessor: Dr. Kimberly L.

Walden University

January 25, 2006

BREADTH ABSTRACT

Deterministic methods of operations research are the heart of the field, and the area of research is absolutely huge. This paper provides an introduction to linear programming, network algorithms, and nonlinear programming. Examples are given throughout, and are generally shown at the end of the theoretical discussion of each subtopic. The section on linear programming covers the geometric representation of a linear programming problem, the simplex method and its variants. A discussion comparing interior point methods to the simplex method is also included. The section on network algorithms includes introductions to the transportation algorithm, the transshipment problem, maximum flow and minimum cost networks, and the assignment problem. Various algorithms are discussed. The section on nonlinear programming includes a discussion of constrained and unconstrained problems, convexity, and the Karush-Kuhn-Tucker conditions.

DEPTH ABSTRACT

Networks represent a large number of application problems in many fields. This paper provides a basic introduction to some of the most important network problems and algorithms. These include Dijkstra's shortest path algorithm, the shortest path problem as a transshipment problem, the Ford-Fulkerson Method for solving maximum flow problems, the critical path method, minimum cost network flow problems, minimum spanning tree problems, and the network simplex method. Examples are shown throughout the paper after the theoretical discussion of each topic.

APPLICATION ABSTRACT

A seminar was given to approximately 60 precalculus and calculus students at Sequoia High School on January 24, 2006. The purpose of the seminar was to introduce and promote the field of operations research to students who are more likely than most to choose a career in a technical field. An appendix to the paper contains an outline for the seminar, and includes examples of linear programming problems that were presented to the students. A computer demonstration was also given to students as part of the presentation. This paper discusses the seminar's purpose, content, and the student response to the presentation.

Learning Agreement

Specialized Knowledge Area Module 5:

Deterministic Operations Research Techniques

Student: Terri Bittner

Faculty Mentor: Dr. Kimberly L. Ross

Faculty Assessor: Dr. Kimberly L.

Walden University

November 20, 2005

Overview

This learning agreement applies to my doctoral study in AMDS/Operations Research. I will use KAM V to gain an understanding of some of the deterministic techniques commonly used in operations research.

I will critically examine deterministic methods throughout this study and demonstrate an understanding of both the techniques and the underlying assumptions associated with each topic. There are three specific deterministic methods to be studied in this KAM. The first is linear programming. This is important largely because its applications include many real problems encountered in business and other applications. The second deterministic topic to be studied is network algorithms. Though the problems solved using network algorithms can generally be solved in the same way as other linear programming problems, specialized algorithms for each type of problem are much more efficient. The third deterministic method to be studied is nonlinear programming. Nonlinear programming provides ways to solve maximization and minimization problems with and without constraints when there are nonlinear components.

During the depth phase of the KAM, a more in-depth study and analysis of network algorithms will be done. In addition, a study will be made of more recent trends and theories in network algorithms. Several applications in the recent literature will be examined and analyzed in the annotations.

For the application portion of the KAM, a seminar will be given for advanced calculus students at a high school in the San Francisco Bay Area that offers the International Baccalaureate program. The seminar will be designed to acquaint students with the mathematics and applications of deterministic techniques in operations research,

and to promote the field of operations research. This seminar will include a demonstration of computer-based solutions to deterministic problems.

Breadth Objectives

Solve a range of textbook problems to demonstrate an understanding of the techniques and underlying assumptions associated with linear programming, network algorithms, and nonlinear programming, and integrate these problems into an essay that discusses all three topics. Examine a range of problems in each of these areas of study and analyze the underlying assumptions and appropriate methods to solve each of these problems, including the Kuhn-Tucker conditions.

Preliminary Breadth References

Bazaraa, M., Jarvis, J., & Sherali, H. (1990). *Linear Programming and Network Flows.* New York: John Wiley & Sons.

Bazaraa, M., Jarvis, J., & Sherali, H. (XXX). *Nonlinear Programming: Theory and Algorithms.* New York: John Wiley & Sons.

Kallrath, J. (2005). *Online Storage Systems and Transportation Problems with Applications: Optimization Models and Mathematical Solutions.* New York: Springer.

Pardalos, P., & Pitsoulis, L. (Eds.) (2000). *Nonlinear Assignment Problems: Algorithms and Applications*. Boston: Kluwer Academic Publishers.

Rardin, R. (1998). *Optimization in Operations Research.* Upper Saddle River, NJ: Prentice Hall.

Winston, W. L. (2004). *Operations Research: Applications and Algorithms*. Belmont, CA: Brooks Cole Publishing Company.

Breadth Demonstration

In a scholarly paper of approximately 30 pages, a set of problems with detailed solutions will be provided along with an introduction and analysis of each topic being studied in the area of deterministic methods. The problems will be provided as examples in the paper. These include linear programming, network algorithms, and nonlinear programming.

Depth Objectives

Examine network algorithms in depth, including recent findings and schools of thought methods to solve these complex problems. If appropriate, each application and/or method found in the recent articles will be analyzed for strengths, weaknesses and applicability of the solution to the specific problem. The focus will be on recent research conducted in the last 5 to 15 years, and the analysis will be built upon the foundation laid in the Breadth section of the KAM. The *Transportation Science Journal* will be used for some sources. Depth references will be altered as necessary and appropriate. An annotated bibliography of at least 15 recent articles will also be included in the Depth section of the KAM.

Preliminary Depth References

Aissi, H., Bazgan, C., & Vanderpooten, D. (2005). Complexity of the min-max regret assignment problems. *Operations Research Letters, 33(6),* 634–640.

Bjørndal, E., Hamers, H., & Koster, M. (2004). Cost allocation in a bank ATM network. *Mathematical Methods of Operations Research, 59(3),* 405–418.

Burdett, R., & Kozan, E. (2004). The assignment of individual renewable resources in scheduling. *Asia-Pacific Journal of Operational Research, 21(3),* 355–377.

Calleja, P., Borm, P., & Hendricks, R. (2005). Multi-issue allocation situations. *European Journal of Operational Research, 164(3),* 730–747.

Chiou, S. (2005). Bilevel programming for the continuous transport network design problem. *Transportation Research: Part B, 39(4),* 361–383.

Ding, H., Lim, A., Rodrigues, B., & Zhu, Y. (2005). The over-constrained airport gate assignment problem. *Computers & Operations Research, 32(7),* 1867–1880.

Greff, J., Idoumghar, L., & Schott, R. (2004). Application of Markov decision processes to the frequency assignment problem. *Applied Artificial Intelligence, 18(8),* 761–773.

Higgins, A. J. (2005). A percentile search heuristic for generalized assignment problems with a very large number of jobs. *Asia-Pacific Journal of Operational Research, 22(2),* 171–188.

Nauss, R. M. (2004). The elastic generalized assignment problem. *Journal of the Operational Research Society, 55(12),* 1333–1341.

Rhee, S. (2005). Sharing-group allocation problems. *Economics Letters, 86(1),* 51–56.

Suh-Wen, C. (2005) Joint optimization for area control and network flow. *Computers & Operations Research, 32(11),* 2821–2841.

Taylor, G. (2005). A simulation-based software system for barge dispatching and boat assignment in inland waterways. *Simulation Modeling Practice & Theory, 13(7),* 550–565.

Toktas, B., Yen, J., & Zabinsky, Z. (2006). Addressing capacity uncertainty in resource-constrained assignment problems. *Computers & Operations Research, 33(3),* 724–745.

Yildirim, M. B., & Hearn, D. W. (2005). A first best toll pricing framework for variable demand traffic assignment problems. *Transportation Research, 39(8),* 659–678.

Yiu-Wing, L. & Hou, R. (2005). Assignment of Movies to Heterogeneous Video Servers. *IEEE Transactions on Systems, 35(5),* 665–681.

Zülch, G., Rottinger, S., & Vollstedt, T. (2004). A simulation approach for planning and re-assigning of personnel in manufacturing. *International Journal of Production Economics, 90(2),* 265–277.

Depth Demonstration

In a scholarly paper of about 30 pages, I will integrate the theories from the breadth component of the KAM with a more in-depth study of networks and algorithms to solve network problems. The depth demonstration will also include an annotated bibliography of at least 15 articles of which at least 75% will be no more than 5 years old.

This bibliography will satisfy the requirement that the depth component include material concerning current research in the area of networks.

Application Objectives

When the term "operations research" comes up in conversation, it is often met with many eyes glazing over, and the sound of footsteps walking briskly away. This is true even of advanced math students taking calculus. However, I believe that the reason for this response is a lack of understanding or a misunderstanding of the terms, and that advanced math students would actually find the material fascinating if they got a chance to really see and experience it. If operations research is to be promoted as a legitimate and reachable career choice, then education about it needs to start with high school students. My objective is to use a seminar format to educate high school students about the basic mathematics and applications of deterministic methods of solving operations research problems, to use this forum as a way to promote operations research as a possible career choice for students, and to answer students' questions about careers in mathematics. A demonstration of computer-based solutions to optimization problems will be included in the seminar.

Preliminary Application References Materials

No specific reference materials except for the breadth texts

Application Demonstration

Prepare lecture notes or outline and conduct a seminar for calculus students in the International Baccalaureate Program at Sequoia High School in Redwood City, CA. The seminar will include examples of problems that can be solved using deterministic

techniques including linear programming problems, network algorithms, and nonlinear programming problems. The mathematics required to solve these problems will be discussed, as will strategies to approach more complex problems. The target audience already has familiarity with calculus, linear algebra, and computer programming. A demonstration of computer-based solutions to optimization problems will also be given during the seminar.

In a scholarly paper of about 10 pages, the seminar will be described and discussed. The paper will include analysis of student responses to the talk as well as a description of the material in the seminar. The lecture outline will be included as an appendix.

Walden University

Self-Evaluation: Knowledge Area Modules (KAMs)

Students: Attach this completed form to each KAM prior to forwarding to the Faculty Assessor
(Please use reverse side as necessary)

Student Name__Teresa (Terri) Bittner_____**Date**_____1/24/06_____

KAM number ___5__ **Title** _____Deterministic Operations Research Techniques_____

1. What knowledge/experience did you bring to this KAM? How did you capitalize/expand on this base?

I had a good background in basic linear programming theory and techniques. I also have a deep knowledge of calculus, and this helped greatly with the nonlinear programming part of the KAM. My previous completion of KAMS 6 and 7 also helped with this difficult KAM.

2. Describe the quality of the **Breadth** section in the light of the intellectual and communication skills demonstrated in this KAM.

I wrote a 45 page essay about linear programming, network algorithms, and nonlinear programming. Examples were interspersed throughout the paper, and followed the theory of each topic. One of the highlights of the breadth section is the discussion of interior point algorithms vs. the simplex method.

3. In the **Depth** section, what key ideas/concepts most engaged your thinking and imagination relative to your area of study?

I knew next to nothing about networks before beginning this KAM. I was thoroughly engaged by the varied and numerous problems that are applications of networks. In some cases it is amazing that one algorithm can be used for such a varied set of applications. At other times it is amazing that so many different algorithms and techniques are used in one topic.

4. Expound on the most meaningful theoretical construct studied and applied to your professional setting in the **Application** section. What can you do differently/better as a result of this KAM?

Teaching high school students is a challenging endeavor, especially when the topic is something outside of their knowledge and immediate area of interest. Conducting a seminar for the purpose of introducing operations research and encouraging students to consider a career in the area was a rich experience. It was wonderful to answer students' questions, and the entire experience enriched my overall interest and expertise in adolescent mathematics instruction.

5. Briefly describe the most important **Social Issue** covered in this KAM.

The social issue covered in this KAM involved including high school students in what is normally a topic reserved for graduate students in mathematics and statistics. If any field of study is to thrive, young people must be encouraged to enter the field. None of the students that I spoke to had even heard of operations research. It was a great experience to define many new words for a group of very intelligent students, and to find that a few found the experience "inspirational."

BREADTH DEMONSTRATION

BREADTH TABLE OF CONTENTS

BREADTH DEMONSTRATION

Introduction

Optimization problems are considered the cornerstone of operations research. They come in many varieties, sizes, and from many applications, and the accurate solution and implementation of these problems is key to the financial success of many businesses. For example, utility companies must use optimization problems to make sure that peak demands are met while costs are minimized. The variations are so numerous that there are many journals devoted to these complex problems. Optimization problems are also continuing to emerge in many fields of science.

In the last 20 years, numerous computer packages have emerged for the express purpose of solving these complex, and often large, problems. LINGO™, LINDO™, ProcessModel™, LOQO™, LIPSOL™, OSL™, HOPDM™, and CPLEX™ are just a few of the programs dedicated to solving optimization problems. Many other programs, such as Excel™ and Mathematica® have specialized modules that solve optimization problems, and each piece of optimization software may use slightly or altogether different algorithms and methods to solve problems.

This paper deals with two main types of problems, linear programming problems and nonlinear programming problems. Several methods for solving these problems are also discussed. These include the Simplex Method and its variants, interior point methods, and several specific network algorithms. The subsection on linear programming includes a brief discussion and several examples of linear programming problems solved by the Simplex Method and its variants. The end of the linear programming subsection includes a discussion and comparison of the Simplex Method with interior point methods

for solving linear programming problems. Interior point methods emerged in the late 1980's (Singh & Singh, 2002), and have been a very lively area for investigation and debate since that time.

Network algorithms are methods to solve a special class of linear programming problems called Network Problems. These problems are solved using special algorithms that exploit the structure of the constraint matrix. Related problems include the Transportation Problem, the Transshipment Problem, which is a more generalized version of the Transportation Problem, Maximum Flow and Minimum Cost Networks, and Job Assignment Problems. This section of the paper includes a basic introduction to this special class of problems. A deeper discussion of these problems will be included in the Depth section of the paper.

Finally, the subsection on nonlinear programming includes a very basic introduction to unconstrained and constrained nonlinear programming problems, including steepest descent (gradient path) and classical search methods. The topics of convexity, optimality conditions (also known as the Karush-Kuhn-Tucker conditions) are also discussed. Since nonlinear programming is a very complex topic, only the basics are covered in this paper.

Note that due to the wide range of material covered, as well as the complexity of the topic, it is assumed that the reader already has familiarity with the basic definitions, theorems, and algorithms associated with the material, especially with the material on linear programming. In many cases, space considerations required that examples of algorithms or methods be shown with little introduction or explanation about the theory involved.

Linear Programming

Linear programming problems are maximization or minimization problems that are subject to constraints, where the constraints and the function to be maximized or minimized are all linear combinations of the variables involved. The first well-known method to solve these problems is called the Simplex Method, and it has been around for decades (Bazaraa, Jarvis, & Sherali, 2005). More recent methods are various kinds of interior point methods. These more recent methods will be discussed later in this section. The basics of linear programming are introduced in this section. This begins with the basic graphical meaning of a linear programming problem, followed by an introduction to the Simplex Method and a short discussion of the revised Simplex Method. Examples are shown throughout. Finally, interior point algorithms will be introduced and compared with the Simplex Method.

The basic linear programming problem is of the form

$$\text{Minimize} \quad z = cx$$

$$\text{Subject to:} \quad \begin{aligned} Ax &\geq b \\ x &\geq 0 \end{aligned}$$

Maximization problems can be formulated similarly as

$$\text{Maximize} \quad z = cx$$

$$\text{Subject to:} \quad \begin{aligned} Ax &\leq b \\ x &\geq 0 \end{aligned}$$

where c is a $1 \times n$ vector, x is an $n \times 1$ vector, b is an $m \times 1$ vector, and A is an $m \times n$ matrix.

Example 1. (Winston, 2004, p. 68, #9, modified)

Graphically determine two optimal solutions to the following LP:

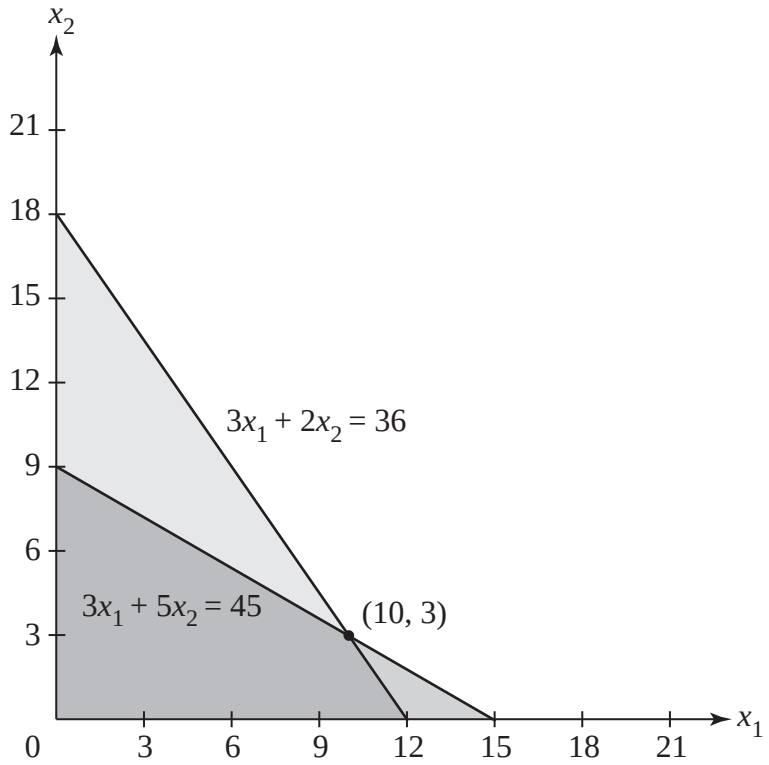
$$\max z = 3x_1 + 5x_2$$

subject to:

$$3x_1 + 2x_2 \le 36$$
$$3x_1 + 5x_2 \le 45$$
$$x_1, x_2 \ge 0$$

Solution:

Graphically determine the optimal solutions to the linear programming problem by graphing the feasible region and identifying the extreme points. Note that it is well known that the solution to a linear programming problem must be one of the extreme points of the feasible region of the problem (Bazaraa, et al., 2005; Winston, 2004).



From the graph, it is obvious that there are extreme points at (0, 0), (12, 0), and (0, 9). To find the fourth extreme point, solve the system of equations

$$\begin{cases} 3x_1 + 2x_2 = 36 \\ 3x_1 + 5x_2 = 45 \end{cases}$$

to determine the point where the lines intersect. The solution to this system is the point (10, 3). Finally, test each of the four extreme points to determine the maximum(s).

Test (0, 0): $3x_1 + 5x_2 = 0$

Test (12, 0): $36 + 0 = 36$

Test (0, 9): $0 + 45 = 45$

Test (10, 3): $30 + 15 = 45$

Two optimal solutions exist and are at (0, 9) and (10, 3). Each has a maximum value of 45.

Sometimes the most difficult part of a linear programming problem is converting a word problem into a standard form linear programming problem. An example of a word problem that can be formulated as a linear programming problem is shown below.

Example 2. (Winston, 2004, p. 120, #45).

Currently we own 100 shares each of stocks 1 through 10. The original price we paid for these stocks, today's price, and the expected price in one year for each stock is shown in Table 1 below. We need money today and are going to sell some of the stocks. The tax rate on capital gains is 30%. If we sell 50 shares of stock 1, then we must pay tax of $(0.3) \cdot 50(30 - 20) = \$150$. We must also pay transaction costs of 1% on each transaction. Thus, our sale of 50 shares of stock 1 would incur transaction costs of $.01 \cdot 50 \cdot 30 = \$15$. After taxes and transaction costs, we must be left with $30,000 from our stock sales. Our

goal is to maximize the expected (before-tax) value in one year of our remaining stock. What stocks should we sell? Assume it is all right to sell a fractional share of stock.

Table 1

Purchase, Current, and Expected Future Prices of Stocks Owned

		Price ($)		
Stock	Shares Owned	Purchase	Current	In One Year
1	100	20	30	36
2	100	25	34	39
3	100	30	43	42
4	100	35	47	45
5	100	40	49	51
6	100	45	53	55
7	100	50	60	63
8	100	55	62	64
9	100	60	64	66
10	100	65	66	70
Tax rate (%)	0.3			
Transaction cost (%)	0.01			

Solution:

Assume we sell x_1 shares of stock 1, x_2 shares of stock 2, and so on. Then we will have $100 - x_1$ shares of stock 1 left after the sale, $100 - x_2$ shares of stock 2 left after

the sale, and so on. The data necessary to formulate the problem is given in Table 2 below.

Table 2
Taxes and Transaction Costs for Stocks from Table 1

Stock	Taxes	Transaction Costs
1	$.3(30-20)x_1$	$.01x_1(30)$
2	$.3(34-25)x_2$	$.01x_2(34)$
3	$.3(43-30)x_3$	$.01x_3(43)$
4	$.3(47-35)x_4$	$.01x_4(47)$
5	$.3(49-40)x_5$	$.01x_5(49)$
6	$.3(53-45)x_6$	$.01x_6(53)$
7	$.3(60-50)x_7$	$.01x_7(60)$
8	$.3(62-55)x_8$	$.01x_8(62)$
9	$.3(64-60)x_9$	$.01x_9(64)$
10	$.3(66-65)x_{10}$	$.01x_{10}(66)$

The linear programming problem is then written as follows:

Maximize:

$$36(100-x_1)+39(100-x_2)+42(100-x_3)+45(100-x_4)+51(100-x_5)$$
$$+55(100-x_6)+63(100-x_7)+64(100-x_8)+66(100-x_9)+70(100-x_{10})$$

Subject to:

$$(30x_1-3.3x_1)+(34x_2-3.04x_2)+(43x_3-4.33x_3)+(47x_4-4.07x_4)+(49x_5-3.19x_5)$$
$$+(53x_6-2.93x_6)+(60x_7-3.6x_7)+(62x_8-2.72x_8)+(64x_9-1.84x_9)$$
$$+(66x_{10}-0.96x_{10})=\$30{,}000$$

Solving this problem using the "Maximize" function in *Mathematica*®, the maximum value of the remaining stock next year is $20,893.70 if 100 shares of stocks 3, 4, 8, 9, and 10 are sold now, along with 63.75 shares of stock 6. The rest should be retained.

Most linear programming problems cannot be solved by graphing because too many variables are involved. To deal with this problem, several methods have been devised. The oldest and most popular of these is the Simplex method. An example is shown below.

Example 3. (Bazaraa, et al., 2005, p. 140, #3.25)

$$\max\ x_1 - 2x_2 + x_3$$

subject to:

$$\begin{aligned} x_1 + 2x_2 + 3x_3 &\le 12 \\ 2x_1 + x_2 - x_3 &\le 6 \\ -x_1 + 3x_2 &\le 9 \\ x_1, x_2, x_3 &\ge 0 \end{aligned}$$

Solution.

Adding slack variables x_5 and x_6, the initial tableau is as follows:

Tableau 1:

	z	x_1	x_2	x_3	x_4	x_5	x_6	RHS
z	1	1	–2	1	0	0	0	0
x_4	0	1	2	3	1	0	0	12
x_5	0	**2**	1	–1	0	1	0	6
x_6	0	–1	3	0	0	0	1	9

Tableau 2:

	z	x_1	x_2	x_3	x_4	x_5	x_6	RHS
z	1	0	$-\frac{5}{2}$	$\frac{3}{2}$	0	$-\frac{1}{2}$	0	–3
x_4	0	0	$\frac{3}{2}$	$\mathbf{\frac{7}{2}}$	1	$\frac{1}{2}$	0	9
x_1	0	1	$\frac{1}{2}$	$-\frac{1}{2}$	0	$\frac{1}{2}$	0	3
x_6	0	0	$\frac{7}{2}$	$-\frac{1}{2}$	0	$\frac{1}{2}$	1	12

Tableau 3:

	z	x_1	x_2	x_3	x_4	x_5	x_6	RHS
z	1	0	$-\frac{22}{7}$	0	$-\frac{3}{7}$	$-\frac{5}{7}$	0	$-\frac{48}{7}$
x_3	0	0	$\frac{3}{7}$	1	$\frac{2}{7}$	$\frac{1}{7}$	0	$\frac{18}{7}$
x_1	0	1	$\frac{5}{7}$	0	$\frac{1}{7}$	$\frac{4}{7}$	0	$\frac{30}{7}$
x_6	0	0	$\frac{26}{7}$	0	$\frac{1}{7}$	$\frac{4}{7}$	1	$\frac{93}{7}$

The maximum value is $\frac{48}{7}$ when $x_1 = \frac{30}{7}, x_2 = 0, x_3 = \frac{18}{7}$.

In addition to solving an existing linear programming problem, the Simplex tableaux can be used to ascertain information about the constraints and the extreme points. The tableaux can even be used to figure out what the original problem was. An example of this is shown below.

Example 4. (Bazaraa, et al., 2005, p. 145, #3.48)

The starting and current tableaux of a given problem are shown below. Find the values of the unknowns *a* through *n*.

Starting tableau:

	z	x_1	x_2	x_3	x_4	x_5	RHS
z	1	*a*	1	–3	0	0	0
	0	*b*	*c*	*d*	1	0	6
	0	–1	2	*e*	0	1	1

Current tableau:

	z	x_1	x_2	x_3	x_4	x_5	RHS
z	1	0	$-\frac{1}{3}$	*j*	*k*	*l*	*n*
	0	*g*	$\frac{2}{3}$	$\frac{2}{3}$	$\frac{1}{3}$	0	*f*
	0	*h*	*i*	$-\frac{1}{3}$	$\frac{1}{3}$	1	*m*

Solution:

We know from the current tableau that $g = 1$ and $h = 0$, since x_1 must be in the basis during the current iteration. We also know that $l = 0$ since x_5 is in the basis in the current iteration. The new basic variables are x_1 and x_5, so x_4 left the basis and x_1 entered. This means that $b = 3$ since the new x_1 row had to be divided by 3. This also gives $f = \frac{6}{3} = 2$, $c = 2$, and $d = 2$. Then it was necessary to add the new row 1 to the old row 2 to make the new row 2. Therefore, $h = 0$, and

$\frac{2}{3}+e=-\frac{1}{3}$,
$e=-1$

$\frac{2}{3}+2=i$
$\frac{2}{3}+\frac{6}{3}=\frac{8}{3}=i$, and

$m = 2 + 1 = 3$. We also needed to add $\frac{2}{3}x+1=-\frac{1}{3}$, where x is the multiplier of the new first row (to make the new top row). By solving the equations we get $x=-2$, $a = 2$, $j=-\frac{13}{3}$, $-\frac{2}{3}=k$, $l = 0$, and $n = -4$.

Unfortunately, the standard Simplex Method can only solve linear programming problems for which an obvious initial feasible solution exists. In most real problems this is not the case. To account for this and many other issues with real problems, the Simplex Method has been revised again and again over the years. One of the first revisions was called the Two-Phase method. Phase 1 of this method calls for the creation of artificial variables in addition to the slack variables. The sum of the artificial variables are then minimized in order to find an initial feasible solution to the problem. Once an initial feasible solution is found, the artificial variables are discarded, and the regular Simplex Method is used to solve the original problem in Phase 2. An example of the Two Phase Method is shown below.

Example 5. (Bazaraa, et al., 2005, p. 184, #4.2)

$$\min x_1 + 3x_2 - x_3$$

subject to:

$$\begin{aligned} 2x_1 + x_2 + 3x_3 &\geq 3 \\ -x_1 + x_2 &\geq 1 \\ -x_1 - 5x_2 + x_3 &\leq 4 \\ x_1, x_2, x_3 &\geq 0 \end{aligned}$$

Solution:

Add slack variables. The constraints become

$$\begin{aligned} 2x_1 + x_2 + 3x_3 - x_4 &= 3 \\ -x_1 + x_2 - x_5 &= 1 \\ -x_1 - 5x_2 + x_3 + x_6 &= 4 \\ x_1, x_2, x_3, x_4, x_5, x_6 &\geq 0 \end{aligned}$$

Since there is no obvious identity matrix, add artificial variables to first and second constraints. Now the constraints become

$$\begin{aligned} 2x_1 + x_2 + 3x_3 - x_4 + x_7 &= 3 \\ -x_1 + x_2 - x_5 + x_8 &= 1 \\ -x_1 - 5x_2 + x_3 + x_6 &= 4 \\ x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8 &\geq 0 \end{aligned}$$

The next step is to set up the Phase 1 tableau in order to minimize $x_7 + x_8$.

Initial Tableau (Phase 1)

	z	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	RHS
z	1	0	0	0	0	0	0	–1	–1	0
x_7	0	2	1	3	–1	0	0	1	0	3
x_8	0	–1	1	0	0	–1	0	0	1	1
x_6	0	–1	–5	1	0	0	1	0	0	4

To obtain $z_6 - c_6 = z_7 - c_7 = z_8 - c_8 = 0$, add rows 1 and 2 to row 0 and replace row 0.

Phase 1: Iteration 0

	z	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	RHS
z	1	1	2	**3**	–1	–1	0	0	0	4
x_7	0	2	1	3	–1	0	0	1	0	3
x_8	0	–1	1	0	0	–1	0	0	1	1
x_6	0	–1	–5	1	0	0	1	0	0	4

Phase 1: Iteration 1— x_7 will leave the basis and x_3 will enter

	z	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	RHS
z	1	–1	1	0	0	–1	0	–1	0	1
x_3	0	$\frac{2}{3}$	$\frac{1}{3}$	1	$-\frac{1}{3}$	0	0	$\frac{1}{3}$	0	1
x_8	0	–1	**1**	0	0	–1	0	0	1	1
x_6	0	$-\frac{5}{3}$	$-\frac{16}{3}$	0	$\frac{1}{3}$	0	1	$-\frac{1}{3}$	0	3

Phase 1: Iteration 2— x_8 will leave the basis and x_2 will enter

	z	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	RHS
z	1	0	0	0	0	0	0	–1	–1	0
x_3	0	1	0	1	$-\frac{1}{3}$	$\frac{1}{3}$	0	$\frac{1}{3}$	$-\frac{1}{3}$	$\frac{2}{3}$
x_2	0	–1	1	0	0	–1	0	0	1	1
x_6	0	–7	0	0	$\frac{1}{3}$	$-\frac{16}{3}$	1	$-\frac{1}{3}$	$\frac{16}{3}$	$\frac{32}{3}$

Now Phase 1 is over and the artificial variables can be eliminated. Start with the feasible solution $\left(0, 1, \frac{2}{3}\right)$. Form first tableau for Phase 2.

Phase 2: Iteration 0

	z	x_1	x_2	x_3	x_4	x_5	x_6	RHS
z	1	–1	–3	1	0	0	0	$-\frac{7}{3}$
x_3	0	1	0	1	$-\frac{1}{3}$	$\frac{1}{3}$	0	$\frac{2}{3}$
x_2	0	–1	1	0	0	–1	0	1
x_6	0	–7	0	0	$\frac{1}{3}$	$-\frac{16}{3}$	1	$\frac{32}{3}$

Multiply row 2 by 3 and row 1 by –1 and add to row 0 to get $z_2 - c_2 = z_3 - c_3 = 0$.

Phase 2: Iteration 1

	z	x_1	x_2	x_3	x_4	x_5	x_6	RHS
z	1	–5	0	0	$\frac{1}{3}$	$-\frac{10}{3}$	0	0
x_3	0	1	0	1	$-\frac{1}{3}$	$\frac{1}{3}$	0	$\frac{2}{3}$
x_2	0	–1	1	0	0	–1	0	1
x_6	0	–7	0	0	$\mathbf{\frac{1}{3}}$	$-\frac{16}{3}$	1	$\frac{32}{3}$

Phase 2: Iteration 2— x_4 enters and x_6 leaves basis

	z	x_1	x_2	x_3	x_4	x_5	x_6	RHS
z	1	2	0	0	0	2	–1	$-\frac{32}{3}$
x_3	0	–6	0	1	0	–5	1	$\frac{34}{3}$
x_2	0	–1	1	0	0	–1	0	1
x_4	0	–21	0	0	1	–16	3	32

It is now clear from the tableau that the problem is unbounded since the positive values in row 0 have only negative values in its corresponding columns. This indicates that any further pivots are pointless and that the feasible region is unbounded. This condition is the necessary and sufficient condition for unboundedness (Bazaraa, et al., 2005).

The methods shown above used the original Simplex Method. The Simplex Method has since been revised in order to save space and time. For example, the revised simplex method goes through exactly the same steps as the original simplex method, but stores all of the pertinent information in a smaller tableau. In the revised method, the

tableaux contain only the columns that correspond to the basis and the RHS of the tableau. Of course the variables are labeled on each row so that it is obvious which variables are part of the basis. Calculations are then done to determine $z_1 - c_1, z_2 - c_2, \cdots z_n - c_n$, and this determines which new variables enter and leave the basis. Then the vector $\left[\frac{z_k - c_k}{y_k}\right]$ is added to the right of the tableau, and the pivot is performed around the appropriate element. As each variable enters and leaves the basis, the tableau is updated, and only the columns in the basis are included in the new tableau. This is a less tedious method since unneeded calculations are ignored.

Finally, the concept of duality is briefly addressed. For every linear programming problem that is solved, there is another related problem that is being solved simultaneously and invisibly. This second problem can be used to obtain the solution to the original problem, and its variables give very useful information about the optimal solutions to the original problem as well. The original problem is usually called the *primal* problem, and the related problem is usually called the *dual* problem. In fact, given the primal problem and its dual below, if x^* and y^* are the optimal solutions of the primal and dual problems respectively, then $cx^* = y^* b$. So, the dual problem gets solved "invisibly" when solving the primal problem.

Suppose that the primal linear programming problem is defined as

Minimize $z = cx$

Subject to $Ax \geq b$

$x \geq 0$

Then the dual problem can be written as

$$\begin{aligned} &\text{Maximize} && z = yb \\ &\text{Subject to} && yA \le c \\ & && y \ge 0 \end{aligned}$$

There is exactly one dual variable for each constraint in the primal problem, and exactly one dual constraint for each variable in the primal problem.

Interior Point Methods vs. Simplex

While the Simplex Method solves linear programming problems using the trial-and-error approach of moving from extreme point to extreme point, Interior Point Methods move through the center of the feasible region of the linear programming problem and converge toward an extreme point. At each iteration, various techniques are used to find the general direction of the optimal solution. There has been much debate in recent days about which method is better. The Simplex Method has been revised and refined over many years in attempts to deal with the many complications of linear programming problems. Interior Point Methods were first taken seriously when Karmarkar (1984) discovered the polynomial time interior point method for linear programming. Since then, many variants of interior point methods have been devised. These include affine-scaling methods that create an ellipsoid to approximate the feasible region and optimize the objective function, potential reduction methods that construct a potential (or merit) function and minimizes it subject to primal-dual constraints, central path methods that attempt to find an approximate solution along the central path of the feasible region, and even infeasible interior-point methods that allow the algorithm to begin without an initial feasible solution. There are many other interior point methods

besides the ones described above, and each has particular advantages and disadvantages (Singh & Singh, 2002).

The current debate generally centers around whether the Simplex Method or interior point methods are the best approach to linear programming problems in general. Of course the answer is rather complex. No one method is best in all cases (Singh & Singh, 2002). Vanderbei (1996) tested 81 problems using both the Simplex Method and one of the interior point methods called the Homogeneous Self-Dual Method. The largest problem he tested contained 3000 constraints and 14,000 variables. This is currently considered only a medium-sized linear programming problem. Vanderbei (1996) found that at least one method had numerical problems or insufficient memory with seven of the problems, the Simplex Method was superior for 55 of the problems, and the interior point method was superior for 18 of the problems. While this might seem to indicate that the Simplex Method outperforms at least that particular interior point method, it must be noted that the interior point method was more than 800 times faster than the Simplex Method on the largest problem in the test. In general, extensive testing has shown that interior point methods are far superior to the Simplex Method for large linear programming problems (Singh & Singh, 2002). This makes sense since a relatively small number of extreme points can be tested quickly using the Simplex Method, but this approach becomes very time-consuming when the problems get big. The reason for the inferior performance of interior point algorithms on small problems is the relative amount of work required for each iteration. In general, each iteration of an interior point algorithm is more computationally intensive than the Simplex Method, but interior point algorithms take fewer iterations to reach an optimal solution (Rardin, 1998).

One final note about interior point algorithms concerns degenerate linear programming problems. The Simplex Method performs very poorly on degenerate problems, while there is very little adverse effect on the performance of interior point algorithms (Singh & Singh, 2002). On the other hand, the Simplex Method has the advantage that computations can be updated and/or reused from any iteration, and this can save considerable time (Singh & Singh, 2002). In the end, the choice of methods to solve a particular linear programming problem should depend on the size of the problem as well as the attributes of the problem. Various things must be considered including degeneracy, difficulty or ease in obtaining an initial feasible solution, as well as the number of variables and the nature of the constraints.

Network Algorithms

Network algorithms are methods of solving linear programming problems with specific types of constraint structures. One of the special cases of problems that can be solved by network algorithms is the transportation problem. This problem is generally designed to minimize costs subject to meeting minimum demand constraints and maximum supply constraints in each of several locations. Sometimes there is a penalty associated with any unmet demand. When total supply is equal to total demand, then a transportation problem is called *balanced.* If total supply exceeds total demand, a transportation problem can be balanced by adding an extra (dummy) demand point that has demand equal to the excess supply. If a transportation problem has a total supply that is less than total demand, then no feasible solution exists for the problem. If total supply is less than total demand, then the penalty associated with unmet demand can balance the

transportation problem (Bazaraa, et al., 2005; Winston, 2004). An example is shown below.

Example 6. (Winston, 2004, p. 371, #1)

A company supplied goods to three customers, who each require 30 units. The company has two warehouses. Warehouse 1 has 40 units available, and warehouse 2 has 30 units available. The costs of shipping 1 unit from warehouse to customer are shown in Table 3 below. There is a penalty for each unmet customer unit of demand: With customer 1, a penalty cost of \$90 is incurred: with customer 2, \$80; and with customer 3, \$110. Formulate a balanced transportation problem to minimize the sum of shortage and shipping costs.

Table 3. *Warehouse Shipping Costs.*

	To		
From	Customer 1	Customer 2	Customer 3
Warehouse 1	\$15	\$35	\$25
Warehouse 2	\$10	\$50	\$40

Solution.

Let x_{ij} be the number of units from warehouse *i* shipped to customer *j*.

Then the costs can be written as follows:

$15x_{11} + 35x_{12} + 25x_{13}$ (Cost of shipping units from warehouse 1)
$10x_{21} + 50x_{22} + 40x_{23}$ (Cost of shipping units from warehouse 2)

If x_{3j} is the number of units of unmet demand for customer *j*, then the costs for unmet demand are:

$90x_{31} + 80x_{32} + 110x_{33}$

Also, the supply constraints are:

$x_{11} + x_{12} + x_{13} \leq 40$
$x_{21} + x_{22} + x_{23} \leq 30$

The demand constraints are:

$x_{11} + x_{21} + x_{31} \geq 30$
$x_{12} + x_{22} + x_{32} \geq 30$
$x_{13} + x_{23} + x_{33} \geq 30$

The total demand is 30 + 30 + 30 = 90 units, and the total supply is 40 + 30 = 70 units. Therefore, there is a shortage of $90 - 70 = 20$ units.

Transportation problems can be modeled in a very efficient table/matrix that shows all the information. For example, the above problem takes a large amount of space several inequalities and expressions to show all the information. However, the entire problem can be modeled in the table below:

Table 4. *Example 6 Tableau*

	Customer 1	Customer 2	Customer 3	
Warehouse 1	15	35	25	40
Warehouse 2	10	50	40	30
Shortage	90	80	110	20
	30	30	30	

In Table 4 above, the costs are in the center part of the table, the supply constraints are in the right hand column, and the supply constraints are in the bottom row of the table. The shortage costs are shown in the "Shortage" row of the table.

In simple linear programming problems, slack variables often provide the initial feasible solution needed in order to start the pivots of the Simplex Method or other algorithm. If an initial feasible solution is not obvious, artificial variables are added, and then a second phase must be added to the Simplex Method in order to find an initial feasible solution that uses only legitimate variables. In transportation problems, the special structure of the constraint matrix is used to find initial feasible solutions. One of the interesting features of the constraint matrix is that it has rank of $m + n - 1$, where m is the number of supply constraints (rows), and n is the number of demand constraints (columns). The fact that the constraint matrix is less than full rank means that one of the constraint equations should be deleted in order to have a set of linearly independent constraints. Furthermore, a simple algorithm called Vogel's Method usually provides an initial feasible solution that has relatively small costs to start with. This is an attractive feature, since an initial solution with relatively small costs decreases the number of pivots necessary to reach an optimal solution. It should be noted that there are other methods of obtaining an initial feasible solution to the transportation problem, but Vogel's Method often provides a starting solutions that requires fewer pivots than other methods Bazaraa, et al., 2005; Winston, 2005).

Another attractive feature of transportation problems is that the Transportation Simplex Method requires only addition and subtraction in its pivots, greatly decreasing the level of complexity and number of computations necessary to find an optimal solution. This simplification is possible because the transportation matrix is *unimodular*; that is, the determinant of every square submatrix formed from it will have a value of 0, –1 or 1. This implies that every basis of the transportation matrix and its inverse will have

a determinant of either 1 or –1. A detailed derivation and explanation of the algorithm would consume a great deal of space, but a simple example of a transportation problem solved by the Transportation Simplex Method is shown below.

Example 7. (Winston, 2005, p. 389, #1)

The transportation problem from Example 6 is now solved using the Transportation Simplex Method. The first step is to find an initial feasible solution. Vogel's method results immediately in the optimal solution, so the Northwest Method is chosen to find the initial feasible solution in order to illustrate the pivots in the Transportation Simplex Method. The Northwest method yields the following initial tableau:

Initial tableau for Example 7

	v 15	35	25	
u	15	35	25	
0	30	10		40
	10	50	40	
15		20	10	30
	90	80	110	
85			20	20
	30	30	30	

Now the $\bar{c}_{ij}$ values need to be computed.

$$u_1 = 0$$
$$u_1 + v_1 = 15$$
$$u_1 + v_2 = 35$$
$$u_2 + v_2 = 50$$
$$u_2 + v_3 = 40$$
$$u_3 + v_3 = 110$$

Solving these equations simultaneously yields:

$$u_1 = 0;\ u_2 = 15;\ u_3 = 85;\ v_1 = 15;\ v_2 = 35;\ v_3 = 25$$

Now,
$\bar{c}_{13} = u_1 + v_3 - 25 = 0 + 25 - 25 = 0;$
$\bar{c}_{21} = u_2 + v_1 - 10 = 15 + 15 - 10 = 20;$
$\bar{c}_{31} = u_3 + v_1 - 90 = 85 + 15 - 90 = 10;$
$\bar{c}_{32} = u_3 + v_2 - 80 = 85 + 35 - 80 = 40;$

The largest of the $\bar{c}_{ij}$ values is $\bar{c}_{32} = 40$, so x_{32} will enter the basis.

A loop involving x_{32} and some of the basic variables is (3, 2), (3, 3), (2, 3), (2, 2). Of these, (3, 3) and (2, 2) are odd. Of these, (3, 3) has the largest cost, so x_{33} will leave the basis. The new tableau after the pivot is shown below.

First iteration tableau for Example 7

	v 15		35		25	
u		15		35		25
0	30		10			40
		10		50		40
15			0		30	30
		90		80		110
85			20			20
	30		30		30	

Now, recomputing $\bar{c}_{ij}$, the largest value is $\bar{c}_{21} = 20$, so x_{21} enters the basis. The loop that involves x_{21} and some of the basic variables is (2, 1), (1, 1), (1, 2), (2, 2). The odd elements are (1, 1) and (2, 2), and these determine the pivot. This method continues until all of the $\bar{c}_{ij}$ values are non-positive, and the final tableau yields the optimal solution. In this problem the optimal solution is 10 units should be sent from Warehouse 1 to Customer 2, 30 units should be sent from Warehouse 1 to Customer 3, 30 units should be sent from Warehouse 2 to Customer 1, and 20 units of Customer 2's demands will remain unsatisfied.

The transportation problem has several variants. Assignment problems are a special case of the transportation problem that have many applications. In practice, an assignment problem attempts to assign specific supply points to specific demand points while minimizing the cost of the overall assignment choice. Mathematically, an assignment problem is a balanced transportation problem where all the supply and demand constraints are equal to one. Because the right hand side of each constraint is

equal to one, it is clear that each of the x_{ij} variables must be equal to either zero or one. These problems are highly degenerate in nature since for an $m \times m$ problem, each basic feasible solution has exactly *m* basic variables equal to one, and $m - 1$ basic variables equal to zero Bazaraa, et al., 2005; Winston, 2005).

The structure of the cost matrices in assignment problems result in further simplifications to the network algorithms used to solve them. One popular method for solving assignment problems is called the Hungarian Method. Though assignment problems can be solved by the Network Simplex Method, the Hungarian Method is simpler and easier to implement (Winston, 2005).

The Hungarian Method is comprised of three basic steps.

1. Find the minimum element in each row and each column of the $m \times m$ cost matrix. First construct a new matrix by subtracting the minimum value for each row from each element of that row. Then construct a second new matrix by following the same procedure for each column.
2. Draw the minimum possible number of horizontal and vertical lines through the rows and columns of the reduced cost matrix so that all of the zeros in the matrix are covered. If fewer than *m* lines are needed, go to step 3. If *m* lines are required, then an optimal solution can be obtained from the covered zeros in the reduced matrix.

3. Find the smallest nonzero element in the reduced cost matrix that is not covered by any line. Call this value *k*. Subtract *k* from each element in the reduced cost matrix that is not covered, and add *k* to each element that is covered by *two* lines. Go back to step 2.

(Winston, 2005)

An example of a balanced assignment problem being solved by the Hungarian Method is shown below.

Example 8. (Winston, 2005, p. 398 #2)

Doc Councillman is putting together a relay team for the 400-meter relay. Each swimmer must swim 100 meters of breaststroke, butterfly, backstroke, or freestyle. Doc believes that each swimmer will attain the times given in Table 5 below. To minimize the team's time for the race, which swimmer should swim which stroke?

Swimmer	Free	Breast	Fly	Back
Gary Hall	54	54	51	53
Mark Spitz	51	57	52	52
Jim Montgomery	50	53	54	56
Chet Jastremski	56	54	55	53

Set up the assignment problem as a simple matrix and begin the algorithm.

				Row min.
54	54	51	53	51
51	57	52	52	51
50	53	54	56	50
56	54	55	53	53

3	3	0	2
0	6	1	1
0	3	4	6
3	1	2	0
0	1	0	0

Column Min

3	2	0	2
0	5	1	1
0	2	4	6
3	0	2	0

It took only three lines to cross out all zeros, so another step is necessary to complete the problem. The minimum uncovered element is 1, so subtract 1 from each uncovered element and add 1 to each twice-covered element.

3	3	0	2
0	6	1	1
0	3	4	6
3	1	2	0

It took 4 lines to cover all zeros, so the optimal solution can be found among the covered zeros. The only covered zero in column 3 is x_{13}, so $x_{13} = 1$. The only covered zero in column 2 is x_{42}, so $x_{42} = 1$. $x_{44} = 0$ since row 4 cannot be used again. Therefore, $x_{24} = 1$ since it is the only other zero in column 4. $x_{21} = 0$ since row 2 cannot be used again, and this leaves $x_{31} = 1$. So, $x_{13} = 1$, $x_{24} = 1$, $x_{31} = 1$, $x_{42} = 1$ is optimal. The time required is $51 + 52 + 50 + 54 = 207$ seconds.

The final problem to be dealt with in this subsection is the transshipment problem. This is another special case of the general transportation where shipment is allowed between supply points and demand points, and there may also be transshipment points through which goods may be shipped as they are in transit from a supply point to a demand point. These problems must generally be transformed into balanced transportation problems in order to find optimal solutions Bazaraa, et al., 2005; Winston, 2004). Specific algorithms to solve this special type of transportation problem will be discussed at length in the depth section of this paper.

This subsection provided an introduction to several special cases of linear programming problems including the transportation problem, the assignment problem, and the transshipment problem. Several specialized algorithms for solving these problems were also introduced. The depth section of the paper will include more in-depth

discussions of these problems as well as an introduction to network models. Dijkstra's algorithm, the Ford-Fulkerson Method for solving Maximum-Flow problems, the Minimum Spanning Tree (MST) algorithm, the shortest-path problem, and the Network Simplex method will also be discussed. Recent research in the field will be integrated with these topics to provide a current viewpoint on the subject area. The next part of this section will discuss nonlinear programming problems.

Nonlinear Programming

Until this point, only linear programming problems have been considered. In other words, both the objective function and the constraints have been assumed to be linear. In this section, the assumption of linearity is lifted, and nonlinear optimization problems are considered. One simple case of a nonlinear programming problem can be written as

Maximize (or minimize) $f(x)$

Subject to $x \in [a, b]$, where a and b are real numbers, $a = \infty$, or $b = \infty$.

This problem can be theoretically solved by finding all the local maxima or minima and then finding the global maximum or minimum by finding the local extreme point having the largest (or smallest) value of $f(x)$. In cases of nonlinear programming problems with one variable, many problems can be solved using classic methods of calculus (Bazaraa, Sherali, & Shetty, 1993; Winston, 2004).

In general, there are three kinds of points where a local maximum or minimum can exist.

Case 1: $f'(x) = 0$, and $a < x < b$

Case 2: $f'(x)$ is undefined at some point $a < x < b$

Case 3: A local maximum or minimum exists at $x = a$ or $x = b$

Well known theorems state that if a nonlinear programming problem like the one stated above has $f(x)$ concave in a feasible region, then any local maximum for the nonlinear programming problem is an optimal solution to the nonlinear programming problem. Likewise, if a nonlinear programming problem like the one stated above has $f(x)$ convex in a feasible region, then any local minimum for the nonlinear programming problem is an optimal solution to the nonlinear programming problem. Further, if $f''(x)$ exists for all x in a convex set S (where S is a feasible region for the problem), then $f(x)$ is a convex function on S if and only if $f''(x) \geq 0$ for every x in S, and $f(x)$ is a concave function on S if and only if $f''(x) \leq 0$ for every x in S (Bazaraa, et al., 1993; Winston, 2004).

Example 9. (Winston, 2004, p. 636, #1)

Determine whether $f(x) = x^3;\ S = [0, \infty)$ is convex, concave, or neither. Find the maximum of the nonlinear programming problem.

Solution.

$f'(x) = 3x^2$. $f''(x) = 6x \geq 0$ when $x \geq 0$, so $f(x)$ is convex on S. Now, since x^3 is monotonically increasing when $x \geq 0$, the maximum can occur only at one of the endpoints of S. $f(x) = 0$ when $x = 0$, and $f(x)$ is unbounded when $x = \infty$, so this problem is unbounded, and has no maximum on S.

Sometimes $f(x)$ is not differentiable, or it may not be possible to solve $f'(x) = 0$ for x. Fortunately, algorithms have been developed to deal with these cases. The basic

assumption required for most of these algorithms is that the function $f(x)$ be *unimodal* on [a, b]. A function $f(x)$ is unimodal on [a, b] if for some point $\bar{x}$ on the interval [a, b], the function $f(x)$ is monotonically increasing on $[a, \bar{x}]$ and monotonically decreasing on $[\bar{x}, b]$ (Winston, 2004). Unimodal functions have the convenient property that the function will have only one local maximum $(\bar{x})$ on the interval [a, b], and that local maximum will solve the nonlinear programming problem.

Maximize $f(x)$ subject to $a \leq x \leq b$.

Most of the algorithms that use the assumption of a unimodal function begin by judiciously selecting two points (x_1 and x_2) in the interval [a, b] such that $x_1 \leq x_2$, and then decreasing the size of the interval in which $\bar{x}$ must lie by evaluating $f(x_1)$ and $f(x_2)$ and comparing the results. Either $f(x_1) < f(x_2)$, $f(x_1) > f(x_2)$, or $f(x_1) = f(x_2)$, and it is possible to use this information to deduce that $\bar{x} \in (x_1, b]$, $\bar{x} \in [a, x_2)$, or $\bar{x} \in [a, x_2]$, respectively. This *interval of uncertainty* is then used to find two new points to repeat the process of reducing the interval of uncertainty again and again until the length of the interval *S* is sufficiently small Bazaraa, et al., 1993; Winston, 2004). The way that each successive x_1 and x_2 are chosen depend on the specific algorithm being used. One such algorithm is called the *Golden Section Search*, and uses the golden ratio to select x_1 and x_2. The golden ratio is defined as the two parts of a line segment of length 1 such that

$$\frac{\text{Length of whole line}}{\text{Length of larger part of line}} = \frac{\text{Length of whole line}}{\text{Length of smaller part of line}}, \text{ and is the number } \frac{\sqrt{5}-1}{2}.$$

(Interestingly, the golden ratio has many applications in mathematics of which the *Golden Section Search* is only one.)

Algorithms such as the *Golden Section Search* and other algorithms of its type have been implemented in some spreadsheet programs such as *Lotus 1-2-3* and *Excel*, as well as Computer Algebra Systems such as *Mathematica*, though different programs use different specific algorithms.

Many nonlinear optimization problems are multivariate rather than univariate. Calculus can be used to solve some nonlinear programming problems with multiple variables. Unconstrained problems of the type

Maximize (or Minimize) $f(x_1, x_2, \cdots, x_n)$ (1)

Subject to $(x_1, x_2, \ldots, x_n) \in \Re^n$

can be solved with classic methods of multivariable calculus.

Define the Hessian of $f(x_1, x_2, \cdots x_n)$ as the $n \times n$ matrix whose *ij*th entry is $\frac{\partial^2 f}{\partial x_i \partial x_j}$. Then an *i*th principal minor of the Hessian is the determinant of any $i \times i$ matrix obtained by deleting *n*– *i* rows and its corresponding *n* – *i* columns of the matrix. If $f(x_1, x_2, \cdots x_n)$ has continuous second-order partial derivatives for every point $x = (x_1, x_2, \cdots x_n) \in S$, then $f(x_1, x_2, \cdots x_n)$ is convex on *S* if and only if all principal minors of its Hessian are nonnegative on $x \in S$, and $f(x_1, x_2, \cdots x_n)$ is concave on *S* if and only if all nonzero principal minors of its Hessian have the same sign as $(-1)^k$, for $k = 1, 2, \cdots, n$, for $x \in S$ (Bazaraa, et al., 1993; Winston, 2004).

Multivariable calculus also shows that if $\bar{x}$ is a local extreme point for (1), then $\frac{\partial f(\bar{x})}{\partial x_i} = 0$, and that any point satisfying this equation for *i* = 1, 2, …, *n*, then $\bar{x}$ is called a *stationary point* of *f*. Then, if the Hessian $H_k(\bar{x}) > 0,\ k = 1, 2, \cdots, n$, then a stationary

point $\bar{x}$ is a local minimum for (1), and if $H_k(\bar{x})$ is nonzero and has the same sign as $(-1)^k$, then a stationary point $\bar{x}$ is a local maximum for (1). If the Hessian is not equal to zero and neither of the previous conditions on the Hessian hold, then a stationary point $\bar{x}$ is not an extreme point of *f*. Stationary points that are not local extrema are called ***saddle points***. If $H_k(\bar{x}) = 0$ for a stationary point $\bar{x}$, then the preceding tests are inconclusive (Winston, 2004).

The methods for actually determining the extreme points are more complex for multivariate functions than for univariate functions. The *Mathematica*® module *Analyze*, shown in the Appendix of this paper, analyzes two-dimensional functions and finds maximums, minimums, and saddle points.

Example 10. (Winston, 2004, p. 658, Example 28)

The function to be analyzed is $f(x,y) = x^2y + y^3x - xy$. The analysis given by the program matches the mathematics shown in the example in the text. This example and module shows the ease with which many algorithms can be implemented from scratch in a Computer Algebra System such as *Mathematica*, or programmed in a higher level programming language such as C^{++}.

Solution:

```
In[34]:= Analyze[x^2 y + y^3x - x y]

        F → -x y + x^2 y + x y^3

          → y (-1) x + y x^2 + y x y^2

        First derivatives:

            Fx → -y + y^3 + y x 2

            Fy → -x + x^2 + x y^2 3

        Second derivatives:

            Fxx → 2 y

            Fxy → -1 + 2 x + 3 y^2

            Fyy → 6 y x

        Saddle point at {x, y} → {0, -1}

        Saddle point at {x, y} → {0, 0}

        Saddle point at {x, y} → {0, 1}

        Local Maximum at {x, y} → {2/5, -1/Sqrt[5]}

        Local Minimum at {x, y} → {2/5, 1/Sqrt[5]}

        Saddle point at {x, y} → {1, 0}
```

Another popular method for solving unconstrained multivariate maximization problems is the method of *Steepest Ascent.* It is based on a Lemma that says that if a small move δ is made from a point **v** on the function to be maximized, and that move is made in a direction **d**, then for any given δ, the maximum increase in the value of the function $f(x_1, x_2, \cdots, x_n)$ will occur if the direction is chosen as

$$\mathbf{d} = \frac{\nabla f(\mathbf{x})}{\|\nabla f(\mathbf{x})\|},$$

where $\nabla f(\mathbf{x})$ is the gradient vector of the function $f(x_1, x_2, \cdots, x_n)$, and $\frac{\nabla f(\mathbf{x})}{\|\nabla f(\mathbf{x})\|}$ is the normalized gradient of $f(x_1, x_2, \cdots, x_n)$, which will be perpendicular to the curve $f(x_1, x_2, \cdots, x_n)$. Moving away from **v** in the direction of the gradient will result in a

maximum rate of increase for the function *f*. For some nonnegative value of *t*, the movement will be from an initial point $\mathbf{v}_0$ to a new point $\mathbf{v}_1 = f(\mathbf{v}_0 + t_0 \nabla f(\mathbf{v}_0))$, subject to $t_0 \geq 0$. This new nonlinear programming problem can be solved by a method such as the Golden Section Search. Once $\|\nabla f(\mathbf{v}_2)\|$ is small enough, the algorithm is terminated and $\mathbf{v}_2$ is chosen as the approximate solution. Note that this method may lead towards relative, rather than global maximums (Bazaraa, et al., 1993; Winston, 2004).

The well known method of *Lagrange Multipliers* can be used to solve nonlinear programming problems with equality constraints. Suppose there exists the problem

Maximize (or minimize) $z = f(x_1, x_2, \cdots, x_n)$

Subject to:

$$g_1(x_1, x_2, \cdots, x_n) = b_1$$
$$g_2(x_1, x_2, \cdots, x_n) = b_2$$
$$\vdots$$
$$g_m(x_1, x_2, \cdots, x_n) = b_m$$

Then if $f(x_1, x_2, \cdots, x_n)$ is a concave function and each of the constraints is a linear function, then any point $(\bar{x}_1, \bar{x}_2, \cdots, \bar{x}_n, \bar{\lambda}_1, \bar{\lambda}_2, \cdots, \bar{\lambda}_m)$ that satisfies $\frac{\partial L}{\partial x_1} = \frac{\partial L}{\partial x_2} = \cdots = \frac{\partial L}{\partial x_n} = \frac{\partial L}{\partial \lambda_1} = \frac{\partial L}{\partial \lambda_2} = \cdots = \frac{\partial L}{\partial \lambda_m} = 0$ will give an optimal solution $(\bar{x}_1, \bar{x}_2, \cdots, \bar{x}_n)$ to the maximization problem. If the original problem is a minimization problem, then the same conditions hold, except that the function must be convex instead of concave (Bazaraa, et al., 1993; Winston, 2004).

This method is illustrated by the example below.

Example 11. (Winston, 2004, p. 669, #2)

It costs me \$2 to purchase an hour of labor and \$1 to purchase a unit of capital. If x hours of labor and y units of capital are available, then $x^{2/3}y^{1/3}$ machines can be replaced. If I have \$10 to purchase labor and capital, what is the maximum number of machines that can be produced?

Solution: The word problem reduces to the problem of maximizing $x^{2/3}y^{1/3}$, subject to the constraint that $2x + y = 10$. If $f(x,y) = x^{2/3}y^{1/3}$, then the logarithm of this function is concave, and therefore can be solved using Lagrange Multipliers. Therefore,

Maximize $\log f = \frac{2}{3}\log x + \frac{1}{3}\log y$

Subject to: $2x + y = 10$

Then $L(x,y) = \frac{2}{3}\log x + \frac{1}{3}\log y + \lambda(10 - 2x - y)$

Set $\frac{\partial L}{\partial x} = \frac{\partial L}{\partial y} = \frac{\partial L}{\partial \lambda} = 0$.

$$\frac{\partial L}{\partial x} = \frac{2}{3x} - 2\lambda = 0$$
$$\frac{2}{3x} = 2\lambda$$
$$3x = \frac{1}{\lambda}$$
$$x = \frac{1}{3\lambda}$$

$$\frac{\partial L}{\partial y} = \frac{1}{3y} - \lambda = 0$$
$$\frac{1}{3y} = \lambda$$
$$y = \frac{1}{3\lambda}$$

$$\frac{\partial L}{\partial \lambda} = 10 - 2x - y = 0$$
$$2x + y = 10$$
$$2\left(\frac{1}{3\lambda}\right) + \left(\frac{1}{3\lambda}\right) = 10$$
$$\frac{3}{3\lambda} = 10$$
$$\lambda = \frac{1}{10}$$

Finally, $x = \frac{1}{3\lambda} = \frac{10}{3}$ and $y = \frac{10}{3}$.

To end the section on nonlinear programming, the Karush-Kuhn-Tucker conditions are discussed. These are sometimes shortened and called the Kuhn-Tucker conditions. This group of theorems provides necessary and sufficient conditions for $\bar{x} = (\bar{x}_1, \bar{x}_2, \cdots, \bar{x}_n)$ to be an optimal solution for the nonlinear programming problem

$$\begin{aligned} &\text{Maximize (or minimize)} && z = f(x_1, x_2, \cdots, x_n) \\ &\text{Subject to:} && g_1(x_1, x_2, \cdots, x_n) \le b_1 \\ & && g_2(x_1, x_2, \cdots, x_n) \le b_2 \\ & && \vdots \\ & && g_m(x_1, x_2, \cdots, x_n) \le b_m \end{aligned} \qquad (2)$$

Notice that this problem, unlike other problems in this section, has inequality constraints rather than inequality constraints. If the inequality constraints are greater than or equal to constraints, then they must be put into the standard form above by multiplying the inequality through by –1. The Kuhn-Tucker conditions are broken into theorems that state necessary conditions in order for $\bar{x} = (\bar{x}_1, \bar{x}_2, \cdots, \bar{x}_n)$ to be an optimal solution to (2), and theorems that state sufficient conditions for $\bar{x} = (\bar{x}_1, \bar{x}_2, \cdots, \bar{x}_n)$ to be an optimal solution to (2) (Bazaraa, et al., 1993; Winston, 2004).

Necessary condition Theorems for (2):

Theorem T-1:

Suppose that (2) is a maximization problem. If $\bar{x} = (\bar{x}_1, \bar{x}_2, \cdots, \bar{x}_n)$ is an optimal solution to (2), then $\bar{x} = (\bar{x}_1, \bar{x}_2, \cdots, \bar{x}_n)$ must satisfy all the constraints (must be a feasible solution), and there must be multipliers $\lambda_1, \lambda_2, \cdots, \lambda_m$ that satisfy

$$\frac{\partial f(\bar{x})}{\partial x_j} - \sum_{i=1}^{i=m} \bar{\lambda}_i \frac{\partial g_i(\bar{x})}{\partial x_j} = 0 \quad (j = 1,2,\cdots,n)$$
$$\bar{\lambda}_i [b_i - g_i(\bar{x})] = 0 \quad (i = 1,2,\cdots,m)$$
$$\bar{\lambda}_i \geq 0 \quad (i = 1,2,\cdots,m)$$

Theorem T-2:

If (2) is a minimization problem, then the solution must satisfy the constraints as in the theorem above, and there must be multipliers $\lambda_1, \lambda_2, \cdots, \lambda_m$ that satisfy

$$\frac{\partial f(\bar{x})}{\partial x_j} + \sum_{i=1}^{i=m} \bar{\lambda}_i \frac{\partial g_i(\bar{x})}{\partial x_j} = 0 \quad (j = 1,2,\cdots,n)$$
$$\bar{\lambda}_i [b_i - g_i(\bar{x})] = 0 \quad (i = 1,2,\cdots,m)$$
$$\bar{\lambda}_i \geq 0 \quad (i = 1,2,\cdots,m)$$

(Bazaraa, et al., 1993; Winston, 2004). The $\lambda_i's$ can be thought of as similar to the Lagrange multipliers in the preceding method.

In many conditions, the values of x_i are restricted to be nonnegative. The Kuhn-Tucker conditions can also be used to find the optimal solution to the following problem:

Maximize (or minimize) $z = f(x_1, x_2, \cdots, x_n)$

Subject to:

$$\begin{aligned} g_1(x_1, x_2, \cdots, x_n) &\le b_1 \\ g_2(x_1, x_2, \cdots, x_n) &\le b_2 \\ &\vdots \\ g_m(x_1, x_2, \cdots, x_n) &\le b_m \\ -x_1 &\le 0 \\ -x_2 &\le 0 \\ &\vdots \\ -x_n &\le 0 \end{aligned} \qquad (3)$$

In this case, the multipliers $\mu_1, \mu_2, \cdots \mu_n$ are associated with the non-negativity constraints on the x_i values.

Necessary condition Theorems for (3):

Theorem T-3:

Suppose that (2) is a maximization problem for (3). If $\bar{x} = (\bar{x}_1, \bar{x}_2, \cdots, \bar{x}_n)$ is an optimal solution to (2), then $\bar{x} = (\bar{x}_1, \bar{x}_2, \cdots, \bar{x}_n)$ must satisfy all the constraints (must be a feasible solution), and there must be multipliers $\lambda_1, \lambda_2, \cdots, \lambda_m, \mu_1, \mu_2, \cdots, \mu_n$ that satisfy

$$\begin{aligned} \frac{\partial f(\bar{x})}{\partial x_j} - \sum_{i=1}^{i=m} \bar{\lambda}_i \frac{\partial g_i(\bar{x})}{\partial x_j} + \mu_j &= 0 \quad (j = 1,2,\cdots,n) \\ \bar{\lambda}_i \left[b_i - g_i(\bar{x}) \right] &= 0 \quad (i = 1,2,\cdots,m) \\ \left[\frac{\partial f(\bar{x})}{\partial x_i} - \sum_{i=1}^{i=m} \bar{\lambda}_i \frac{\partial g_i(\bar{x})}{\partial x_j} \right] \bar{x}_j &= 0 \quad (j = 1,2,\cdots,n) \\ \bar{\lambda}_i &\ge 0 \quad (i = 1,2,\cdots,m) \\ \bar{\mu}_j &\ge 0 \quad (j = 1,2,\cdots,n) \end{aligned}$$

Theorem T-4:

If (3) is a minimization problem, then the solution must satisfy the constraints as in the theorem above, and there must be multipliers $\lambda_1, \lambda_2, \cdots, \lambda_m, \mu_1, \mu_2, \cdots, \mu_n$ that satisfy

$$\frac{\partial f(\bar{x})}{\partial x_j} + \sum_{i=1}^{i=m} \bar{\lambda}_i \frac{\partial g_i(\bar{x})}{\partial x_j} - \mu_j = 0 \quad (j = 1,2,\cdots,n)$$

$$\bar{\lambda}_i \left[b_i - g_i(\bar{x}) \right] = 0 \quad (i = 1,2,\cdots,m)$$

$$\left[\frac{\partial f(\bar{x})}{\partial x_i} + \sum_{i=1}^{i=m} \bar{\lambda}_i \frac{\partial g_i(\bar{x})}{\partial x_j} \right] \bar{x}_j = 0 \quad (j = 1,2,\cdots,n)$$

$$\bar{\lambda}_i \geq 0 \quad (i = 1,2,\cdots,m)$$

$$\bar{\mu}_j \geq 0 \quad (j = 1,2,\cdots,n)$$

(Bazaraa, et al., 1993; Winston, 2004).

In addition to the theorems that express necessary conditions for an optimal solution to (2) or (3), there are also Kuhn-Tucker conditions that provide sufficient conditions for $\bar{x} = (\bar{x}_1, \bar{x}_2, \cdots, \bar{x}_n)$ to be an optimal solution to (2) or to (3).

Theorem T-5:

Suppose that (2) is a maximization problem. If $z = f(x_1, x_2, \cdots, x_n)$ is a concave function and $g_1(x_1, x_2, \cdots, x_n) \cdots g_m(x_1, x_2, \cdots, x_n)$ are convex functions, then any point $\bar{x} = (\bar{x}_1, \bar{x}_2, \cdots, \bar{x}_n)$ that satisfies Theorem T-1 is an optimal solution to (2). If (3) is a maximization problem, $z = f(x_1, x_2, \cdots, x_n)$ is a concave function, and $g_1(x_1, x_2, \cdots, x_n) \cdots g_m(x_1, x_2, \cdots, x_n)$ are convex functions, then any point $\bar{x} = (\bar{x}_1, \bar{x}_2, \cdots, \bar{x}_n)$ that satisfies Theorem T-3 is an optimal solution to (3).

Theorem T-6:

Suppose that (2) is a minimization problem. If $z = f(x_1, x_2, \cdots, x_n)$ is a convex function and $g_1(x_1, x_2, \cdots, x_n) \cdots g_m(x_1, x_2, \cdots, x_n)$ are convex functions, then any point $\bar{x} = (\bar{x}_1, \bar{x}_2, \cdots, \bar{x}_n)$ that satisfies Theorem T-2 is an optimal solution to (2). If (3) is a minimization problem, $z = f(x_1, x_2, \cdots, x_n)$ is a convex function, and

$g_1(x_1, x_2, \cdots, x_n) \cdots g_m(x_1, x_2, \cdots, x_n)$ are convex functions, then any point $\bar{x} = (\bar{x}_1, \bar{x}_2, \cdots, \bar{x}_n)$ that satisfies Theorem T-4 is an optimal solution to (3). (Bazaraa, et al., 1993; Winston, 2004)

While the Kuhn-Tucker theorems are quite useful in solving certain nonlinear programming problems, many problems do not fit the conditions of the Theorems. Nonlinear programming quickly becomes a very complex area of study, often requiring the creation of custom algorithms to solve one real-world problem after another. Furthermore, even when algorithms work, the problem of local, rather than global, maximums and minimums often present a problem to investigators (Bazaraa, et al., 1993).

Conclusion

The area of optimization with constraints is clearly a huge area that forms much of the backbone of operations research. This section has introduced the very basics of linear programming, network algorithms, and nonlinear programming problems. In the Depth section of the paper, network algorithms will be explored in more depth. Furthermore, recent articles in the area will be read and critiqued to add additional depth to the overall paper, and to provide a glimpse of recent research in the field.

References

Bazaraa, M., Jarvis, J., & Sherali, H. (2005). *Linear Programming and Network Flows.* New York: John Wiley & Sons.

Bazaraa, M., Sherali, H., & Shetty, C. (1993). *Nonlinear Programming: Theory and Algorithms.* New York: John Wiley & Sons.

Karmarkar, N. (1984). A new polynomial time algorithm for linear programming, *Combinatorica, 4,* 373–395.

Rardin, R. (1998). *Optimization in Operations Research.* Upper Saddle River, NJ: Prentice Hall.

Singh, J. N., & Singh, D. (2002). Interior-point methods for linear programming: a review, *International Journal of Mathematical Education in Science and Technology, 33(3),* 405–423.

Vanderbei, R. J. (1996). *Linear Programming: Foundations and Extensions.* Dordrecht: Kluwer.

Winston, W. L. (2004). *Operations Research: Applications and Algorithms*. Belmont, CA: Brooks Cole Publishing Company.

Appendix: Mathematica module "Analyze2D"

The following is the text of the *Mathematica*® module to analyze two dimensional functions. An example using this module is shown in the section on nonlinear programming.

```
(* LTS Proprietary. Author: Terri Bittner *)
Required input: Analyze[function name,function
expression,independent variables ({x,y}),showDetails(True or
False)]. All inputs are optional, except for the function
expression.

Analyze[F_Symbol:F,f_,showDetails_Symbol:True]:=Analyze[F,f,
{x,y},True]
In[9]:=
Analyze[F_Symbol:F,f_,{x_Symbol,y_Symbol},showDetails_Symbol
:True]:=Module[

(*Declare Variables*)

{A, B, C, fx, fy, fxx, fxy, fyy, tmp, roots, workrule,i},

plusAtt=Attributes[Plus];
timesAtt=Attributes[Times];
ClearAttributes[Plus,Orderless];
ClearAttributes[Times,Orderless];

(*Take first and second order partial Derivatives*)

fx = D[f,x];

fy = D[f,y];

fxx = D[D[f,x],x];

fxy = D[D[f,x],y];

fyy = D[D[f,y],y];


(*Find critical points*)

tmp = Solve[{fx == 0, fy==0}, {x,y}];

roots = {x, y} /. tmp;

If[showDetails,
```

```
  Print[F," = ",f];
  If[f=!=Expand[Simplify[f]],Print["   =
",Expand[Simplify[f] ] ] ];
  Print["First derivatives:"];
  Print["     ",F,x," = ",Expand[Simplify[fx ] ] ];
  Print["     ",F,y," = ",Expand[Simplify[fy ] ] ];
  Print["Second derivatives:"];
  Print["     ",F,x,x," = ",Expand[Simplify[fxx ] ] ];
  Print["     ",F,x,y," = ",Expand[Simplify[fxy ] ] ];
  Print["     ",F,y,y," = ",Expand[Simplify[fyy ] ] ];
  ]; (* end if *)


(*Loop through critical points and test whether max, min

saddle point or neither*)

For [i = 1, i<=Length[roots], i++,




  workrule = MapThread[Rule, {{x,y}, roots[[i]]}];

  A = fxx /. workrule;

  B = fxy /. workrule;

  C = fyy /. workrule;

If[Head[N[roots[[i]][[1]]]]===Complex||Head[N[roots[[i]][[1]
]]]===Complex,Print["Complex critical point: ",{x,y}," = ",
roots[[i]]];Return[]];


  Which [


    B^2 - A C == 0, Print["Test failed at ",{x,y}," = ",
roots[[i]] ],

    B^2 - A C > 0,  Print["Saddle point at ",{x,y}," = ",
roots[[i]] ],

    A > 0 && C > 0,  Print["Local Minimum at ",{x,y}," = ",
roots[[i]] ],

    A< 0 && C < 0, Print["Local Maximum at  ", {x,y}," = ",
roots[[i]] ],

    Print[ "No conditions were met at ",{x,y}," = "
,roots[[i]] ],
```

```
] (*end which*)


]; (*end for*)

SetAttributes[Plus,plusAtt];
SetAttributes[Times,timesAtt];

] (*end Module*)
```

DEPTH DEMONSTRATION

DEPTH TABLE OF CONTENTS (FIX PAGE #'s)

Depth Essay

Citation 1: Project scheduling in and-or graphs: Dijkstra's algorithm

Adelson-Velsky, G. & Levner, E. (2002). Project scheduling in and-or graphs: A generalization of Dijkstra's Algorithm, *Mathematics of Operations Research, 27(3),* 504–517.

Critical Summary

This extremely well organized paper dealt with a project scheduling problem where only one predecessor task needs to be complete before the next task can be started. In the classic project scheduling problem, all tasks that precede a new task must be complete before the new task can be started. The problem dealt with in this paper has a number of applications in the areas of computer and communication networks and manufacturing systems, as well as other areas. The difficulty that arises is that the mathematical network that represents the problem has arcs of length 0. Previously developed Dijkstra-like algorithms did not accurately solve the slight variation to the classic problem.

The project scheduling problem presented is a generalization of the shortest path problem and the critical path problem. The assumptions involved are extremely similar to the assumptions used in Dijkstra's algorithm and its variants. For example, the network must be directed, the root node must be known, and bipartite graphs are assumed. The assumption about predecessor tasks is dealt with by representing nodes as either AND or OR nodes. AND nodes indicate that all predecessor tasks must be completed before a new task is begun, and OR nodes indicate that only one predecessor task needs to be completed first. The OR nodes result in connecting arcs with length 0. The problem turns into the critical path problem if there are no OR nodes, and the shortest path problem if there are no AND nodes in the problem. It is interesting to note that the constraints of the

problem are all strict equality constraints. The reasons for this are complex, but well explained in the paper.

While the paper did not deal with the process of finding the root node, the author noted that the root node must belong to all the critical subnetworks of the problem if there is a feasible solution to the network. This is stated formally as a theorem and proven. All of the other key properties of the network that ultimately lead to the authors' new algorithm were also handled formally. This included the problem's optimality conditions. The authors also explained, both theoretically and by example, why previous algorithms fail when some arc lengths are allowed to be zero.

The algorithm is fairly complex, but some key properties are that each iteration results in either a minimum starting time of an operation (task), or deletes non-critical arcs from the network. It also keeps track of both time labels and status of each node. This is accomplished by using numerical labels for the starting times, and color labels for the status. In fact, the optimal solution can be easily seen when no status nodes are red. The strongly polynomial labeling algorithm presented in this paper solves the problem in $O(p'p)$ time, where p' is the original number of AND-nodes in the network, and p is the total number of arcs in the network.

Critical Analysis

This paper was extremely well developed. The differences between the problem presented and problems that have previously been solved was beautifully explained. The properties of the network were clearly and sequentially developed and proven when necessary, and the authors took the fairly unusual step of providing an example that illustrated every important aspect of the algorithm. The examples in the paper were

thoroughly explained, while many papers gloss over examples and render them almost useless. This paper represents an important generalization of the project scheduling problem where arc lengths are allowed to be zero.

Citation 2: Network models in railroad planning

Ahuja, R., Cunha, C., & Sahin, G. (2005). Network models in railroad planning and scheduling, *Informs, Tutorials in Operations Research,* 54–101.

Critical Summary

The contributing author of this paper, Ravindra Ahuja, is also the author of perhaps the most widely used textbook on network flows. The paper reads like a well organized textbook on the main operations research problems associated with railroads. The paper described in detail the complexity of the railroad problem in comparison with other transportation problems such as airline problems or postal delivery problems. It also explained how U, S. railroads have not, until now, benefited from the operations research based scheduling and operating processes.

The paper consisted of several sections, each of which described one of the six most important planning and scheduling subproblems associated with railroads. These include the railroad blocking problem, the yard location problem, the train scheduling problem, the locomotive scheduling problem, the train dispatching problem, and the crew scheduling problem. An introduction and conclusion began and ended the paper. Each of the individual problems was described in detail, including assumptions and simplifications that were involved in formulating the mathematical problems. Each problem was then formulated mathematically, and solution methods were described. A numerical example was sometimes included along with a brief section on computational results.

Each of the six important problems associated with the optimal operation of a railroad can be formulated as a network problem. The railroad blocking problem is a problem of how to consolidate a large number of shipments into a smaller group, or

blocks of shipments, and can be formulated as a network design multi-commodity flow problem. The yard location problem is based on the solution method of the blocking problem, where the blocking algorithm must be run several thousand times to solve over the set of yard locations on a railroad. Amazingly, even this represents a simplified version of the true problem. The train scheduling problem can be represented as a large scale integer programming problem, while the planning version of the locomotive scheduling problem can be formulated as an integer multicommodity flow problem. Similarly, the train dispatching problem can be modeled as a multicommodity flow problem with side constraints, while the crew scheduling problem can be formulated as a minimum cost network flow problem where each crew is represented as a commodity that flows. Most of these problems had to be simplified, sometimes greatly simplified, in order to make the mathematical problem solvable.

While the authors do a wonderful job in describing the complexity of the problems that railroad companies face, they also mentioned the great success that the Canadian Pacific Railroad has experienced since implementing a scheduling approach to their operations based on operations research optimization techniques. The paper about the Canadian Pacific Railroad that the authors referenced is reviewed in Citation 10 below.

Critical Analysis

This paper is an excellent introduction to railroad problems. Railroads also provide an example of a very large operations research problem that involves billions of variables or more. This is important in the current study of operations research problems since more and more modern problems are this size or even larger.

The work in this paper is thorough, clear, and presented at an introductory level that is appropriate for anyone familiar with network problems. While the paper assumes previous experience with networks, it does not assume prior knowledge about railroad operations. The organization of the paper was extraordinarily good. The overall introduction was followed by thorough descriptions of each subproblem, the mathematical formulation, solution process, and comments on computational experiments. This paper should be required reading for all operations research students interested in railroads.

Citation 3: An analysis of inventory and transportation costs

Benjamin, J. (1989). An analysis of inventory and transportation costs in a constrained network, *Transportation Science, 23(3),* 177–183.

Critical Summary

This article is a key older article that is the first study of simultaneous solution of three well known problems. These are the transportation problem, the economic production lot size problem, and the economic order quantity problem. These problems represent decisions that are normally made serially in companies that wish to optimize their inventory control and flow. Older work had looked at these problems separately, at simultaneous solution of two of the three at once, or simultaneous solution of all three without constraints. Therefore this paper claimed to be the first that approaches the three constrained problems simultaneously.

The overall problem was first formulated as a total logistics problem, and then the author provided a simultaneous solution approach. He also addressed the efficiency of his approach by providing a heuristic that is more efficient than the original approach presented in the paper. When combined, the three linear programming/network problems the previously assumed independent decisions at each stage becomes a nonlinear programming problem. The basic solution approach is a reduced gradient algorithm. Of course the implementation of this class of algorithms requires an upper and lower bound. The author neatly provides the upper bound with basic calculus techniques by finding a relative minimum for the problem, and finds the lower bound with first order optimality conditions followed by solving a carefully formulated linear programming problem.

Finally, hypothetical data was studied in order to perform some rough sensitivity analysis. It was found that the simultaneous solution technique provides an advantage

over the independent technique, but the size of the advantage is dependent upon the size of the setup costs at each of the supply and demand points. The author noted that the model must actually be applied in order to ascertain the size of the advantage unless the inventory or transport costs are near zero.

Critical Analysis

This paper was very well organized, and provided some insight into the development of optimization techniques. The author was also very careful to state assumptions and limitations of his heuristic. Though these techniques have come a long way since this paper was written, each step provided a valuable link towards improvements in solution techniques and algorithms.

Citation 4: Cost allocation in a bank ATM network

Bjorndal, E., Hamers, H., & Koster, M. (2004). Cost allocation in a bank ATM network, *Mathematical Methods of Operations Research, 59,* 405–418.

Critical Summary

This unusual perspective on networks used game theory to describe network cost allocations and cost saving allocations between nodes. The specific application was bank ATM machines. These cost allocations were also explored from the perspective the solution concept of core elements of games and to whether allocations are monotonic. In a *core element,* a bank that doesn't have its own ATM can only get nonnegative cost savings for its own transactions, and this bank gets to keep half of its cost savings. The rest of the cost savings goes to the bank that owns the ATM used by the customers of the first bank. A *monotonic* allocation requires that the entire cost of transactions needs to be included in the allocation, and that no player can be disadvantaged by a new player joining the game/network.

The idea behind the cost allocations computed in this paper was that (1) costs are lowest when a bank's customer uses that bank's ATM machine, (2) costs are next lowest when a customer uses a different bank's ATM, and (3) costs are highest when customers use non-ATM methods to complete a transaction. Using these assumptions, the network was modeled by a cooperative game because it is assumed that banks that are normally competing with each other will all benefit by allowing use of its ATMs by other banks' customers. Cost allocations were given as vectors, where each element v_i of a vector represented the cost savings of bank *i*'s transactions. The entire article was devoted to computing theoretical cost allocations under different scenarios about the how the number of banks with ATMS and the number of locations are related.

Critical Analysis

The expectation in reading the abstract of this paper was that the paper would deal with cost optimality in bank ATM networking problems using game theory as a basis for solving the problem. In fact, the paper only dealt with cost saving allocations whether or not they were core and monotonic. Further, in order to compute the cost allocation vector, it is necessary to know exactly how many customers will process transactions at each bank, and with which bank each customer has an account. In some cases this would limit the usefulness of the process, specifically in cases where this previous information is not known, but decisions must be made about business relationships between banks.

While many theoretical points were made and several theorems were stated and proven, the usefulness of the results was unclear and was left unexplained. In fact, the paper completely lacked a summary, and simply ended with the statement of a theorem. However, the theorems were at least illustrated with clear examples that made the mathematics easier to follow. Unfortunately, the examples simply showed how to calculate cost allocations and were not explanatory in terms of the use of these calculations. This was unfortunate since the concept of dealing with a network application using game theory seems innovative, at least on the surface. It would be interesting to know whether or not some sort of optimality could be derived using these allocations. Note that game theory is heavily based on abstract algebra, and therefore the notation and mathematics of the paper was tedious at times.

Citation 5: The assignment problem of individual renewable resources

Burdett, R. & Kozan, E. (2004). The assignment problem of individual renewable resources in scheduling, *Asia-Pacific Journal of Operational Research, 21(3),* 355–377.

Critical Summary

This paper examined resource constrained scheduling problems in the form of attempting to assign human resources (workers) to operations (machines or tasks) in industrial environments. This problem can be trivial if the number of workers is exactly equal to the number of stations or operations. However, as either of these increase or change in various ways, the problem quickly becomes complex. This problem can also be presented as a network problem in the form of a degree constrained minimum spanning tree problem.

The constraints of the problem ensure that each operation is assigned the correct number of workers, and that each worker is assigned to at most one operation at a time. Three separate objective functions are suggested. Each requires a slightly differently formulation of the problem and different solution techniques. The first objective function is to find an assignment that balances the workload evenly amongst workers in the group. In a mathematical form, the objective is to minimize the sum of the absolute difference between each workload and the average workload. The second possible objective function minimizes the total movement of workers so that traveling distances or traveling times are minimized. The third objective aims to minimize the number of machines a worker is assigned to, which keeps a high level of specialization. This third objective function has some similarity to the objective of minimizing travel time.

After the problem, constraints, and objective functions were formulated mathematically, properties of the problems were discussed, and then possible solution techniques were discussed. Generally, the authors felt that the problem could best be solved by dividing the workers and operations into groups, and then solving the group assignment problems separately.

Mathematical models were discussed as an alternative to heuristics, but it was then explained why heuristic solutions were required. Several algorithms were then discussed, including simulated annealing and tabu search approaches. Evolutionary strategies were investigated but found to be unsuitable, but perhaps promising with further research. Finally, two constructive solution generation algorithms were explored and used to calculate high quality starting solutions for both the simulated annealing (SA) and tabu search approaches. These performed adequately for small problems, but did not reach as good of solutions on larger problems.

The various algorithms were tested on some real data and on simulated data, and the results compared. The analysis showed the (SA) approach to be superior in most cases. Tabu search gave good solutions, but were more computationally expensive, and with a couple of exceptions, solutions were not always as good as SA. Finally, the objective of minimizing total movement was found to be much more difficult for heuristics to solve than the other objective functions. Unfortunately, the real data that was used was found to be computationally prohibitive for one of the algorithms under test, and so the results for that algorithm are from the simulated data only.

Critical Analysis

This paper was not as well organized as some papers describing similar problems. It was difficult to tell where the author was going until the very end of the paper. The paper could have been improved with two simple changes. First, the abstract could have been greatly improved. It could describe the outline of the paper and the purpose of the analysis (which was to investigate and compare various solution approaches to the problem being examined) rather than simply explaining the problem itself. This could easily have been done without making the abstract longer. Second, the introduction to the paper should have explained the problem and the contents of the rest of the paper. Instead, it just started explaining the problem in detail, and the paper simply rambled on from there towards its conclusion.

Despite the weaknesses of the paper, the actual investigative work was good. The mathematics was sound and clear, and the analysis was thorough. It was disappointing that one of the algorithms was unable to process the real data because of prohibitive computational times. Simulated data is rarely as good as real data for investigating an algorithm, and the failure of one of the algorithms to converge to a solution on the real data in a reasonable amount of time would, it seems, indicate a problem with that algorithms when it comes to working with real data. At least it indicates that the algorithm proposed cannot solve the problem for some real data. Unfortunately, this was not mentioned in the paper.

Citation 6: Joint optimization for area control and network flow

Chiou, S. W. (2005). Joint optimization for area control and network flow. *Computers & Operations Research, 32(11),* 2821–2841.

Critical Summary

This paper investigated a joint optimization method for traffic control and network flow. The paper assumes Wardrop's principles in its formulation. The author does a review of past results on the same and similar problems, introduces his notation and formulation of the problem, including assumptions, and then develops the mathematical conditions for finding the local optimum of the joint optimization problem. Three different formulations are developed for the problem. Some are constrained, and the others are unconstrained. Finally, a four part heuristic is presented for finding the global optimum. Finally, the new algorithm is tested numerically against the older solutions and compared for a wide range of simulated data.

The joint problem that results from the traffic flow application is a nonlinear programming problem. Further, the objective function is not convex, which further complicates the problem. A variant of the gradient search is the basis for the new algorithm that is presented in the paper. The author's numerical results indicate that his new algorithm is very robust over a wide range of test data, and beats older algorithms in terms of finding the optimal solution faster.

Critical Analysis

The author makes many assumptions about the reader's knowledge about the specific application being discussed. For example, Wardrop's principles are stated as if they are common knowledge. Perhaps they are, but a contemporary paper on the same subject in a competing journal explained briefly what these principles are rather than

assuming that every reader knows them. While the author's development and mathematics seemed sound, it was suspicious that his algorithm would give better results on essentially all the data. While this isn't explicitly stated, it is implied, and no cases or conditions are given in the author's discussion where another algorithm gives better performance. Further, the results shown of individual runs of the new algorithm only shows comparisons with one other algorithm, while several different approaches were discussed in the survey at the beginning of the paper. A complete paper should have shown comparisons for several different competing algorithms.

Finally, the author was obviously not a native English speaker. It was annoying to find the obvious and frequent grammatical errors in the paper. The editors should have corrected the grammatical errors, and there seems no excuse for the presence of these errors in the final printed version of the paper. In fact, the presence of these errors causes one to question how carefully the mathematically material in the paper was checked.

Citation 7: The over-constrained airport gate assignment problem

Ding, H., Lim, A., Rodrigues, B., & Zhu, Y. (2005). The over-constrained airport gate assignment problem, *Computers & Operations Research, 32(7),* 1867–1880.

Critical Summary

This paper dealt with the problem of trying to assign airplanes to airport gates so that there are a minimum number of flights without gates, and so that the walking distances between gates are minimized for passengers. This is a very well-known and commonly studied problem. It is very easy to conceptualize, but the problem has many hidden issues that make it difficult to solve. For example, the concept of walking distances is not in itself simple. Walking distances depend upon how many passengers embark and disembark at each gate, how many passengers are transferring to other flights vs. leaving the airport, the distance between gates, and the distance between check-in gates to embarkation gates. Assignments of flights to gates are complicated by stochastic flight delays, costs associated with the distance between a gate and key service areas, and many other factors.

As a result of these complicating factors, past work has dealt with one or another complicating factor, but never with all of them. The authors claimed that no previous work has considered both the objective of minimizing the number of ungated flights and the objective of minimizing walking distances simultaneously. The constraints associated with this problem guarantee that (1) each flight can be assigned to one and only one gate or can alternatively be left unassigned if no gates are available, and (2) flights cannot overlap if they are assigned to the same gate.

This paper first discussed a greedy algorithm for minimizing ungated flights. After this, a simulated annealing approach (SA), an interval exchange tabu search

approach (ITS) and a hybrid simulated annealing with tabu search approach were discussed as methods to complete the problem solution. A number of search methods were examined in the course of the overall discussion between the two basic approaches. All are fairly simple to implement, and many consist of only three simple moves. They actually resemble computer sort algorithms. Finally, various simulated experiments were run using both approaches, and the results compared.

The (SA), (ITS), and hybrid methods were compared using experiments that made various assumptions about the number of flights in relation to the number of available gates and about the sizes of the problems. In general, the ITS approach gave better results than SA, and the hybrid approach gave better results than either. The SA approach had running time advantages because it was able to reach relatively good objective function values in short times. However, when the time was increased, the objective function values improved little when the SA approach was used. The hybrid approach's better results were reached in reasonable times, so it was determined to be the best solution method for the problem posed.

Critical Analysis

The approaches posed were extremely elegant, and surprisingly simple. However, the true problem is so complex that it is unlikely that such elegant solutions will solve it. The paper was very well written and led the reader to the conclusion in a step by step fashion. The main usefulness of the results will likely be to develop solutions for extensions to the simplified version of the real problem that was posed in the paper. This paper was also useful in showing the incremental approach that most researchers use in solving optimization problems.

Citation 8: Optimal network capacity planning: A shortest-path scheme

Doulliez, P. & Rao, M. (1975). Optimal network capacity planning: A shortest-path scheme. *Operations Research, 23(4),* 810–818.

Critical Summary

This older paper described a way to use Dijkstra's algorithm to solve a multiterminal network capacity planning problem. The problem presented was adapted to be a shortest path problem with two changes to the classic problem. First, the initial arc lengths (presented as the costs of an investment decision) are allowed to increase within certain limits in order to meet the demand on the network. These additional investments would be allowed when the initial arclengths (supply) do not meet the network demand. Second, some arcs are allowed to "fail" during the time horizon assumed for the network. In this case, no capacity at all is allowed through that arc, and an alternative path must be found. In the authors' problem, only one arc was allowed to fail at any one time.

The authors described dynamic programming as an alternative method for solving the problem they posed, but noted that dynamic programming does not guarantee a global optimum will be found. Dijkstra's algorithm, which was much newer at the time than it is today, was an innovative alternative for solving the problem. The results of various computations were also given as results in the paper, along with an illustrative example.

Critical Analysis

This work was quite innovative for its time, and extremely well written besides. The example at the end of the paper cleared up any possible questions about details, and the work was complete and clearly described. The computation section at the end was certainly useful at or near the time of publication, but the huge advances in computer technology render useless any comments about the running time of the algorithm. Still,

the problem and the algorithm itself were described well enough that it could be easily implemented today without further research. This paper is an example of excellent work in the field.

Citation 9: Maximal flow through a network

Ford, L. R., & Fulkerson, D. R. (1956). Maximal flow through a network. *Canadian Journal of Mathematics, 8,* 399–404.

Critical Summary

This landmark paper develops and describes what is now the well-known *Ford-Fulkerson method* for solving a maximum flow network problem. In just six pages, numerous theorems and lemmas were stated and sometimes proven, culminated in a new, and straightforward process for solving the maximum flow problem through a network.

The main problem considered in this paper is the issue of maximizing flow from one node to another in a network, where each arc has a given capacity. It was noted that this problem can be set up as an ordinary linear programming problem, with the same number of constraint equations as nodes in the network. The authors presented a much more efficient means of solving this problem when the network has certain properties. Namely the network must be planar in a restricted sense that is described in the paper. The authors also stated that this restriction is the kind of network that arises in most practical problems.

Key theorems presented in the paper include (1) the minimal cut theorem, (2) its corollary concerning adding a number *k* to the capacity of each arc that meets each cut of the network in one arc, and (3) the theorem stating the an *ab*-planar network has a chain joining *a* and *b* that meets each cut of the network exactly once, all lead directly to the Ford Fulkerson method for solving the problem discussed in this paper. Further, the authors even noted networks for which their theorems are not valid, and recognized the duality between the maximum flow problem and the minimal path problem.

Critical Analysis

This paper is amazing in the amount of key brand new information presented in such a small number of pages. It was surprising to discover the length of the paper given the importance attributed to it. Further, the information was presented clearly and thoroughly, and the insight the authors had about weaknesses and limitations to their work is quite impressive, especially given the state of the art of networks at the time the paper was written. The importance of this paper in the literature is clear after reading it, and the work done to develop the Ford Fulkerson method deserves the attention it now receives. When taken in the context of the time of the discovery, the new method presented in this paper is truly remarkable.

Citation 10: The Canadian Pacific Railway's operating plan

Ireland, P., Case, R., Fallis, J., Van Dyke, C., Kuehn, J., & Meketon, M. (2004). The Canadian Pacific Railway transforms operations by using models to develop its operating plans, *Informs, 34(1),* 5–14.

Critical Summary

This unusual paper is a cross between a paper about the business of railway companies, and a paper about the usefulness of operations research in this business. The main point of the paper was to convey the profound transformation of The Canadian Pacific Railway (CPR) after the company implemented the modeling techniques developed for them by MultiModal Applied Systems, Inc.

According to the authors, rail companies have a very long history of using a "tonnage" method for their rail operation decisions. This system suggests that the optimal way to run a railroad is to wait until a freight train is completely full before beginning a trip. If delays or cancellations of planned trips are necessary as a result, then this is a price the rail companies have traditionally been willing to pay in exchange for the cost savings they felt they were getting. It turns out that there are credible pros and cons to the tonnage system of operations. However, it turns out that a scheduling-based operation actually saves a surprising amount of money over tonnage based operations. Further, tonnage based operations result in poor customer service, and poor use of assets when compared to scheduling based operations.

Scheduling based operations is the use of operations research based techniques to decide upon fixed scheduling, routing, uses of resources, and even optimal speed of trains (which saves fuel). These techniques are used specifically to optimize train size, reduce complexity of "blocking" on the train, decrease time spent at yards, calculate running

times between yards, determine connections, and minimize fuel consumption. Strangely enough, this method not only saves surprisingly large amounts of money for the rail company, it also cuts transit times for customers, decreases overall fuel consumption, balances workloads among yards, increases the capacity of the system, reduces time that trains spend sitting in yards, and improves reliability for customers. It is a win-win situation in many ways. The solution is based on five sequential subproblems that involve forecasting, designing a plan, designing actual trains, simulating yard and train workloads by week and time of day, and passing the schedule on to planning tools that develop crew and locomotive cycle plans.

The problems that rail companies such as CPR face are incredibly, and surprisingly complex. As a result, many different algorithms, modeling techniques, and other processes must be used to solve the problem of minimizing costs. It cannot be boiled down to a single problem, at least not at this point. The authors claimed that their combination of techniques has proven to be the best in practice to date. The paper describes some of the general subproblems and solution approaches, but does not go into any mathematical detail, nor does it describe specific step by step algorithms. The article does, however, describe the entire transformation, in both culture, operations, and financial results that was seen by CPR one year after implementation of the new process. Financial improvement was staggering, all the more so because of the customer service improvements and even ecological improvements (e.g., fuel consumption decreases) seen.

Critical Analysis

This article gives a wonderful introduction to the problems faced by rail companies. Few people realize the complexity of the problem. It was particularly surprising to find that the minimization of costs could not be expressed as a single mathematical problem, but rather was so complex that it had to be expressed as a series of different problems that all had to be solved using different techniques.

Further, businesses normally see a diametrically opposed problem between customer needs and company profits. That both of these needs could be met simultaneously it truly amazing. This is probably the most interesting aspect of the material presented in the paper, and should be of interest to business people and operations researchers alike.

Citation 11: The elastic generalized assignment problem

Nauss, R. M. (2004). The elastic generalized assignment problem. *Journal of the Operational Research Society, 55(12),* 1333–1341.

Critical Summary

This short, but extremely technical challenging paper deals with an *elastic* version of the generalized assignment problem. Specifically, start with the task of assigning tasks to agents so that each task is assigned to only one agent and so that each agent's resource capacity is not violated. This problem is then modified so that the violation of resource constraints is allowed at a price. Even in this problem, the resource constraints cannot be violated without limit. The violation of resource constraints can be thought of as overtime hours. This analogy works because overtime is possible, up to a limit, but costs an additional amount over the regular hourly wage. "Undertime" is also allowed at a cost, and can be thought of as unused resources. In the analogy of overtime pay, "undertime" is the equivalent of the amount paid to an employee when no work is available.

In this new problem, the objective function sums the costs of the assignments and the undertime and overtime costs. The constrains enforce agent resource limitations, ensure that each job is assigned to exactly one agent, and place limits on the amount of "undertime" and overtime allowed.

Once the problem is formulated, the remainder of the paper deals with a suggested solution algorithm for the problem. Finally, the algorithm is tested extensively on simulated data. The author suggests the use of a *branch-and-bound* algorithm. This family of algorithms is based on the fact that it is enough to prove that a lower bound for the objective function is greater than or equal to a valid upper bound. The idea is to decrease the upper bound and at the same time increase the lower bound until a

sufficiently accurate solution is obtained. The lower bound can be increased, and the upper bound decreased using several methods. Some of the lower bound methods include the use of cuts, Lagrangian relaxation, and the use of penalties. The upper bound can be decreased through the use of feasible-solution generators. A number of techniques are discussed for efficiently adjusting the upper and lower bounds, and then the main algorithm is discussed. The computational testing indicated that the algorithm works better with smaller problems than with larger problems, but solutions to larger problems were acceptable as well.

Critical Analysis

The question posed is this paper seemed useful and practical. However, one of the first question in analyzing a paper concerning a new algorithm is whether off the shelf software would provide equally good or better solutions than the proposed algorithm to the problem being studied. The author wisely noted that while off the shelf software (such as LINGO and others) are continuously improving, the elastic generalized assignment problem is one of the classes of problems for which off the shelf software is still inadequate. Therefore, the new algorithm proposed is worth studying.

Two other notes are worth making. In discussing the development of the problem solution, the author continuously refers back to technical details from other papers that he wrote. This is confusing and frustrating since these details are not stated in the current paper at all. Therefore it is necessary to obtain the other papers in order to fully understand certain details of the paper. This should not be the case in scholarly research, as each paper should stand alone. References to other papers for backup or further details are very appropriate, but should be ancillary to the work, not primary material needed in

order to understand the material. Second, there should always be a caution to readers when an algorithm is tested only with simulated data. Simulated data is often “cleaner” than real data, and algorithms that work well with simulated data may sometimes not work as well on real data.

Citation 12: Sharing-group allocation problems

Rhee, S. (2005). Sharing-group allocation problems. *Economics Letters, 86(1),* 51–56.

Critical Summary

This short but very dense technical paper deals with the group allocation problem. Specifically, given a certain amount of a good, how does it get allocated among different groups so that the good cannot be used by a group to which it is *not* assigned, but can be used by any member of a group to which it is assigned? The paper developed one key lemma and two theorems. The terms *equality, solidarity, group consistency, Pareto efficiency,* and *symmetry* are defined mathematically, and axioms regarding these terms were given. The two main results of the paper stated that (1) if an allocation rule satisfies equal treatment of equal groups as well as solidarity, then it will allocate the same amount to all groups, and (2) there is no allocation rule that satisfies equal treatment of equal groups, solidarity, and group consistency that also satisfies Pareto optimality, symmetry, and solidarity. The other key result in the paper dealt with the uniqueness of the *egalitarian rule* of allocation.

The results developed in this paper were based on game theory and therefore the mathematics was heavily dependent on abstract algebra.

Critical Analysis

This paper was based on parts of the author's Ph.D. dissertation. There are two main problems that were quickly apparent with the paper. One is that the author's notation was awkward and therefore difficult to read, and the mathematical development was awkward and inelegant. This problem is probably due to the author's inexperience in writing technical papers. The other problem is that the author gave no indication of key

applications where his results could be used. The work is by nature abstract, but there are undoubtedly applications that inspired this particular work, and a statement of them would have greatly helped the readers' interest level and understanding of the point of the paper.

The title indicating that the paper was about allocation of resources among groups indicated an operations research application. Further, the library search that produced the paper was from keywords involving operations research and assignment problems. The paper may have been incorrectly classified, since it is more of a game theory solution to an operations research problem.

Citation 13: Universal properties of shortest paths in random potentials

Schorr, R. & Rieger, H. (2003). Universal properties of shortest paths in isotropically correlated random potentials. *The European Physical Journal B, 33,* 346–354.

Critical Summary

This paper considered the shortest path problem in two dimensions and Three dimensions, where the bonds (arcs of the network) have weights that are *isotropically correlated.* That is, the weights are correlated according to a distribution function. The network under consideration was modeled as either a two or three dimensional lattice, and the application was the chemical makeup of polymers. Dijkstra's algorithm was used to determine the optimal path from one node to another is both directed and undirected lattices. The results suggested that the correlations (of the weights) are relevant if they decay slower than a particular value that the authors calculate. The differences between the two dimensional and three dimensional cases were also discussed in relation to the weights.

Critical Analysis

The terminology used in this paper was quite different than that used when discussing classic networks algorithms or properties. This made the paper difficult to read. The interesting aspects of this paper were twofold. The first is that it was interesting to read about three dimensional networks. The second interesting aspect of the paper was the idea that the arcs of the network (called "weights" in this paper) were described as having correlations that behaved according to various distribution functions. Therefore, this work was a combination of networks and probability theory, since the correlations added a stochastic element to the network. Still, the results seemed to only be useful in

the particular application described. It would have been interesting to know if there are broader applications for which this work might be useful. The use of more classic network notation would also have been of great help in understanding the work more thoroughly.

Citation 14: Capacity uncertainty in resource-constrained assignment problems

Toktas, B., Yen, J., & Zabinsky, Z. (2006). Addressing capacity uncertainty in resource-constrained assignment problems. *Computers & Operations Research, 33(3),* 724–745.

Critical Summary

This study investigated the generalized assignment problem when the resource constraints are uncertain. This uncertainty of resource constraints adds a stochastic element to what otherwise would be a deterministic problem. The authors noted that one of the most common methods for solving this problem has been to plug in the average values of the stochastic parameters and then solve the problem deterministically. However, this method may often result in solutions far from the actual optimal solution. Another method in use is the implementation of stochastic programming-based approaches. However, this approach is not always feasible because the underlying distribution may be completely unknown.

The authors suggest two different deterministic methods for solving the stochastic problem. The first involves looking at a number of possible ways for estimating the constraint parameters, one of which is the mean that is used by many other researchers. However, a trimmed mean, median, mode, and several other estimates are also investigated. The second approach considered deterministic solutions under sampled sets of capacities, and a final solution was constructed from this information.

A number of simulated data sets were constructed so that the "real" cost of the objective function and capacity constraints were known. Several different distribution functions with various shapes were used to generate the constraints. Results showed that the trimmed mean outperformed the arithmetic mean, median, mode, and all other parameter estimators for all cases. This is likely due to the fact that outliers in the data were eliminated with the trimmed mean. The comparative performance evaluation approach (the second of the two deterministic approaches studied) outperformed the trimmed mean capacity approximation approach. When the mean was used to approximate capacities, the results generally showed higher actual costs, especially when the underlying distribution was not symmetric.

Critical Analysis

This paper was very thorough and clear in its development and explanations. If anything, the paper could have been shortened by about 25% since it was redundant and tedious in places. However, the authors used tables and graphs in a very nice way to illustrate various options and procedures. This was fortunate given that there were nine different capacity approximation approaches that would have been almost impossible to follow without a descriptive table.

Overall, this paper was nicely written, and the authors were careful to describe the problem, applications, previous work, their own work, and comparisons between their work and previous work. A final comment is that the authors' work was not as unique as expected, especially given the long buildup to it and the criticism of others' previous work on the same problem. In the end, the authors' technique was not all that different from what others had done, and in fact was really a variant of previous work on the problem rather than a brand new approach.

Citation 15: A toll pricing framework for traffic assignment problems

Yildrim, M.B. & Hearn, D. W. (2005). The first best toll pricing framework for variable demand traffic assignment problems. *Transportation Research, 39(8),* 659–678.

Critical Summary

This rather complex paper suggested a framework for toll pricing for traffic assignment problems with variable (rather than fixed) demand. In many ways the paper was a survey of previous work on toll pricing, and the authors synthesized previous work on elastic demand toll pricing theory to provide a single toll pricing framework. Other special cases of the problem were also discussed.

Traffic management is a very real problem that public transportation departments must deal with in one way or another. While economists have argued that economic efficiency can only occur when customers/users pay the full cost of products that they use, this is not considered to be practical in traffic control. The authors assumed that *marginal social cost pricing* (MSCP) be used. In this pricing scheme, users would pay toll to compensate for the negative consequences of their use of the roads. For example, tolls would be higher in congested areas in order to encourage drivers to use alternative routes. The traffic management problem is extremely complex, and the only way to formulate any version of the problem is to make many assumptions about the interests of management and users. As examples, an assumption must be made about the value of time for users. Yet, different users will certainly value time differently. The management problem and the user problem must actually be formulated separately since each group has competing objectives and constraints. One way to solve the problem is to find an equilibrium point between management and user interests.

This authors formulated the various problems by making the necessary assumptions using classic works, solved the resulting nonlinear network problems using mostly packaged optimization software, and then discussed the results. They noted that a major disadvantage of using MSCP is that all of the congested routes with traffic are being charged toll, and this is likely to be impractical because it will result in high fixed costs and maintenance costs.

Critical Analysis

The topic of the paper was fascinating, but the authors made many assumptions about the reader's knowledge about the specific problem and frameworks. It was therefore difficult at times to connect the mathematics and results of the problems to their meanings in the application. It would have been extremely helpful if the authors had allocated a bit more of the space in this rather long paper to explaining the practical meanings of the various lemmas, problem formulations, and solutions. It also would have been helpful if some of the important terms were explained at the beginning of the paper. Research papers should avoid lingo when possible, as it confuses readers who may not be in the exact field in which the author works.

DEPTH DEMONSTRATION

Introduction

In the previous section, the basic concepts and algorithms associated with linear and nonlinear programming were covered. In addition, network algorithms were covered. These are special ways to specific kinds of linear programming problems including transportation problems, transshipment problems, and assignment problems. In this section, network problems will be covered in further depth. The concept of representing network problems graphically will be introduced, as will various methods of solving these special problems.

Many linear optimization problems can best be analyzed by looking at a graphical representation of the problem. In this section transportation problems, transshipment problems, and assignment problems will be discussed by means of graphs. Several specific algorithms and problem types will be discussed. These include Dijkstra's shortest path algorithm, the shortest path problem as a transshipment problem, the Ford-Fulkerson Method for solving maximum flow problems, the critical path method, minimum-cost Network Flow problems, Minimum Spanning Tree problems, and the Network Simplex Method.

Graphs

A *network*, sometimes called a *graph*, is defined by *nodes*, and *arcs*. In a graph or network, a set of points, called *vertices* are also sometimes called nodes. An arc is an ordered pair of vertices and represents a possible direction of motion between vertices. A *chain* is a sequence of arcs such that every arc has exactly one vertex in common with the

previous arc, and a *path* is a chain in which the terminal node of each arc is the same as the first node of the next arc (Bazaraa, Jarvis, & Sherali, 2005; Winston, 2004). An example of a network is shown in Figure 1. This figure is adapted from Winston (2004), and will be used in an example of Dijkstra's Algorithm. Notice that each arc has a length associated with it. For example, the length from node 2 to node 4 is 12 units. It is often of interest to find the shortest path between two nodes. This would be of interest, for example, if the nodes represented places and the lengths between nodes can be assumed to be proportional to the cost of traveling from one node to the other (Bazaraa, Jarvis, & Sherali, 2005; Winston, 2004).

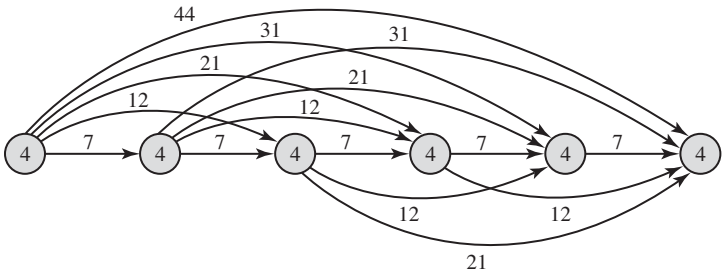


Figure 1. Example of a Network showing various Lengths between Nodes (Adapted from Winston, 2004)

Dijkstra's Shortest Path Algorithm

The basis of the concept of the *shortest path* is defined as follows: Define *d*(.) as the shortest path distances of a network. A well known theorem states that if *P* is a shortest path from a source node to some other node *k*, and c_{ij} is the arc length from node *i* to node *j*, then $d(j) = d(i) + c_{ij}$ for every arc $(i, j) \in P$. The converse is also true. Further, if the nodes in a network are examined in topological order and if for any arc (i, j) in the network it is found that $d(j) > d(i) + c_{ij}$, then we set $d(j) = d(i) + c_{ij}$. When

all the nodes have been examined once in this order, the distance labels are optimal (Ahuja, Magnanti, & Orlin, 1993).

If all the arc lengths of a network are nonnegative, then a method called *Dijkstra's Algorithm* can be used to find the shortest path from a node to all of the other nodes in the network. Before this algorithm can be used, however, topological sorting is necessary in order to determine the root, or starting, node. This root node may be one of the original nodes, or it may be an artificial node that is constructed only for modeling purposes, somewhat like artificial variables in a linear programming problem. An array has a *topological ordering* of nodes if every arc joins a lower-labeled node to a higher-labeled node (Ahuja, et al., 1993**)**.

Dijkstra's algorithm is a *label-setting* algorithm rather than a *label-correcting* algorithm. Though both approaches are iterative, label-setting algorithms designate one label as permanent (optimal) at each iteration. All other labels are designated as temporary. A label-correcting algorithm, on the other hand, considers all labels temporary until the final step, at which time permanent (optimal) labels are set. Dijkstra's algorithm as it was originally conceived was guaranteed to run in O(n^2) time (Ahuja, et al,, 1993**).**

Fortunately, label setting algorithms converge in O(n) time under certain conditions. This is a huge difference when the number of nodes is large. Specifically, a network is *acyclic* if and only if it contains a topological ordering of its nodes. A network is *directed* if the graph consists of a set N of nodes and a set A of arcs where the elements are ordered pairs of distinct nodes. If a network is acyclic and directed, then Dijkstra's algorithm will give the optimal solution in O(n) time instead of O(n^2) time. Not all networks meet these conditions, and so variants of Dijkstra's algorithm have been

devised that improve upon the original performance of the algorithm in worst-case complexity, even when some arc lengths are negative. Some of these improvements utilize *heap* (or *priority queue*) data structures, which are special data structures that allow for efficient storage and manipulation of a collection of objects (Ahuja, et al,, 1993**).**

In its simplest form, Dijkstra's algorithm is implemented by first marking node 1 with a permanent label of 0. Then each node *i* that is connected to node 1 is given a "temporary" label that is equal to the length of the arc joining node 1 to node *i.* Every node other than node 1 and the nodes connected to it are given temporary labels of ∞. Suppose now that node *i* is the $(k + 1)$th node to be labeled with a permanent label. Then node *i* is the *k*th closest node to node 1. Now the temporary label of any node *j* is the length of the shortest path from node 1 to node *j* that only passes through the $k - 1$ nodes closest nodes to node 1. Now, for every node *m* that has a temporary label and is connected to node *i* by an arc, the temporary label on node *j* should be replaced with the minimum of node *m*'s temporary label, or node *i*'s permanent label + length of the arc from *i* to *j*. Once all the permanent labels are found, it is possible to work backward to find the shortest path from node 1 to node *j*. Starting at node *j*, and finding nodes that have labels different by exactly the length of the connecting arc can do this.

This algorithm is illustrated in the example below (Bazaraa, Jarvis, & Sherali, 2005; Winston, 2004).

Example 1. (Winston, 2004, p. 418, #1)

Find the shortest path from node 1 to node 6 in Figure 1 above.

Solution:

First, notate the labels of each node from 1 to 6 by the notation (1, 7, 12, 21, 31, 44), where the first element in the array corresponds to the label for node 1, etc., and the *i*th element of the array is the direct distance from node 1 to node *i*. Permanent labels will be shown in boldface type. Now begin by labeling node 1 with a permanent label of 0. This gives the label

(**0**, 7, 12, 21, 31, 44)

Node 1 is connected directly to nodes 2, 3, 4, 5, and 6, but the minimum of these is the distance from node 1 to node 2, so node 2 gets a permanent label of 7. This gives

(**0**, **7**, 12, 21, 31, 44)

Now a comparison is made for each node connected directly to node 2.

Node 3 min{12, 7 + 7} = **12**

Node 4 min{21, 7 + 12} = 19

Node 5 min{31, 7 + 21} = 28

Node 6 min{44, 7 + 31} = 38

Clearly, node 3 gets a permanent label of 12, and nodes 4, 5, and 6 get temporary labels of 19, 28, and 38, respectively. This gives

(**0**, **7**, **12**, 19, 28, 38)

The procedure is repeated, making comparisons for each node connected to node 3.

Node 4 min{19, 12 + 7} = **19**

Node 5 min{28, 12 + 12} = 24

Node 6 min{38, 12 + 21} = 33

Node 4 now gets a permanent label of 19, and nodes 5 and 6 get temporary labels of 24 and 33, respectively. This gives

(**0**, **7**, **12**, **19**, 24, 33)

Repeat the procedure again for all nodes connected to node 4.

Node 5 $\min\{24, 19 + 7\} = \mathbf{24}$

Node 6 $\min\{33, 19 + 12\} = 31$

This gives

(**0**, **7**, **12**, **19**, **24**, 31)

Finally, the procedure is repeated to find the permanent label for node 6.

Node 6 $\min\{31, 24 + 7\} = \mathbf{31}$

The final array is (**0**, **7**, **12**, **19**, **24**, **31**).

The shortest path from node 1 to node 6 can now be found by working backwards.

Note that $31 - 24 = 7$. This is the same as the length of the arc from node 5 to node 6, so we go back to node 5.

Also, $24 - 12 = 12$, which is the same as the length of the arc from node 3 to node 5, so we go back to node 3.

Finally, $12 - 0 = 0$, which is the same as the length of the arc from node 1 to node 3.

Therefore, 1–3–6 is the shortest path from node 1 to node 6, and has total length 31.

Shortest Path Problem as a Transshipment Problem

If one views the cost of shipping one unit of goods as the length of the arc from one node to another, a network can be thought of as a transshipment problem. Goods are shipped from node *i* to node *j,* where all vertices in between are transshipment points. In

this case the goal is simply to find the shortest path from one node to the other, as this will yield the minimum cost. Note that the cost of shipping goods from node *i* to node *i* is zero. Using the methods described in the breadth section, a network can be translated into a balanced transportation problem. Note that if there is no direct path from node *i* to node *j*, then the distance between the two nodes is represented by *M*, which is a very large positive number. This is illustrated in the example below (Bazaraa, Jarvis, & Sherali, 2005; Winston, 2004).

Example 2. (Winston, 2004, p. 418, #3)

Formulate the network in Figure 2 below into a transshipment problem.

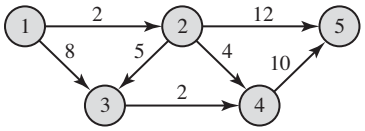


Figure 2. Network for Example 2
(Adapted from Winston, 2004)

Solution:

	Node 2	Node 3	Node 4	Node 5	Supply
Node 1	2	8	*M*	*M*	1
Node 2	0	5	4	12	1
Node 3	*M*	0	6	*M*	1
Node 4	*M*	*M*	0	10	1
Demand	1	1	1	1	1

Ford-Fulkerson Method

A problem that is complementary to shortest path problems is the maximum flow problem. Both of these problems arise as subproblems in algorithms for the minimum cost flow problems that were introduced in the breadth section and will be described in more detail later in this section. Both also have numerous applications, including feasible flow problems, scheduling on uniform parallel machines, scheduling for tanker ships, maximizing the number of daily flights on airlines, a variant of the assignment problem where only certain pairs are compatible, and many others (Ahuja, et al., 1993; Winston, 2004). While shortest path problems model the arc costs but not the arc capacities, maximum flow problems model arc capacities but not arc costs. Combined, these two problems give the basic constructs of network problems (Ahuja, et al., 1993).

The statement of the maximum flow problem is quite simple. If, in a capacitated network, it is desired to send as much flow as possible between a *source* node and a *sink* node without exceeding the arc capacity of any arc, this can be formulated as a maximum flow problem. The crux of this problem was solved in a journal article by Ford and Fulkerson (1956). The basic theorems and concepts necessary to solve these special problems were defined in this landmark paper. The basics of these concepts are defined and explained below.

When working with network flow algorithms, measure flow can sometimes be more conveniently measured in incremental flow about some given feasible solution, rather than in absolute terms. This incremental flow is generally the solution to the network at some intermediate point in an algorithm. A *residual network* is basically a "remaining flow network" that carries the incremental flow. The advantage here is that residual networks are the same as the original network in that they give a one-to-one correspondence between feasible solutions to the two problems, and maintain the cost of the solutions. Residual networks are key to maximum flow algorithms. Given a flow x, the residual capacity r_{ij} of any arc (i, j) *in* the network is the maximum additional flow that can be sent from node i to node j using the arcs (i, j) and (j, i). If u_{ij} denotes the unused capacity of the arc, and x_{ij} denotes the current flow on arc (j, i), then $r_{ij} = u_{ij} - x_{ij} + x_{ji}$.

A *cut* is a partition of the node set N into two subsets S and $\overline{S} = N - S$. A cut is called an *s-t cut* if $s \in S$ and $t \in \overline{S}$. An arc (i, j) is called a *forward arc* of the cut if $i \in S$ and $j \in \overline{S}$, and a *backward arc* of the cut if $i \in \overline{S}$ and $j \in S$. A *minimum cut* is an *s-t* cut whose capacity is minimum among all *s-t* cuts. A *residual capacity of an s-t cut* is the

sum of the residual capacities of forward arcs in the cut (Ahuja, et al., 1993; Winston, 2004).

Five basic assumptions are necessary in order to solve maximum flow problems using the Ford-Fulkerson labeling method. (1) The network must be directed. (2) All capacities must be nonnegative integers. (3) The network cannot contain a directed path from node *s* to node *t* that has only infinite capacity arcs. (4) Whenever an arc (*i, j*) belongs to the set of arcs in the network, then (*j*, *i*) also belongs to that set. Finally, (5) the network cannot contain parallel arcs (two or more arcs with the same tail and head nodes) (Ahuja, et al., 1993).

There are a number of important properties and theorems that ultimately lead to a labeling algorithm and validation of this algorithm for maximum flow problems. Note that this algorithm is a label correcting algorithm rather than a label setting algorithm, because the final label for any node or arc is not given until the last iteration. This is because any given node is looked at only once in a label setting algorithm, while nodes may have to be revisited multiple times in a label correcting algorithm. Therefore, the final nodes are not known in a label correcting algorithm until the entire process is complete. The complexity of the algorithm is described at the end of this subsection.

First, the value of any flow is less than or equal to the capacity of any cut in the network (Ahuja, et al., 1993; Winston, 2004). Further, for any flow *x* of value *v* in a network, any additional flow that can be sent from the source node to the sink node must be less than or equal to the residual capacity of any *s*-*t* cut.

So, given a maximum flow problem, how can a feasible, but non-optimal flow be modified to obtain a new feasible flow that has a larger flow from the source to the sink?

And, given a feasible flow, how does one know whether it is optimal? First, if the flow through (i, j) is below the capacity of the arc (i, j), then the flow through arc (i, j) can be increased. Let I represent arcs with this property. If the flow in arc (i, j) is positive, then the flow in arc (i, j) can be reduced. Denote this set of arcs as R. Assume that at least in simple cases, an initial feasible solution can be found by setting the flow in each arc equal to zero. The labeling algorithm follows. It is an attempt to label a feasible flow from the source node to the sink node.

Step 1: Label the source node.

Step 2: Label nodes and arcs according to the following rule: If node x is labeled and node y is unlabeled and arc $(x, y) \in I$, then label node y and arc (x, y). Note that in this case (x, y) is a forward arc. If node y is not labeled, node x is labeled and arc $(y, x) \in R$. Label node y and arc (y, x). Note that in this case, (y, x) is a backward arc.

Step 3: Continue is this way until either the sink node has been labeled or until no more vertices can be labeled.

At the end of this process there will be a chain of labeled arcs, denoted C. C can be adjusted to keep the flow feasible and to increase the total flow. Either C consists of only forward arcs, or C consists of both forward and backward arcs. The assumptions do not permit C to consist entirely of backward arcs. Recall that for each forward arc, r_{ij} is the amount by which the flow in arc (x, y) can be increased without violating its capacity constraint. Then, if k is the minimum amount of r_{ij} over C, then all forward arcs can be increased by k without violating any capacity constraints. In the case where C contains only forward arcs, this is guaranteed to increase the total flow across the network. If C contains both forward and backward arcs, then the flows of some arcs can be increased,

and some can be reduced. Define k_1 as the minimum over *R* of the amounts by which all arcs can be reduced, and define k_2 as the minimum over *I* of the amounts by which all arcs can be increased. To increase the total flow of the network, decrease the flow of all backward arcs in *C* by $\min(k_1, k_2)$, and increase the flow of all forward arcs in *C* by $\min(k_1, k_2)$.

Now the question of optimality arises again. First, the capacity of a cut is the sum of the capacities of the arcs in the cut, and the flow from the source node to the sink node is less than or equal to the capacity of any cut. Further, if the sink node cannot be labeled using the labeling algorithm, then the capacity of a cut is the current flow from the source node to the sink node. Therefore, if the sink node cannot be labeled, then the current feasible flow is the maximum flow. If the sink node is labeled, then adjust the feasible flow and increase the flow from the source to the sink. Go back and try to label the sink again, and continue in this way until an optimal solution is reached (Ahuja, et al., 1993; Winston, 2004). This can be formally stated with well-known theorems. (1) The maximum value of the flow from a source node *s* to a sink node *t* in a capacitated network equals the minimum capacity among all *s*-*t* cuts in the network. Also, it is of note that the labeling algorithm solves the maximum flow problem, under the assumptions stated, in O(*nmU*) time. This occurs because at worst the search method examines each arc only once, making the algorithm's complexity O(*m*) times the number of augmentations. If the capacities of all arcs are integral and bounded by some large finite number *U*, then the maximum capacity of the cut $(s, N - \{s\})$ is at most *nU*. This means that the maximum flow is bounded by *nU*. However, the labeling algorithm increases the value of the flow by at least one unit in any augmentation. Therefore, it will

yield an optimal solution within *nU* augmentations, and O(*nmU*) time (Ahuja, et al., 1993).

This method is best illustrated by an example.

Example 3. (Winston, 2004, pg. 429, #1)

Find the maximum flow from the source to the sink node in the network shown in Figure 3 below.

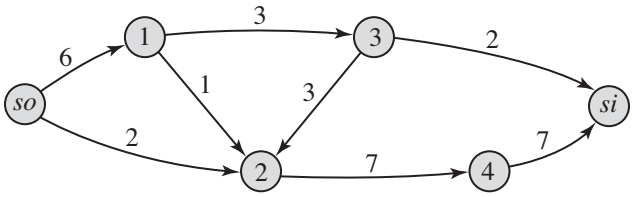


Figure 3. Network for Example 2
(Adapted from Winston, 2004)

Solution:

Maximize $z = x_0$ (the flow)

Subject to

$x_{so,1} \le 6,\ x_{so,2} \le 2,\ x_{12} \le 1,\ x_{32} \le 3,\ x_{13} \le 3,\ x_{3,si} \le 2,\ x_{24} \le 7,\ x_{4,si} \le 7$

$x_0 = x_{so,1} + x_{so,2}$ (source node)
$x_{so,1} = x_{13} + x_{12}$ (Node 1)
$x_{12} + x_{32} + x_{so,2} = x_{24}$ (Node 2)
$x_{13} = x_{32} + x_{3,si}$ (Node 3)
$x_{24} = x_{4,si}$ (Node 4)
$x_{3,si} + x_{4,si} = x_0$ (Sink node)

All variables are also ≥ 0. Note that these constraints are called "flow conservation constraints."

Let the initial flow of each arc be 0. This provides an initial feasible solution. Label the sink via a path of forward arcs (so, 1) – (1, 3) – (3, 2) – (2, 4), (4, si). Now,

since minimum(6, 3, 3, 7, 7) = 3, increase the flow from each of these feasible arcs by 3. This gives the following new feasible flow:

Arc	Flow
so-1	3
so-2	0
1-3	3
1-2	0
2-4	3
3-si	0
3-2	3
4-si	3

Flow to sink: 3

Now label the sink by (so-2) – (2-4) – (4-si). Each arc is a forward arc and the flow can be increased by min(2, 7, 7) = 2. This gives the following flow:

Arc	Flow
so-1	3
so-2	2
1-3	3
1-2	0
2-4	5
3-si	0
3-2	3
4-si	5

Flow to sink: 5

Now label the sink node by (so-1) – (1, 2) – (3, 2) – (3, si). All arcs on the path are forward arcs except for (3, 2). (3, 2) is a backward arc. Therefore, the flow of all the forward arcs can be increased by min(6, 1, 2) = 1, and the backward arc (3, 2) can be decreased by 1. This gives the following feasible flow:

Arc	Flow
so-1	4
so-2	2
1-3	3
1-2	1
2-4	5
3-si	1
3-2	2
4-si	5

Flow to sink: 6

Now the sink cannot be labeled, so this is the maximum flow. The maximum flow is 6 units. The minimum cut is obtained from (3, 2, 4, si). This cut consists of arcs (1, 3) – (1, 2) – (so, 2), and has capacity of $3+1+2=6$ units of maximum flow.

One further note should be made about maximum flow problems. Finding a minimum cut in a network is equivalent to finding the maximum flow. This is analogous to the duality relationship in linear programming. Therefore, the maximum flow algorithm is often called the max-flow min-cut algorithm. There are a number of relationships between these two problems, but they are beyond the scope of this paper (Ahuja, et al., 1993; Winston, 2004).

Critical Path Method (CPM)

Sometimes network models are used to schedule large, complex projects. If there are several tasks in a project, and the length required for each task is known, then the

critical path method can be used to determine the minimum total length of time required to complete the project. If any task is delayed then the project duration increases. This method can also be used to determine how long each task can be delayed without delaying the project's completion time. If the length of time required for each activity is not known, then the *Program Evaluation and Review Technique* can be used to estimate the probability that the project will be completed by a given time (deadline). These two methods are used in applications such as scheduling construction projects like office buildings and highway projects, scheduling the movement of large companies, developing a launching procedure for space flights, designing and marketing new products, and even completing corporate mergers. This section will cover the critical path method, but will only touch on PERT (Ahuja, et al., 1993; Winston, 2004).

To apply either CPM or PERT, it is necessary to have a list of tasks that make up the project. The entire project is considered complete when all the tasks have been completed. For each task there exist a subset of tasks, called *predecessors* of the task that must be completed before the main task begins. Likewise, *successors* of a task must be completed after a task. In a graphical network, directed arcs represent tasks, and nodes are used to denote the completion of a set of activities. The precedence relationships are also shown on the network. The nodes are often called *events*, and this type of network is called an *activity on arc* (AOA) network. For example, in Figure 4, node 3 shows the event that tasks A and B must be completed before task C begins.

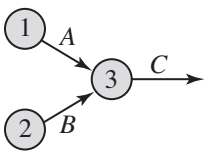


Figure 4. Network illustrating precedent events
(Adapted from Winston, 2004)

Five basic rules govern AOA networks: (1) Node 1 denotes the beginning of the project. An arc should lead from node 1 to each activity that has no predecessors. (2) A node (the *finish* node) should be included in the network. This node denotes the completion of the project. (3) The nodes in the network should be numbered so that the node denoting the completion of a task always has a larger number than the node denoting the beginning of a task, exactly like topological sorting. This need not be a unique numbering scheme. (4) One task should not be denoted by more than one arc in the network. (5) Two nodes can be connected by at most one arc. Sometimes *dummy activities* are necessary in order to avoid violating rules (4) and (5). These dummy activities take no time, and are analogous to artificial variables in a linear programming problem.

The earliest time at which the event corresponding to node *i* can occur is called the *early event time,* $ET(i)$. The latest time at which the event corresponding to node *i* can occur without delaying the completing of the project is called the *late event time*, $LT(i)$. $ET(i)$ can be computed by calculating the earliest time that each node can be completed, given all other constraints. It can be reduced to three basic steps: (1) Find each event prior to node *i* that is connected to node *i* by an arc. These are the predecessor events of node *i*. (2) Add the duration of the task connecting the immediate predecessor to node *i* to the *ET*. (3) $ET(i)$ is the maximum of the sums computed in step 2.

Likewise, $LT(i)$ can be computed using three similar steps. (1) Find each node that occurs after node *i* that is connected to node *i* by an arc. These are the immediate successor events of node *i*. (2) Subtract the duration of the task joining the successor from

the *LT* for each immediate successor to node *i*. (3) *LT*(*i*) is the minimum of the differences from step 2.

Define *total float* as follows: Consider an arbitrary arc representing a task (*i*, *j*), denoted *TF*(*i*, *j*), of the task denoted by (*i*, *j*). Assuming no other tasks are delayed, the total float is the amount of time by which the starting time of the task (*i*, *j*) could be delayed past its earliest possible starting time without delaying the completion of the project. Any task with zero total float is a *critical activity* since it cannot be delayed without affecting the project completion time. A path from node 1 to the finish node that consists of entirely critical activities is a *critical path*. *Free float* is a measure of the flexibility in the duration of a task. The free float of the activity corresponding to arc (*i*, *j*) is the amount of time by which the starting time of the task corresponding to (*i*, *j*) can be delayed without delaying the start of any later task past its earliest starting time (Ahuja, et al., 1993; Winston, 2004).

Linear programming can be used to find a critical path in a project network by defining x_j as the time that the event corresponding to node *j* occurs. For each task (*i*, *j*), node *i* must occur before node *j* occurs. Therefore, for each arc (*i*, *j*), a linear inequality constraint can be placed on the arc. The goal is to minimize the difference $x_F - x_1$, where *F* is the final node in the network. Once defined, many of these problems can be easily solved with existing computer packages time. In many problems, a project must be completed in a time that is less than the length of its critical path. This can often be accomplished by allocating additional resources to certain tasks and reformulating the problem, which is equivalent to removing some predecessor constraint arcs (Ahuja, et al., 1993; Winston, 2004).

CPM makes the assumption that the time requirements for each task are precisely known. In reality this is often not the case. PERT models the time requirement for each task as a random variable and attempts to estimate the time requirements for each task. However, it must be noted that PERT makes some strong assumptions that may not always be met. First, PERT assumes that task durations are independent from each other. This may not always be true, as one task may depend upon its predecessor(s) and/or successor(s). PERT assumes that task durations follow a Beta distribution. This may also fail to occur in real problems. Finally, PERT assumes that the critical path found by CPM will always be the critical path for the project. This may not be justified in some real problems. This last assumption is usually the most potentially serious flaw of PERT, since delays or ahead of schedule completions in tasks can change the critical path. A simulation such as a Monte Carlo simulation is usually the safest way to estimate task durations when they are unknown and to decide whether a given task is critical (Ahuja, et al., 1993; Winston, 2004).

Minimum-Cost Network Flow Problems

The transportation, assignment, transshipment, shortest-path, maximum flow, and CPM problems are special cases of the minimum-cost network flow problem. Any of these problems can be solved by the Network Simplex Method, which will be covered in detail shortly. A minimum-cost network flow problem can be defined by

$$\min \sum_{\text{all arcs}} c_{ij} x_{ij}$$

subject to $\sum_j x_{ij} - \sum_k x_{ki} = b_i$ for each node i in the network, and

$$L_{ij} \le x_{ij} \le U_{ij},$$

where

x_{ij} is the number of units of flow sent from node i to node j through the arc (i, j)

b_i is the net supply, or outflow minus inflow at node i

c_{ij} is the cost of transporting one unit of flow from node i to node j through arc (i, j)

L_{ij} is the lower bound of flow through the arc (i, j). L_{ij} is allowed to be zero if there is no lower bound, and

U_{ij} is the upper bound of flow through the arc (i, j). U_{ij} is allowed to be ∞ if there is no upper bound. The constraints of the form $\sum_j x_{ij} - \sum_k x_{ki} = b_i$ state that the flow out of node i must equal b_i, and are called *flow balance equations*. The constraints $L_{ij} \le x_{ij} \le U_{ij}$ state that the flow through each arc is in compliance with the arc capacity restrictions. Using these definitions, transportation problems, maximum-flow problems, and all of the other special cases can be written as minimum-cost network flow problems. Once formulated in this way, these problems can be solved using the Network Simplex Method. However, Minimum Spanning Tree Problems will be covered before ending with the description and an example of the Network Simplex Method (Ahuja, et al., 1993; Winston, 2004).

Minimum Spanning Tree Problems

A *spanning tree* is a connected acyclic subgraph that spans all the nodes. Every spanning tree of a network has $n - 1$ arcs. A *minimum spanning tree* is a spanning tree that has the smallest total cost of its member arcs. This cost is measured as the sum of the costs of the arcs of the spanning tree. Spanning trees are central to network flows in that they are related to shortest path problems, minimum cost flow problems, and maximum flow problems. In fact, minimum cost flow problems always have spanning tree solutions, and spanning trees can be constructed that are rooted at the source node of a shortest path problem. This is key to solving these problems. However, there are key differences between shortest path problems and minimum spanning tree problems. Minimum spanning tree problems deal with undirected arcs, rather than the directed arcs in shortest path problems. Second, objective functions for minimum spanning tree problems are different from the objective functions in shortest path problems. For minimal spanning trees, the cost of each arc is counted exactly once, while arcs can be counted several times in shortest path problems (Ahuja, et al., 1993; Winston, 2004).

Minimum spanning tree problems occur in many applications. Some of these include the design of physical systems, optimal message passing in covert situations, problems that involve the reduction of data storage, cluster analysis, and problems in computer networking. As with other network problems, the optimality conditions play a key role in the development of algorithms to solve the problems. There are several key optimality conditions connected with minimum spanning tree problems. To begin, for every nontree arc (k, l), there is a unique path from node k to node l in the spanning tree. The arc (k, l) and this path define a cycle. Also, any tree arc (i, j) that is deleted from a

spanning tree divides the node set *N* into two subsets. The arcs whose endpoints belong to the different subsets are a cut. A spanning tree is a minimum spanning tree if and only if for every tree arc that is a part of the spanning tree, $c_{ij} \leq c_{kl}$ for every arc (*k*, *l*) that is part of the cut formed by deleting the arc (*i*, *j*) from the spanning tree. This optimality condition has to do with cuts of a spanning tree. The next optimality condition deals with paths of a spanning tree. A spanning tree is a minimum spanning tree if and only if for every nontree arc (*k*, *l*), $c_{ij} \leq c_{kl}$ for every arc (*i*, *j*) that is in the path of the spanning tree that connects nodes *k* and *l.* These optimality conditions lead directly to algorithms to solve minimum spanning tree problems. Perhaps the most famous of these is Kruskal's algorithm and its variants (Ahuja, et al., 1993; Winston, 2004).

Kruskal's algorithm starts with any spanning tree and tests the path optimality conditions. If the spanning tree satisfies the condition, it is optimal. If it doesn't, then it is clear that $c_{ij} > c_{kl}$ for at least some nontree arc (*k*, *l*) and some tree arc (*i*, *j*) that is in the unique path in the spanning tree that connects nodes *k* and *l*. If this is the case, add the arc (*k*, *l*) to the spanning tree to replace the arc (*i*, *j*). This will yield a spanning tree with a lower cost. Repeating this step will yield a minimum spanning tree within a finite number of iterations. However, the simplicity of the algorithm notwithstanding, its running time is not polynomially bounded in the size of the problem data (Ahuja, et al., 1993; Winston, 2004).

Fortunately, an improved version of Kruskal's algorithm solves the minimum spanning tree problem in $O(m + n \log n)$ time plus time required for sorting the arcs, where *n* is the number of nodes, and *m* is the number of arcs in the network. This algorithm requires that all the arcs first be sorting in nondecreasing order of their costs

(Ahuja, et al., 1993). There is also an algorithm that is derived directly from Kruskal's algorithm that is called a *greedy* algorithm because the algorithm chooses the shortest arc that can be used to expand a connected set of nodes at each step. One such algorithm proceeds as follows:

Step 1: Beginning at any node i, join node i to the node in the network (call it j) that is closest to node i. The two nodes i and j now form a connected set of nodes and the arc (i, j) will be in the minimum spanning tree. The remaining nodes are called the *unconnected* set of nodes.

Step 2: Choose a member of the unconnected set of nodes (call it p) that is closest to some node in the connected set. Let q be the node in the connected set that is closest to p. Then the arc (q, p) will be in the minimum spanning tree. Update the connected and unconnected sets, and eliminate node p from the unconnected set.

Step 3: Repeat this process until a minimum spanning tree is found. Ties for the closest node and arc can be broken arbitrarily. This will ultimately yield a set of p elements (Winston, 2004).

Minimum spanning tree problems can also be formulated as linear programming problems. The formulation is as follows:

$$\min \sum_{\text{all arcs}} c_{ij} x_{ij}$$

subject to:

$$\sum_{\text{all arcs}} x_{ij} = n - 1$$

$$\sum_{\text{all arcs in } S} x_{ij} \leq |S| - 1 \quad \text{(for any set } S \text{ of nodes)}$$

$$x_{ij} \geq 0 \text{ and are integers}$$

Note that the linear programming formulation of the minimum spanning tree problem requires an exponential number of constraints, but it still can be solved very efficiently using Kruskal's algorithm (Ahuja, et al., 1993).

Network Simplex Method

The Network Simplex method is one of the most elegant of network algorithms. Recall that the task of finding a basic feasible solution, pivoting, and computing the coefficient of a nonbasic variable in row 0 were simplified in the transportation Simplex Method shown in the breadth section. Similar simplifications when solving a minimum cost network flow problem (MCNFP) using the network Simplex method (Ahuja, et al., 1993; Bazaraa, et al., 2005; Winston, 2004).

To determine whether a feasible solution to an MCNFP is a basic feasible solution, note that an MCNFP contains three basic types of variables. If the problem is not degenerate, every basic variable x_{ij} will be between the lower bound L_{ij} and the upper bound U_{ij}. Degeneracy allows the possibility that a basic variable x_{ij} to equal the arc (i, j)'s upper or lower bound. The other two variable types are nonbasic variables x_{ij} that are equal to the arc (i, j)'s upper bound U_{ij}, and nonbasic variables x_{ij} that are equal to the arc (i, j)'s lower bound L_{ij} (Ahuja, et al., 1993; Bazaraa, et al., 2005; Winston, 2004).

Consider an MCNFP with n conservation of flow constraints. Now, recall that in the transportation problem that any solution that satisfies $n - 1$ of the constraints will automatically satisfy the last constraint. This means the one constraint can be dropped, just as in the transportation problem, and there are $n - 1$ basic variables. For a set of $n - 1$

variables or arcs, the set will give a basic feasible solution if and only if the arcs corresponding to the basic variables form a spanning tree for the network. This means that they connect all the nodes of the graph and are acyclic. For small problems, such a basic feasible solution can often be found by trial and error. An initial spanning tree structure can also be found by establishing a feasible flow in the network and then solving a maximum flow problem (Ahuja, et al., 1993; Bazaraa, et al., 2005; Winston, 2004).

The next problem is to determine the objective function for the MCNFP. This is row 0 in the Simplex algorithm. If the conservation of flow constraint for node 1 is dropped, then for a given basic feasible solution, define $c_{BV}B^{-1} = [y_2, y_3, \cdots y_n]$. It is easy to see that this is analogous to the other variants of the Simplex method. Now every x_{ij} will have a coefficient of +1 in the node *i* flow constraint and a coefficient of –1 in the node *j* constraint. Letting $y_1 = 0$, the coefficients in row 0 of the tableau can be written as $\bar{c}_{ij} = y_i - y_j - c_{ij}$. Since $\bar{c}_{ij} = 0$ for every basic variable, the y_i can be found by solving the system of equations

$$\begin{aligned} y_1 &= 0 \\ y_i - y_j &= c_{ij} \end{aligned}$$

for each basic variable. The y_i's that correspond to a basic feasible solution are called the *simplex multipliers* for the basic feasible solution (Ahuja, et al., 1993; Bazaraa, et al., 2005; Winston, 2004).

Now that a basic feasible solution has been found, the issue of optimality arises. If a basic feasible solution is optimal then it must be possible to improve the objective function by changing the value of a nonbasic variable. Also, $\bar{c}_{ij} \leq 0$ if and only if

increasing x_{ij} cannot decrease the objective function, and $\bar{c}_{ij} \geq 0$ if and only if decreasing x_{ij} cannot decrease the objective function. Therefore, a basic feasible solution is optimal if and only if the following two conditions hold:

1. If a variable x_{ij} is equal to the lower bound L_{ij}, then increasing x_{ij} cannot result in a decrease in the objective function. So, if $x_{ij} = L_{ij}$ and the basic feasible solution is optimal, then it must be true that $\bar{c}_{ij} \leq 0$.
2. If a variable x_{ij} is equal to the upper bound U_{ij}, then decreasing x_{ij} cannot result in a decrease in the objective function. So, if $x_{ij} = U_{ij}$ and the basic feasible solution is optimal, then it must be true that $\bar{c}_{ij} \geq 0$.

As in other variants of the Simplex method, if the current basic feasible solution is not optimal, then it can be improved by pivoting. There are actually multiple potential pivot rules that are used in the network Simplex method, but only one will be described here. If the current basic feasible solution is not optimal, then a nonbasic variable that violates either condition will enter the basis. If a particular variable x_{ij} is to enter the basis, then it will create a unique cycle when it is added to the spanning tree of the current basic feasible solution. The new values of all of the variables in the cycle are determined using the conservation of flow constraints. Then the variable that leaves the basis will be the variable that reaches its upper or lower bound first as the value of the variable entering the basis is changed. Finally, the new basic feasible solution is found by changing the flows of the arcs in the cycle found during the pivot (Ahuja, et al., 1993; Bazaraa, et al., 2005; Winston, 2004).

Pivoting using the network Simplex method moves from one feasible spanning tree structure to another until one is found that satisfies the optimality conditions. So long as each pivot is nondegenerate, the algorithm will terminate in finite time. Degenerate pivots potentially result in a failure of the algorithm to terminate, but a special alternative implementation of the algorithm called *strongly feasible spanning tree implementation* will guarantee that the algorithm will terminate in finite time even with degenerate problems (Ahuja, et al., 1993). Example 4 illustrates a simple case of the network Simplex method.

Example 4. (Winston, 2004, p. 466, #3)

Find the optimal solution to the MCNFP in Figure 5 using the basic feasible solution in Figure 6 as a starting basis.

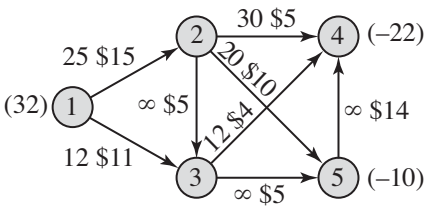


Figure 5. Network for Example 4
(Adapted from Winston, 2004)

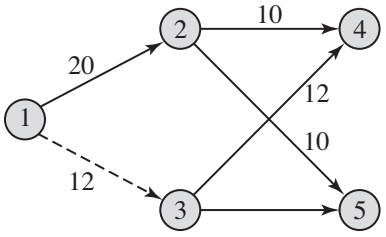


Figure 6. Starting Basic Feasible Solution for Example 4
(Adapted from Winston, 2004)

Solution:

First, find the y_i values. The system of equations is

$$\begin{aligned} y_1 &= 0 \\ y_1 - y_2 &= 15 \\ y_2 - y_4 &= 5 \\ y_2 - y_5 &= 10 \\ y_3 - y_4 &= 4 \end{aligned}$$

Solving this system of equations gives $y_1 = 0,\ y_2 = -15,\ y_3 = -16,\ y_4 = -20,\ y_5 = -25$.

Now calculate the coefficients for row 0 for all the nonbasic variables. Use the formula $\bar{c}_{ij} = y_i - y_j - c_{ij}$.

Since the cost from node 1 to node 3 is \$11, $\bar{c}_{13} = 0 - (-16) - 11 = 5$.

Since the cost from node 2 to node 3 is \$5, $\bar{c}_{23} = -15 - (-16) - 5 = -4$.

Since the cost from node 3 to node 5 is \$5, $\bar{c}_{35} = -16 - (-25) - 5 = 4$.

Since the cost from node 4 to node 5 is \$14, $\bar{c}_{45} = -20 - (-25) - 14 = -9$.

All of the $\bar{c}$ satisfy the optimality conditions except for $\bar{c}_{35}$. Therefore, x_{35} needs to enter the basis. The pivot is shown in Figure 7 below.

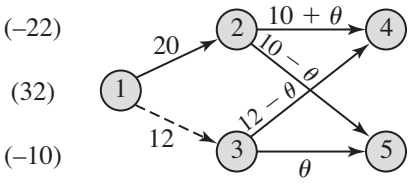


Figure 7. Pivot Process for Example 4

Now, observe that x_{35} needs to be increased as much as possible. It can be increased until a basic variable first reaches its upper or lower bound. From Figure 7, $\theta = 10$, so x_{25} leaves the basis. The new basic feasible solution is shown in Figure 8.

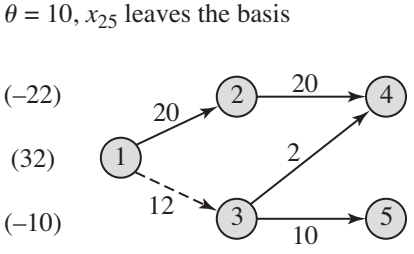


Figure 8. New Basic Feasible Solution After x_{25} Leaves Basis

Now the new y_i values and $\bar{c}_{ij}$ are computed and tested for optimality.

The new system of equations is

$$\begin{aligned} y_1 &= 0 \\ y_1 - y_2 &= 15 \\ y_2 - y_4 &= 5 \\ y_3 - y_4 &= 4 \\ y_3 - y_5 &= 5 \end{aligned}$$

The solution to this system gives

$$y_1 = 0,\ y_2 = -15,\ y_3 = -16,\ y_4 = -20,\ y_5 = -21$$

Now calculate the coefficients for row 0 for all the nonbasic variables. Use the formula

$$\bar{c}_{ij} = y_i - y_j - c_{ij}.$$

$$\bar{c}_{13} = 0 - (-16) - 11 = 5$$

$$\bar{c}_{23} = -15 - (-16) - 5 = -4$$

$$\bar{c}_{25} = -15 - (-21) - 10 = -4$$

$$\bar{c}_{45} = -20 - (-21) - 14 = -13$$

All of the $\bar{c}$'s satisfy the optimality conditions, so an optimal solution to this MCNFP is $x_{12} = 20,\ x_{24} = 20,\ x_{34} = 2,\ x_{35} = 10$. These are the basic variables. The nonbasic variables are $x_{13} = 12,\ x_{23} = 0,\ x_{25} = 0,\ x_{45} = 0$. Clearly, x_{13} is at its upper bound, and

the remaining variables are at their lower bounds. The optimal objective function value is $z = 20(15) + 12(11) + 20(5) + 2(4) + 10(5) = \$590$.

Conclusion

The field of network flows is staggeringly large. The work shown in this paper is a mere introduction to the almost limitless applications and solution methods in this field. Research continues and is prolific as new problems in genetics and other fields give rise to network problems with astounding numbers of variables and constraints. This field is in no way mature, and the future will undoubtedly result in new problems and even more efficient algorithms to solve these important problems. Their solutions will improve financial and time efficiency in numerous fields.

References

Ahuja, R., Cunha, C., & Sahin, G. (2005). Network models in railroad planning and scheduling, *Informs, Tutorials in Operations Research,* 54–101.

Ahuja, R., Magnanti, T., & Orlin, J. (1993). *Network Flows,* Upper Saddle River, New Jersey: Prentice Hall Publishers.

Adelson-Velsky, G. & Levner, E. (2002). Project scheduling in and-or graphs: A generalization of Dijkstra's Algorithm, *Mathematics of Operations Research, 27(3),* 504–517.

Bazaraa, M., Jarvis, J., & Sherali, H. (2005). *Linear Programming and Network Flows.* New York: John Wiley & Sons.

Benjamin, J. (1989). An analysis of inventory and transportation costs in a constrained network, *Transportation Science, 23(3),* 177–183.

Bjorndal, E., Hamers, H., & Koster, M. (2004). Cost allocation in a bank ATM network, *Mathematical Methods of Operations Research, 59,* 405–418.

Burdett, R. & Kozan, E. (2004). The assignment problem of individual renewable resources in scheduling, *Asia-Pacific Journal of Operational Research, 21(3),* 355–377.

Chiou, S. W. (2005). Joint optimization for area control and network flow. *Computers & Operations Research, 32(11),* 2821–2841.

Ding, H., Lim, A., Rodrigues, B., & Zhu, Y. (2005). The over-constrained airport gate assignment problem, *Computers & Operations Research, 32(7),* 1867–1880.

Doulliez, P. & Rao, M. (1975). Optimal network capacity planning: A shortest-path scheme. *Operations Research, 23(4),* 810–818.

Ford, L. R., & Fulkerson, D. R. (1956). Maximal flow through a network. *Canadian Journal of Mathematics, 8,* 399–404.

Ireland, P., Case, R., Fallis, J., Van Dyke, C., Kuehn, J., & Meketon, M. (2004). The Canadian Pacific Railway transforms operations by using models to develop its operating plans, *Informs, 34(1),* 5–14.

Nauss, R. M. (2004). The elastic generalized assignment problem. *Journal of the Operational Research Society, 55(12),* 1333–1341.

Rhee, S. (2005). Sharing-group allocation problems. *Economics Letters, 86(1),* 51–56.

Schorr, R. & Rieger, H. (2003). Universal properties of shortest paths in isotropically correlated random potentials. *The European Physical Journal B, 33,* 346–354.

Toktas, B., Yen, J., & Zabinsky, Z. (2006). Addressing capacity uncertainty in resource-constrained assignment problems. *Computers & Operations Research, 33(3),* 724–745.

Yildrim, M.B. & Hearn, D. W. (2005). The first best toll pricing framework for variable demand traffic assignment problems. *Transportation Research, 39(8),* 659–678.

Winston, W. L. (2004). *Operations Research: Applications and Algorithms*. Belmont, CA: Brooks Cole Publishing Company.

APPLICATION DEMONSTRATION

APPLICATION TABLE OF CONTENTS

APPLICATION DEMONSTRATION

Introduction

In the breadth section of this paper, introductions to linear programming, network algorithms, and nonlinear programming were discussed. The depth section included a broader discussion of network algorithm and included a review of 15 journal articles on the subject. To follow up this work with an application, a seminar was conducted for approximately 60 high school students in Redwood City, California for the purpose of introducing and promoting careers in operations research. In this section, the details and results of that seminar are reviewed.

On January 24, 2006, approximately 60 high school students attended a seminar that introduced and promoted operations research as a possible career choice. The seminar began at approximately 8:30 a.m. and lasted until 11:15 a.m., though not all students were there for the entire sessions. The students were almost exclusively juniors and seniors in high school, and the majority were seniors. The majority of students were taking calculus at the time of the seminar, though some interested precalculus students from another teacher's class were allowed to attend.

The seminar was conducted at Sequoia High School in Redwood City, California. This particular school was chosen because it is the only school in the San Francisco Bay Area that has an International Baccalaureate (IB) Program. The IB program is an extremely advanced program that requires that students take college level courses in math, science, language, writing, and history during their junior and senior years of high school. Students who elect to complete the program are then given tests during their

junior and senior years in each of these subjects as well as completing several other requirements. Many top colleges offer students guaranteed or preferred admission and up to a full year of college credit to IB graduates. Many of the students who attended the seminar were IB students, and therefore it is highly likely that the majority are college and graduate school bound after high school.

Seminar Contents

A general outline of the topics discussed in the seminar is shown in the appendix. The seminar began with a very brief introduction to the origins of the disciplines of mathematics and science. Specifically, these disciplines all have roots in philosophy and then branched out into studies of narrower subjects. While many fields of mathematics and science are continuing to branch out, (e.g., chemistry now includes genetics, organic chemistry, etc.), recent years have seen other fields combining to make new disciplines. Examples of this include the relatively new fields of social psychology and operations research. Operations research was described to students as a combination of mathematics, business, and systems engineering. Systems engineering was also defined for students. The definition of operations research was further broken down into the two major areas of deterministic and stochastic techniques. These were also defined, as was the general definition of decision science. Students were also given descriptions of the basic topics of mathematics that are generally used in operations research.

Since the heart of operations research is generally agreed to be deterministic optimization techniques, most of the time spent in the seminar was on this area. Several applications of deterministic optimization techniques were described. These included

maximization of profits, minimization of costs, transportation problems, transshipment problems, and assignment problems. Railroad problems were also described as a specific example, though this was not done in a lot of detail due to time and student interest constraints. A graphical linear programming problem in two dimensions was presented in detail in order to illustrate the graphical representation of an optimization problem. Students participated in the presentation of this example by answering questions throughout. Most had seen similar problems in their Algebra 2 courses one or two years before. An explanation of extreme points and feasible region were given, and the idea of searching algorithms was briefly discussed using the simple graphical example.

After the graphical problem was discussed, the mathematics involved in solving linear programming problems with a larger number of variables was briefly discussed. Students were told that some linear programming problems that researchers tackle today have billions or even trillions of variables and constraints. The concept of networks was also briefly discussed, and the concept of algorithms introduced. Computer software was discussed as a way of solving large and complex problems, but it was also stressed that it is necessary to know the theory and assumptions behind a problem even when software tools are available.

Mathematica® was demonstrated to the class to show that it can solve mathematics problems, including calculus problems, algebraically rather than numerically. The linear two dimensional graphical linear programming problem that was solved on the board was also solved using *Mathematica*® in order to show the ease with which software tools can solve tedious problems. In addition, *Excel's "Solver*" tool was discussed and demonstrated to the class. A word problem with four variables and five

constraints was discussed and solved using the *Solver* tool. Students were also given a basic tutorial in how to use *Solver*.

The classroom teacher pointed out that an article had just come out in the latest version of *Business Week* that discussed careers in mathematics and the new demand for mathematicians in various fields. The existence of this article was used as a way to introduce various careers related to mathematics. It was specifically pointed out that people who work in mathematics don't sit and solve textbook problems all day, and that writing and general communication are extremely important skills that mathematicians and engineers need in order to be successful. Students were also told that perhaps the biggest problem that mathematicians and engineers face is properly defining the problems that they are to solve. Much of the job entails working with the client or other person in charge to understand or even help that person determine what the real problem definitions are.

Finally, students were given an opportunity to ask questions about careers in math or suggested college coursework. Since most students did not yet know what they wanted to do for their careers, college coursework was the main topic of discussion. Students were told that taking a few extra math courses in college would be a wise idea, even if their desired area of studies turns out to be something seemingly far from mathematics. . For example, psychology researchers use experimental design and statistics regularly, and lack of skills in this area can undermine all of their research.

Students also specifically asked which calculus class they should start with in college if they pass the advanced placement test in math with a score or 4 (very good) or 5 (superior). The answer depends upon the university and whether it is on the quarter or

semester system. These questions about college math courses were answered for students. Some students also had questions about which math courses to take to meet a specific career goal. These questions were answered individually during the approximately 15 minutes that were available after the seminar officially ended.

There was not time to cover all of the examples in the outline. The Simplex Method could not be covered with the entire group due to time constraints, but that problem and the word problem involving stocks was covered with two students who asked questions after the seminar about what a real problem looked like. Those students seemed to enjoy the problems.

Response

The group of students involved in the seminar by and large seemed interested in the material. None of the students had ever heard of operations research before the seminar, and most were surprised to hear of its existence after hearing what was involved in the discipline.

The group was especially interested in *Mathematica*®, and it seemed that most had never seen any sort of computer algebra system software. A show of hands indicated that only 3 students knew that *Solver* was a tool available on *Excel.* Most were surprised to find that *Excel* had such sophisticated mathematical tools. The students who were aware of some of the software tools available in *Excel* were the students who were already planning a career in mathematics. Two of these students were actually second year calculus students who were working on their own due to the lack of a course offering in second year calculus.

There was a large disparity between the interests of the juniors and the seniors in the seminar. The seniors were far more attentive and interested in the material. This was almost certainly due to the fact that the seniors were in the current position of selecting universities and majors, and therefore were closer to having to make a career decision than the juniors. However, the juniors were quite advanced in math in the sense that they were taking calculus so early, and therefore it was expected that there might be a higher proportion of them who were planning to go into careers in math. Either this was not the case, or the juniors had not yet made up their minds about their level of interest in math (or any other subject for that matter). Only three students indicated that they had made any kind of choice about their college major. Two of these were hoping to major in math and the other wanted to be a psychology major.

Fortunately, the message seems to have gotten through to all the students that math is becoming more important in today's society even than it has been in the past. Several students indicated that the seminar was "inspirational," "interesting," and "informative." All in all, the seminar seems to have been a success in introducing students to operations research and informing them about the many career opportunities for those with mathematical skills.

Conclusion

Dealing with high school students is a difficult endeavor, especially when the goal is to introduce something to them that is out of their area of immediate interest. However, the payoffs can be enormous if even one student out of the group reconsiders math as a possible career choice. Even if no change occurs in career choice, the knowledge that students gained from this seminar will hopefully follow them into whatever career they

choose. Operations research is used in so many industries that a well informed employee can have an impact of the optimal running of an operation even if they themselves are not doing the optimization work.

Appendix: Seminar Outline

I. What is Operations Research?

II. What kind of mathematics is involved in OR?

A. General—Calculus, Linear Algebra, etc.

B. Deterministic Processes

C. Stochastic Processes

III. The Heart of Operations Research….Optimization Problems

A. What are optimization problems?

B. Applications

What is a LP Problem?

Transportation, Assignment, and Transshipment Problems (Define)

Networks (Define)

IV. Example of a graphical LP problem: use example from breadth paper

Example 1. (Winston, 2004, p. 68, #9, modified)

Graphically determine two optimal solutions to the following LP:

$$\max z = 3x_1 + 5x_2$$

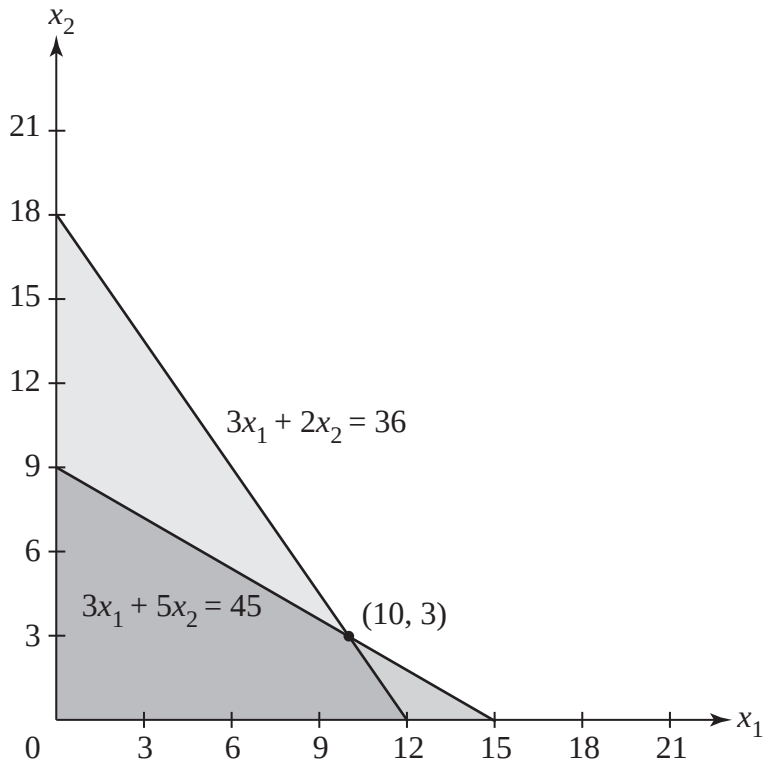
subject to:

$$3x_1 + 2x_2 \le 36$$
$$3x_1 + 5x_2 \le 45$$
$$x_1, x_2 \ge 0$$

Solution:

Graphically determine the optimal solutions to the linear programming problem by graphing the feasible region and identifying the extreme points. Note that it is well

known that the solution to a linear programming problem must be one of the extreme points of the feasible region of the problem (Bazaraa, et al., 2005; Winston, 2004).



From the graph, it is obvious that there are extreme points at (0, 0), (12, 0), and (0, 9). To find the fourth extreme point, solve the system of equations

$$\begin{cases} 3x_1 + 2x_2 = 36 \\ 3x_1 + 5x_2 = 45 \end{cases}$$

to determine the point where the lines intersect. The solution to this system is the point (10, 3). Finally, test each of the four extreme points to determine the maximum(s).

Test (0, 0): $3x_1 + 5x_2 = 0$

Test (12, 0): $36 + 0 = 36$

Test (0, 9): $0 + 45 = 45$

Test (10, 3): $30 + 15 = 45$

Two optimal solutions exist and are at (0, 9) and (10, 3). Each has a maximum value of 45.

Example 2. (Winston, 2004, p. 120, #45).

Currently we own 100 shares each of stocks 1 through 10. The original price we paid for these stocks, today's price, and the expected price in one year for each stock is shown in Table 1 below. We need money today and are going to sell some of the stocks. The tax rate on capital gains is 30%. If we sell 50 shares of stock 1, then we must pay tax of $(0.3) \cdot 50(30 - 20) = \$150$. We must also pay transaction costs of 1% on each transaction. Thus, our sale of 50 shares of stock 1 would incur transaction costs of $.01 \cdot 50 \cdot 30 = \$15$. After taxes and transaction costs, we must be left with $30,000 from our stock sales. Our goal is to maximize the expected (before-tax) value in one year of our remaining stock. What stocks should we sell? Assume it is all right to sell a fractional share of stock.

Table 1

Purchase, Current, and Expected Future Prices of Stocks Owned

		Price ($)		
Stock	Shares Owned	Purchase	Current	In One Year
1	100	20	30	36
2	100	25	34	39
3	100	30	43	42
4	100	35	47	45
5	100	40	49	51
6	100	45	53	55
7	100	50	60	63
8	100	55	62	64
9	100	60	64	66
10	100	65	66	70
Tax rate (%)	0.3			
Transaction cost (%)	0.01			

Solution:

Assume we sell x_1 shares of stock 1, x_2 shares of stock 2, and so on. Then we will have $100 - x_1$ shares of stock 1 left after the sale, $100 - x_2$ shares of stock 2 left after the sale, and so on. The data necessary to formulate the problem is given in Table 2 below.

Table 2
Taxes and Transaction Costs for Stocks from Table 1

Stock	Taxes	Transaction Costs
1	$.3(30-20)x_1$	$.01x_1(30)$
2	$.3(34-25)x_2$	$.01x_2(34)$
3	$.3(43-30)x_3$	$.01x_3(43)$
4	$.3(47-35)x_4$	$.01x_4(47)$
5	$.3(49-40)x_5$	$.01x_5(49)$
6	$.3(53-45)x_6$	$.01x_6(53)$
7	$.3(60-50)x_7$	$.01x_7(60)$
8	$.3(62-55)x_8$	$.01x_8(62)$
9	$.3(64-60)x_9$	$.01x_9(64)$
10	$.3(66-65)x_{10}$	$.01x_{10}(66)$

The linear programming problem is then written as follows:

Maximize:

$$36(100-x_1)+39(100-x_2)+42(100-x_3)+45(100-x_4)+51(100-x_5)$$
$$+55(100-x_6)+63(100-x_7)+64(100-x_8)+66(100-x_9)+70(100-x_{10})$$

Subject to:

$$(30x_1-3.3x_1)+(34x_2-3.04x_2)+(43x_3-4.33x_3)+(47x_4-4.07x_4)+(49x_5-3.19x_5)$$
$$+(53x_6-2.93x_6)+(60x_7-3.6x_7)+(62x_8-2.72x_8)+(64x_9-1.84x_9)$$
$$+(66x_{10}-0.96x_{10})=\$30{,}000$$

Discuss feasible region

V. Example of the Simplex Method – Set up and perform one pivot.

Example 3. (Bazaraa, et al., 2005, p. 140, #3.25)

$\max\ x_1 - 2x_2 + x_3$

subject to:

$$\begin{aligned} x_1 + 2x_2 + 3x_3 &\le 12 \\ 2x_1 + x_2 - x_3 &\le 6 \\ -x_1 + 3x_2 &\le 9 \\ x_1, x_2, x_3 &\ge 0 \end{aligned}$$

Solution.

Adding slack variables x_5 and x_6, the initial tableau is as follows:

Tableau 1:

	z	x_1	x_2	x_3	x_4	x_5	x_6	RHS
z	1	1	–2	1	0	0	0	0
x_4	0	1	2	3	1	0	0	12
x_5	0	**2**	1	–1	0	1	0	6
x_6	0	–1	3	0	0	0	1	9

Tableau 2:

	z	x_1	x_2	x_3	x_4	x_5	x_6	RHS
z	1	0	$-\frac{5}{2}$	$\frac{3}{2}$	0	$-\frac{1}{2}$	0	–3
x_4	0	0	$\frac{3}{2}$	$\mathbf{\frac{7}{2}}$	1	$\frac{1}{2}$	0	9
x_1	0	1	$\frac{1}{2}$	$-\frac{1}{2}$	0	$\frac{1}{2}$	0	3
x_6	0	0	$\frac{7}{2}$	$-\frac{1}{2}$	0	$\frac{1}{2}$	1	12

Tableau 3:

	z	x_1	x_2	x_3	x_4	x_5	x_6	RHS
z	1	0	$-\frac{22}{7}$	0	$-\frac{3}{7}$	$-\frac{5}{7}$	0	$-\frac{48}{7}$
x_3	0	0	$\frac{3}{7}$	1	$\frac{2}{7}$	$\frac{1}{7}$	0	$\frac{18}{7}$
x_1	0	1	$\frac{5}{7}$	0	$\frac{1}{7}$	$\frac{4}{7}$	0	$\frac{30}{7}$
x_6	0	0	$\frac{26}{7}$	0	$\frac{1}{7}$	$\frac{4}{7}$	1	$\frac{93}{7}$

The maximum value is $\frac{48}{7}$ when $x_1 = \frac{30}{7}, x_2 = 0, x_3 = \frac{18}{7}$.

VI. Computer example:

Minimize $z = 15x_1 + 10x_2 + 9x_3 + 7x_4$

subject to:

$$x_1 + x_2 + x_3 + x_4 = 1000$$
$$x_3 \geq 400$$
$$2x_1 + 3x_2 + 4x_3 + 5x_4 \leq 3300$$
$$3x_1 + 4x_2 + 5x_3 + 6x_4 \leq 4000$$
$$x_1,\ x_2,\ x_3,\ x_4 \geq 0$$

VII. Unbounded problems: What are they?

VIII. More advanced methods and alternative algorithms for solving LP problems. Include interior point methods

IX. Nonlinear programming methods. Explain tie-in with calculus, especially convexity.

X. Jobs in OR and mathematics