AD-A258 910

AFIT/GSO/ENS/92D-01

DTIC
ELECTE
JAN 8 1993
S C D

SPECTRAL AND SPATIAL PATTERN RECOGNITION IN DIGITAL IMAGERY

THESIS

John M. Amrine
Captain, USAF

AFIT/GSO/ENS/92D-01

93-00174

Approved for public release; distribution unlimited.

93 1 04 087

AFIT/GSO/ENS/92D-01

SPECTRAL AND SPATIAL PATTERN RECOGNITION IN DIGITAL IMAGERY

THESIS

Presented to the Faculty of the School of Engineering
of the Air Force Institute of Technology
Air University
In Partial Fulfillment of the
Requirements for the Degree of
Master of Science in Space Operations

John M. Amrine, B.S.
Captain, USAF

DTIC QUALITY INSPECTED 5

December 1992

Accession For
NTIS GRA&I ☑
DTIC TAB ☐
Unannounced ☐
Justification
By
Distribution/
Availability Codes
Dist | Avail and/or Special
A-1

Approved for public release; distribution unlimited.

Thesis Approval

Student: Captain John M. Amrine, USAF

Class: GSO-92D

Thesis Title: Spectral and Spatial Pattern Recognition in Digital Imagery

Defense Date: November 25, 1992

Committee:	*Name/Title/Department*	*Signature*
Advisor:	YUPO CHAN, PhD Professor of Operations Research Department of Operational Sciences School of Engineering	
Reader:	GEORGE E. CIPPERLY, Maj, USAF Instructor of Physics Department of Engineering Physics School of Engineering	

Acknowledgements

Now thanks be to God, [who] always causeth us to triumph in Christ ...

2 Cor 2:14

First and foremost, I would like to thank Jesus Christ, my Lord and Savior who helped me to persevere and endure through the AFIT experience. Next I would like to thank my thesis advisor, Dr. Yupo Chan and my thesis reader Dr. George E. Cipperly who were invaluable in teaching me how the disciplines of operations research and digital image processing can be married together. Without their guidance and resourcefulness, this thesis would not have been possible. Additional thanks are owed to Dr. Thomas S. Kelso. He provided the SPOT imagery used in this thesis and made improvements to the Training System–Image Processor so that I could complete my research. I would also like to thank all those officers in my Space Operations section who helped me work out the bugs in my programs and in LaTeX. My time here at AFIT was made much more enjoyable by these officers.

Special thanks are extended to those outside of AFIT who helped me in this thesis effort. Alex Merraus and Erwin Kalvelagen of the GAMS corporation helped me to understand the power and limitations of the GAMS system. Glen Kelly of the USGS EROS Data Center assisted me by explaining the finer points of the image processing performed on the SPOT imagery.

Finally, I would like to thank my wife, Elizabeth, for supporting me during my entire stay here at AFIT. It was sure nice to be able to concentrate on my studies since you took care of everything else. Thanks for helping me keep everything in perspective when I started to get too wrapped up in my studies. I can't wait to begin our new life in California with you. I love you.

John M. Amrine

Table of Contents

	Page
Acknowledgements	ii
Table of Contents	iii
List of Figures	ix
List of Tables	xi
Abstract	xiii
I. Introduction	1
1.1 Background	1
1.2 Problem Description	3
1.3 Research Objectives	4
II. Literature Review	7
2.1 Introduction	7
2.2 Spectral Pattern Recognition	8
2.2.1 Single Band Classification	9
2.2.2 Algebraic Combinations	12
2.2.3 Multispectral Classification	22
2.3 Spatial Pattern Recognition	28
2.3.1 Correlation Filtering	29
2.3.2 Line Detection	29
2.3.3 Directional Edge Detection	31
2.3.4 Texture Enhancement	33
2.3.5 Region Oriented Segmentation	35
2.4 Conclusion	36

Page

III. Methodology 38

3.1 Benabdullah & Wright Model 38

3.1.1 Example Problems 41

3.1.2 Analysis of Results 43

3.1.3 Analysis of Model 45

3.2 Examination and Modification of the Benabdullah & Wright Model 46

3.2.1 Examination of Constraints 46

3.2.2 Examination of Objective Function 59

3.2.3 Modification of Constraints 61

3.2.4 Modification of Objective Function 68

3.3 Combining of Single Channel Images 71

3.3.1 Multicriteria Optimization Description 72

3.3.2 Application of Multicriteria Optimization 78

3.4 Conclusion 84

IV. Spectral and Spatial Pattern Recognition in SPOT Imagery 86

4.1 Description of Imagery Data 87

4.2 Analysis of Data 89

4.2.1 Data Assumptions 89

4.2.2 Visual Analysis 93

4.2.3 Description of Target 94

4.2.4 Spectral Analysis 95

4.3 Computer Model Runs 102

4.3.1 Selection of Computer Model Equations 102

4.3.2 Running of the Model 106

4.3.3 Efficient Frontier Programs 112

4.3.4 Vegetation Index Programs 142

4.4 Conclusion 150

Page

V. Conclusions and Recommendations 153

5.1 Pattern Recognition Model 153

5.2 Combining Single Channel Images 156

5.3 Vegetation Indices and Band Ratio 160

5.4 Comparison of Processing Methods 161

5.5 Creation of GAMS Programs 162

5.6 Final Conclusion 162

Appendix A. LINDO Computer Runs 163

A.1 LINDO S2-1-1, Run #1 163

A.2 LINDO S2-1-1, Run #2 169

A.3 LINDO S2-1-1, Run #4 172

A.4 LINDO S2-1-1, Run #5 177

A.5 LINDO S2-1-1, Run #6 182

Appendix B. GAMS Computer Runs With Image Frame 188

B.1 GAMS S2-1-1, Run #1 188

B.2 GAMS S2-1-3, Run #2 192

B.3 GAMS S2-2-5, Run #3 196

B.4 GAMS S2-3-3, Run #4 201

B.5 GAMS S2-1-4, Run #5 205

B.6 GAMS S2-1-5, Run #6 209

B.7 GAMS S2-1-6, Run #7 213

Appendix C. GAMS Computer Runs; Min/Max Objective Function . . 217

C.1 GAMS S1-4, Run #1 217

C.2 GAMS S1-4, Run #2 222

Page

Appendix D. GAMS Computer Runs With Subregion Bounds 226

D.1 GAMS S2-3-4, Run #1 226

D.2 GAMS S2-3-4, Run #2 231

D.3 GAMS S2-3-4, Run #3 236

Appendix E. SPOT Subimage Gray Values 241

E.1 Channel 1 Gray Values 241

E.2 Channel 2 Gray Values 243

E.3 Channel 3 Gray Values 245

Appendix F. SPOT Subimage Gray Value Count and Program 247

F.1 Channel 1 Gray Value Count 247

F.2 Channel 2 Gray Value Count 249

F.3 Channel 3 Gray Value Count 251

F.4 FORTRAN Program: Gray Value Count 253

Appendix G. SPOT Subimage Water Regions and Program 257

G.1 Channel 1 Water Regions 258

G.2 Channel 2 Water Regions 259

G.3 Channel 3 Water Regions 260

G.4 FORTRAN Program: Characterization of Water 261

Appendix H. GAMS Program with Modified Objective Function 263

Appendix I. GAMS Program: Three–Channel Filter 268

I.1 GAMS Program to Locate Water Regions 268

I.2 Character Representation of Water Based on Three-Channel Filtering 274

Page

Appendix J. Channel 1 – Channel 2 Program 275

J.1 GAMS Program: Channel 1 – Channel 2 275

J.2 FORTRAN ProgramL: GAMS Adjacency Matrix Generation 279

J.3 FORTRAN Program: GAMS Cost Matrix Generation . . . 281

Appendix K. Channel 1 – Channel 3 Program 284

K.1 GAMS Program: Channel 1 – Channel 3 284

Appendix L. Vegetation Indicies Results & Programs 289

L.1 VI Matrix 290

L.2 NVI Matrix 291

L.3 Near-IR/Red Matrix 292

L.4 FORTRAN Program: VI and NVI 293

L.5 FORTRAN Program: Near-IR/Red 295

Appendix M. GAMS/OSL Program and Summary Output 297

M.1 GAMS/OSL Program 297

M.2 GAMS/OSL Summary Output 299

Appendix N. Network Formulation of B&W Model 302

N.1 Computer Runs 302

N.1.1 Analysis of Computer Runs 302

N.1.2 Conclusion on Network Formulation 303

N.1.3 Run #1 (S1-2) 304

N.1.4 Run #2 (S2-1-3) 306

N.1.5 Run #3 (S2-1-4) 308

N.1.6 Run #4 (S2-1-4) 311

N.1.7 Run #5 (S2-1-5) 313

N.1.8 Run #6 (S2-1-5) 316

Page

N.1.9 Run #7 (S2-2-2) 318

N.1.10 Run #8 (S2-2-4) 320

N.1.11 Run #9 (S2-2-4) 323

N.1.12 Run #10 (S2-2-4) 326

Bibliography 328

Vita 331

List of Figures

Figure		Page
1.	Integration of Spectral and Spatial Analysis	5
2.	Example of Segmentation by Thresholding	11
3.	Reflectance Curves for Various Land Covers	14
4.	Example of Minimum Distance to Means Classifier	24
5.	Linear Boundaries For Each Crop Type	25
6.	Example of Parallelepiped Classifier	26
7.	Example of Maximum Likelihood Classifier	27
8.	General 3 × 3 Convolution Mask	30
9.	Convolution Example	31
10.	Line Detection Masks	32
11.	Sobel Operators	33
12.	Laplacian Texture Enhancement Mask	34
13.	First Order Neighbors For Pixel #10	40
14.	3 × 3 Pixel Image With Associated Costs	42
15.	Border Constraint Examples	47
16.	4 × 4 Image Data With Frame	48
17.	3 × 3 Image Data With Frame	49
18.	Effect of Rotation on Border Length	51
19.	Area Constraint Examples	53
20.	Illustration of First and Second Order Neighbors	56
21.	Subregions with Diagonal Appendages	57
22.	Subregions with Diagonal Appendages of Width Greater Than One	58
23.	Stratified Digital Image	60
24.	Results of Runs #1 and #2	65

Figure		Page

25. Graphical Display of Constraints and Objectives in X Space 75

26. Graphical Display of Constraints and Objectives in Y Space 76

27. Mapping of D' Breakpoint in Z-Space 79

28. Graphical Display of Gray Value Total Bounds By Channel 81

29. Graphical Display of Combined Gray Value Total Bounds 81

30. Two-Channel Image with Gray Values and Locations 82

31. Efficient Frontier For Two Pixel Sized Subregion 83

32. Portion of Washington D.C. Mall Under Analysis 88

33. Multi-Channel Rectification Example 90

34. Efficient Frontier For Border Length = 192 122

35. Geometric Shapes for Area = 26 and Border Length = 44 139

36. Land Cover Reflectance Curves for SPOT Subimage 145

37. Principle of Linear Stretch Enhancement 146

38. Principle of Multipoint Linear Stretch Enhancement 148

39. Parallelepiped Classifier For Water 158

List of Tables

Table		Page
1.	Level Slicing Look Up Table	12
2.	Results for Location of Two Unitary Subregions	43
3.	Results of Seven Runs With Frame	50
4.	Location of a Four Pixel Subregion With Min Objective	60
5.	Location of a Four Pixel Subregion With Max Objective	61
6.	Relative Contribution of Pixels to Objective Function	71
7.	Mapping of Points From X Space to Y Space	76
8.	$\Lambda(D)$ and $\Lambda(E)$ Mapping in **Y**-Space	78
9.	Mapping of Efficient Frontier in **Y**-Space	83
10.	SPOT Imaging Bands	88
11.	SPOT Satellite Characteristics	89
12.	Classification of Water in Channel 1	98
13.	False Color of Water in Channel 2	99
14.	Classification of Water in Channel 2	100
15.	False Color of Water in Channel 3	102
16.	Classification of Water in Channel 3	103
17.	Results of Channel 1 – Channel 2 Runs	109
18.	Results of Channel 1 – Channel 3 Runs	112
19.	Results of Channel 1 – Channel 3 Runs with 192 Border Length	114
20.	Results of Channel 1 – Channel 3 Runs with 100 Border Length	116
21.	Results of Channel 1 – Channel 3 Runs with 64 Border Length	118
22.	Results of Channel 1 – Channel 3 Runs with 60 Border Length	119
23.	Results of Channel 1 – Channel 3 Runs with 52 Border Length	121
24.	Efficient Frontier Points For Border Length = 192	122

Table Page

25. Results of Channel 1 – Channel 2 Runs; Border Length ≤ 60 124

26. Results of Channel 1 – Channel 2 Runs; Border Length ≤ 100 125

27. Results of Channel 1 – Channel 2 Runs; Border Length ≤ 196 127

28. Varying Area and Border Length Runs 128

29. Results of Channel 1 – Channel 3 Runs; Area ≤ 28, Border Length $\leq$ 64 129

30. Results of Channel 1 – Channel 3 Runs; Area ≥ 24, Border Length $\leq$ 48 131

31. Results of Channel 1 – Channel 3 Runs; Area ≥ 24, Border Length $\leq$ 64 132

32. Results of Channel 1 – Channel 3 Runs; Area ≥ 26, Border Length $\leq$ 64 134

33. Results of Channel 1 – Channel 3 Runs; Area ≥ 24, Border Length $\leq$ 48 136

34. Results of Channel 1 – Channel 3 Runs; Area ≥ 24, Border Length $\leq$ 64 138

35. Results of Channel 1 – Channel 3 Runs; Area ≥ 26, Border Length $\leq$ 64 141

36. Results of Ten Runs 303

Abstract

The main purpose of this research is to develop a model that can be used to solve combined spectral and spatial pattern recognition problems. Many models in the past have focused on using only the spectral or spatial information available to solve a recognition problem. The basis of my model is a multiobjective discrete programming model developed by Benabdullah and Wright (B&W) (1). The model will be modified and then tested by solving a real world problem with SPOT multispectral imagery.

Several improvements were added to the B&W model, namely standard border length accounting and control over which pixels are selected by the model. The model was also improved to process more than one channel of imagery at a time. The model was successful in locating a user specified target, but this was not possible with all SPOT channel combinations. The Channel 2–3 combination caused the program to abort after 5,000 iterations.

Three improvements to this model are necessary. In the combining of the two channels, a multicriteria optimization, weighted sums approach was used. I manually changed the weights in my program. An "ADBASE" representation of this model would automatically try all the different lambda weights possible and thus find all the noninferior solutions. Another improvement needed is to reformulate the pattern recognition problem as a network with side constraints problem. This formulation of the model would ensure integer solutions and should result in quicker solutions. The final improvement is in the area of the objective function. In this thesis, several new objective functions are developed, but I was not able to take advantage of them. This problem was the result of the GAMS software limitations. The objective function needs to be reformulated or a new software package selected that is compatible with the new objective functions.

SPECTRAL AND SPATIAL PATTERN RECOGNITION IN DIGITAL IMAGERY

I. Introduction

1.1 Background

Remote sensing from space is a common, every-day occurrence that is witnessed nightly by millions of people as they watch the television weather report. In these reports, satellite imagery is shown and briefly analyzed to explain the current weather conditions. Although the era of artificial satellites began in 1957,[1] the thought of remote sensing from space is over 100 years old. In 1891, Ludwig Rahrmann of Germany was given a patent for a " 'New or Improved Apparatus for Obtaining Bird's Eye Photographic Views' " (20:528-529). Rahrmann's apparatus consisted of a rocket-propelled camera system that was recovered by parachute. By 1907, Alfred Maul expanded upon this rocket-camera system by adding gyrostabilization. In 1912, he successfully launched a 200 × 250 mm format camera to a height of 790 m (22:43-44).[2]

After World War II, remote sensing from space began in earnest. Between 1946 and 1950, small cameras were mounted on captured V-2 rockets that were launched from White Sands Proving Ground in New Mexico (20:529). These first space photos were of inferior quality, but they were significant for two reasons: (1) these photos were truly from space (altitudes ranged from 160 to 320 km) and (2)

[1]The first artificial Earth-orbiting satellite was launched in October 1957. This satellite was named Sputnik and was launched by the former Soviet Union.

[2]The author of this thesis was not able to view the original source document due to lack of access. The editor of referenced source listed the following information on the original source: Katz, A.H. "Oberservation Satellites: Problems and Prospects." Rand Paper P-1707. May 25, 1959, p. 1.

they "demonstrated the potential value of remote sensing from space" (20:529). Ten years after these V-2 rocket camera tests were complete, the US. launched the first remote sensing satellite. The TIROS (Television Infrared Observation Satellite) meteorological satellite was launched in April of 1960 (34:8-9). As sensors on-board the early meteorological satellites improved, "the prospect of looking *through*, not just *at*, the [Earth's] atmosphere had begun" (20:529). After the initial success of these early weather satellites, remote sensing from space was expanded into the manned space programs of Mercury, Gemini, Apollo, and Skylab (20:529-530), (29:51-61).

Since these early years of the space age, space remote sensing systems have improved and multiplied. Several commercial space systems that are currently operational are Landsat, NOAA (National Oceanographic Atmospheric Administration), and SPOT (systeme probatoire d'observation de la terre). The potential of remote sensing from space was also recognized by the U.S. military. The military is so dependent on weather data that it has developed its own weather satellite called DMSP (Defense Meteorological Satellite Program). In addition to this satellite system, the U.S. military has also developed surveillance/reconnaissance-type satellites to monitor its enemies. While these programs are highly classified, President Jimmy Carter acknowledged the existence of photoreconnaissance satellites in 1978. He also added that the U.S. would continue to build them (32:186).[3] As man has become dependent upon the use of space remote sensing systems, a new problem has developed. This problem is the analysis and interpretation of data collected from space remote sensors.

[3]The author of this thesis was not able to view the document cited by Stares due to lack of availability. Stares listed the following information on the original source document: *Documents on Disarmament 1978*, Publication 107 (U.S. Arms Control and Disarmament Agency, Washington, DC, October 1980), p. 586.

1.2 Problem Description

Before the data collected from a weather satellite is of any value, it must first be analyzed and interpreted by the meteorologist. This same principle also holds true for any user of data collected from a space remote sensor. Before imagery of the Earth can be used, it must first be analyzed and interpreted. Herein lies the problem, satellites are currently collecting data at a rate that is exceeding our rate to analyze and interpret it.

With the advent of multispectral imagery, the image interpretation problem has been even more compounded. Instead of taking one image of a particular area, multiple images are now taken simultaneously. Several operational systems currently have this multispectral capability. With the Landsat thematic mapper, seven images are taken simultaneously. With the NOAA AVHRR, this number is five and with the French SPOT system, this number is three. Rather than analyze and exploit one image, image analysts are now required to analyze multiple images of the same scene. This explosion in the amount of data requires computer analysis methods be developed for the imagery analyst.

Two Landsat examples vividly illustrate this data. The Landsat satellite is a multispectral sensor in sun-synchronous orbit at a nominal altitude of 900 km (20:532, 535-540). At this altitude, the total field of view is approximately 43,000 km^2 (20:544). This means that each image taken by Landsat contains this amount of area. The level of detail that can be seen in one of these images is limited by the resolution.[4] Full spatial analysis and interpretation of one image, on a pixel-by-pixel basis, would become too time consuming to accomplish manually. Another factor to consider is the amount of data contained in one image. A nominal Landsat image consists of 2,340 scan lines with 3,240 pixels per line for a total of 7,581,600 pixels per channel. If the four-channel multispectral scanner (MSS) is used to image the scene, four spectral observations per pixel will be collected for a total of over 30

[4]Resolution for Landsats 1-5 varies from 240 to 30 meters (21:536).

million points per image data set (20:538). Added on top of this are the gray values that each pixel can assume. For Landsat MSS imagery, each pixel can take one of 64 shades of gray (21:541). If the pixels are multiplied by the 64 gray levels, a data set of over 1.9 billion possible values results. Remember, this is the data set for only one scene. If an analyst was able to interpret each data point at a rate of one point per second, the time required to process the entire data set would take nearly one year! Not all remote sensing satellites are like the Landsat system, but the point is well illustrated. Some remote sensing satellites image at even higher gray levels. For example, the NOAA AVHRR images at 1024 gray levels (7). The sheer volume of data demands that much of the image analysis and interpretation be conducted by computer methods.

With this increase in spectral data, the logical question is how to best process the imagery. Should the spectral images first be combined and then processed or should each spectral image be processed individually (and then combined for a *composite answer*)? In Figure 1 these two approaches are shown shown graphically. The first method, indicated by the solid lines, combines the images from the different spectral bands. This composite image is then processed through a *normalized vegetative index* for identification of the various land covers. These land covers are then bounded in a spatial *pattern recognition model* and the final *bounded regions* are output. The second method, indicated by dashed lines, is similar, but the different spectral images are processed individually and combined at the output stage. These methodologies will be explored in detail in this thesis research.

1.3 Research Objectives

In general, this thesis applies operations research methods to locate user specified spectral and spatial patterns in digital imagery. It is assumed that the analyst knows what he or she is looking for and thus will know the spectral signature, size, and border length of the object. The overall goal of this thesis is to develop a general

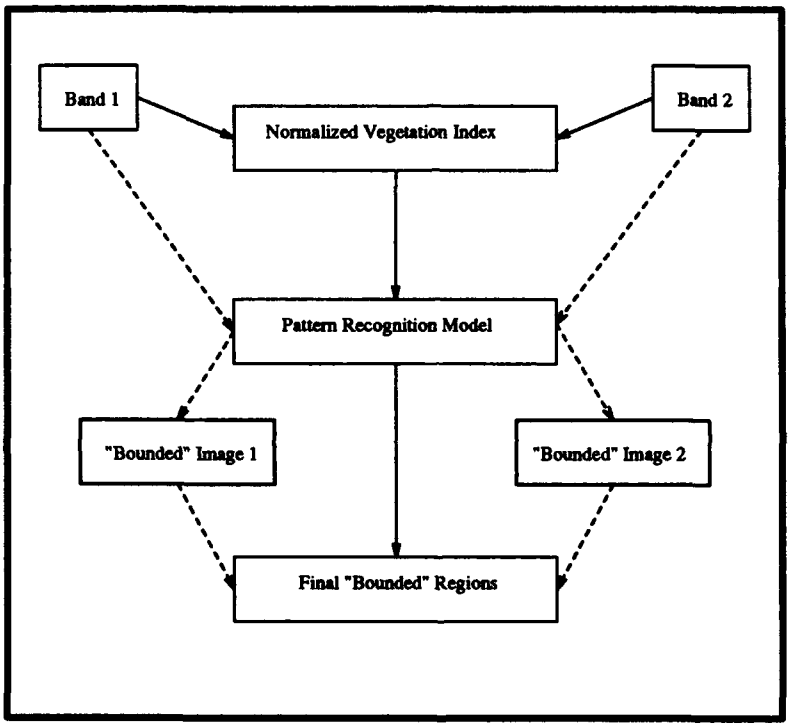

Figure 1. Integration of Spectral and Spatial Analysis

purpose model that can be used by an imagery analyst to locate spectral and spatial patterns. More specifically, the following objectives will be pursued in this research:

- *Analysis of Benabdullah and Wright Model:* One of the first objectives is to adapt the analysis of the Benabdullah and Wright Model. This model was developed to bound areas of land based on the users inputs (1:1). As currently written, the model is not designed to process digital imagery. This model will be reviewed from the perspective of an imagery analyst and improvements will be added as necessary. This model will form the basis for creating a new model.
- *Method to Combine Single Channel Answers:* In Figure 1, a method is needed to combine single channels that have been processed. A method that will be examined for this purpose is multiobjective linear programming. The concept of an efficient frontier will also be explored in its application to digital image processing.
- *SPOT Imagery Problem:* Apply the modified model to solve a real-world problem with SPOT imagery. Show that the improved model can locate targets of interest.
- *Comparison of Processing Methods:* In the above figure, the two processing approaches will be implemented and their accuracy and efficiency compared. Efficiency will be measured in terms of solution accuracy and computer execution time.
- *Computer Programs:* Develop all computer programs necessary to carry out the above objectives.
- *Review Of Current Methods:* In Chapter 2, I will review the current techniques used in digital image processing to recognize spectral and spatial patterns. Some of these techniques will be used to combine imagery channels before processing by the model.

II. Literature Review

2.1 Introduction

In this chapter, several methodologies for extracting patterns from digital imagery will be explored. Before these techniques can be explored, it is necessary to define what the term *pattern recognition* means. Tou and Gonzalez define *pattern recognition* as "the categorization of input data into identifiable classes via the extraction of significant features or attributes of the data from a background of irrelevant detail" (36:6). Harlow defines *classical pattern recognition* as "the means of deciding in which class a given pattern belongs" (12:77). For this thesis, the input data are confined to multispectral digital imagery. The "significant features or attributes" to be extracted from the imagery are different land cover types. The "irrelevant data" can be thought of as the data that remains unclassified once the "significant features" have been extracted.

Pattern recognition techniques for multispectral imagery can be accomplished in many fashions, but are usually based on one of three general characteristics of the imagery. The extraction process can be based on the spectral, spatial, or temporal characteristics of the data. This thesis will focus on the spectral and spatial characteristics of the imagery data to extract the desired land cover types. Several of the techniques currently used today are discussed in detail in the following sections. For completeness, a short description of temporal pattern recognition is given.

Temporal pattern recognition uses time as an aid in feature extraction. For example, crop surveys use multidate imagery to extract distinct spectral and spatial changes throughout a growing season (21:668). Without the multidate images, this analysis would not be possible since there would be no basis for comparison. An example is helpful in demonstrating how temporal pattern recognition can be accomplished. A field of winter wheat may be indistinguishable from bare soil when

freshly seeded in the fall and spectrally similar to alfalfa in the spring. If the significant feature to be extracted was the winter wheat field, neither image (fall or spring) would allow identification of the field. Even increasing the number of spectral bands would not necessarily help. If both images are used, the wheat field could be easily extracted since no other field cover would be bare in late fall and green in late spring (21:668). For the remainder of this chapter, only spectral and spatial pattern recognition techniques will be explored since the scope of this thesis is confined to these two methods.

2.2 Spectral Pattern Recognition

This method of pattern recognition is based on the principle that different feature types result in different gray value patterns, often among several imaging bands.[1] These patterns result from the "inherent spectral reflectance and emittance properties" of the imaged features (21:668). In this context, a spectral pattern does not refer to the geometric nature of the features.[2] Rather the term refers to the "set of radiance measurements obtained in the various wavelength bands for each pixel" (21:668). Spectral pattern recognition thus refers to the classification processes which use this pixel-by-pixel spectral information as the basis for extracting the desired information.

In this chapter, three general approaches to spectral pattern recognition will be explored along with several techniques under each approach. The three approaches are single band classification, algebraic combinations, and multispectral classification. The first only processes a single band of imagery at a time while the others process multiple bands at a time.

[1]For this thesis, gray value is defined as the "number that is assigned to a position (x,y) on an image and that is proportional to the integrated image value (reflectance, radiance, brightness, color coordinate, density) of a small area, called a resolution cell, or a pixel, centered in position (x,y) (24:316).

[2]Note a spectral pattern will show up as a geometric pattern in a gray value scattergram.

2.2.1 Single Band Classification In this portion of the chapter, methods for recognizing spectral patterns in a single band of imagery will be explored. These techniques can be applied to the output of a one band imaging system or to one band of a multiband imaging system. The concept of segmentation as applied to single band imagery is explained followed by a more detailed description of single band classification techniques. In general, segmentation is "a process of pixel classification; the picture is segmented into subsets by assigning the individual pixels to classes" (28:57). As was mentioned earlier, the basis for this assignment into different classes is by the different gray values (spectral characteristics) of the feature types. When segmenting an image, the analyst is only interested in parts (or segments) of the image. Segmentation is the process by which the image is broken into these parts (28:56). The level of subdivision in the image depends on the problem being solved. In other words, segmentation should stop when the objects of interest have been found (11:413). In this section, two methods of spectrally oriented segmentation will be described. They are gray level thresholding and level slicing.

2.2.1.1 Gray Level Thresholding Many types of images consist of two regions which have different gray level ranges. Several examples include an image of printed text or writing, an image of chromosomes, or an image of the Earth. In the first two examples, the writing and chromosomes are darker than the background. In the last example, the clouds are lighter than most types of terrain (28:61-62). In general, gray-level thresholding is used to segment the image into two regions — one for pixels below an analyst specified gray value and another for those pixels above the threshold (21:626).

Selection of the threshold gray value is usually accomplished by viewing the spectral histogram of the image. If the image consists of two spectrally different regions, the histogram should have two peaks. The threshold is then selected as the lowest gray value between the peaks. Ideally, threshold selection is an iterative process (24:225). The selection guideline given above is just the starting point. After

the threshold has been selected, the image is remapped into two analyst specified gray values. If the initial segmentation does not result in the desired image, the histogram of the original image is redisplayed, and a new threshold is selected and the image is again remapped. This iterative process ends when the analyst has extracted the desired regions of interest. A variation on this segmentation methodology is called *semithresholding.* With this method, pixels that are lighter than the threshold can be displayed as white and those darker than the threshold are displayed with their original gray values, or vice versa[3] (28:62).

Pattern recognition by thresholding is an effective and simple technique, but it is also of limited use. As was mentioned earlier, the objects of interest must have a characteristic range of gray levels different from the rest of the image. If this condition is not met, this technique will not work since segmentation of the objects of interest and background will overlap. Another limitation of this technique is that the remapped display consists of only two gray values. This binary image is useful for counting objects of interest in the image, but subtleties about each object will be missing. This limitation can be partially overcome by the technique of semithresholding mentioned above. Another pattern recognition technique of more utility is level slicing which will be discussed in the next section.

2.2.1.2 Level Slicing Level Slicing is a pattern recognition technique whereby the x axis of the image histogram is divided into several intervals or "slices". Like the thresholding technique, all the gray values within a single interval are remapped to a single gray value (21:627). With this technique, the analyst has the option of selecting where the boundary limits of each interval will fall. By viewing the histogram, the analyst can set the intervals between regions (between the histogram peaks, at the valleys), within a region (several intervals within a single val-

[3]The pixels lighter (greater in value) than the threshold do not have to be remapped to white. The gray value is set by the image analyst. White is used since the contrast between the two regions is enhanced.

ley to valley range of the histogram), or a combination of both. Thus, the remapped image appears to be a contour map with areas between the boundaries filled in with a single gray value[4] (21:627). For an example of this segmentation technique see Figure 2 below.

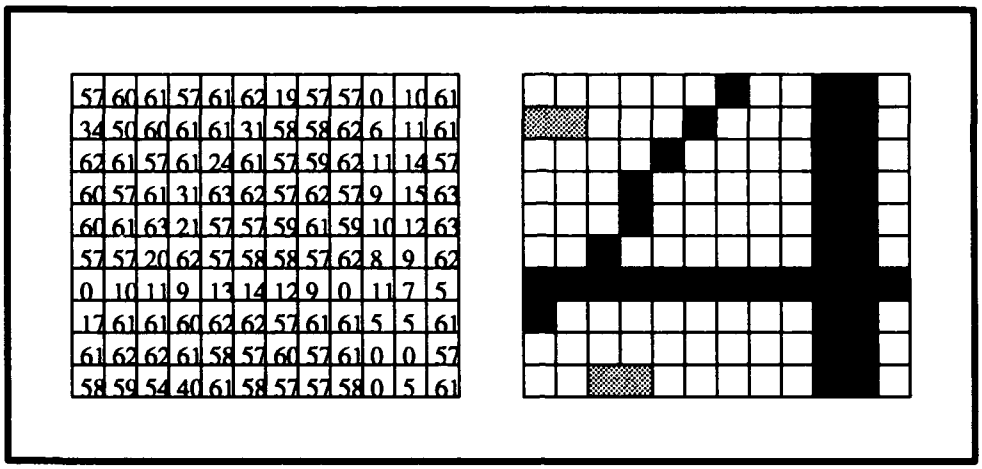

Figure 2. Example of Segmentation by Thresholding

In this example, I used four intervals to segment the image. On the left are gray values for six-bit imagery data.[5] In the right figure, the image has been segmented based on Table 1 below.[6]

From this example, one can see the utility of this segmentation technique in making sense out of all the gray value numbers in a digital image. This technique is still limited by the condition that the objects of interest must have a characteristic range of gray values different from the background. However, with this technique the whole image is still present in the remapped image. This is contrasted with thresholding, where a binary image results and thus little can be derived about the background. Which segmentation technique is best will depend on what the analyst

[4]Some texts refer to this segmentation technique as "density slicing" (12:86-87).

[5]With six-bit imagery, there are 64 possible gray values ($2^6 = 64$). Thus, the gray value range is from 0–63.

[6]Note: These are not the actual gray values for the objects listed. The gray value numbers are used to illustrate the concept of level slicing segmentation.

Table 1. Level Slicing Look Up Table

Gray Value Range	*Output Color*	*Actual Object*
0–15	Dk. Gray	Road
16–32	Gray	Stream
33–56	Lt. Gray	Buildings
57–63	White	Snow

is attempting to accomplish. "Level slicing is used extensively in the display of thermal infrared images in order to show discrete temperature images coded by gray level" (21:627).

2.2.2 Algebraic Combinations In this section, the process of recognizing spectral patterns is expanded to include the simultaneous processing of two or more bands of imagery at the same time. The multiple bands of imagery are combined through algebraic combinations of the imaging bands. Two methods commonly used are ratioing and vegetation indices. Some of these techniques are generic in nature while others are specific to a particular imaging system.

2.2.2.1 Ratioing With multispectral imagery, the data can be analyzed individually on a band-by-band basis or in a combination of two or more bands. Two bands can be combined through addition, subtraction, or most commonly, division of individual pixel gray values (12:87). This process of dividing the pixels in one band by the corresponding pixels in another band is known as "ratioing" (23:189). These ratios are very helpful to the image analyst in recognizing patterns since more information is usually contained in the ratioed image than in any single band image (21:390). There are two reasons why an analyst typically ratios an image. The first is "that certain aspects of the shape of spectral reflectance curves of different Earth-surface cover types can be brought out by ratioing. The second is that unde-

sirable effects on the recorded radiances, such as the effect of variable illumination (and consequently radiance) resulting from variations in topography can be reduced" (23:189).

Two examples of ratioing are helpful in demonstrating the pattern recognition capabilities of this method. Using image data obtained from the Landsat Multispectral Scanner (MSS) in bands 5 (red) and 7 (near-IR), ratioing can be used to differentiate among water, soil, and vegetation.[7] The spectral reflectance curve for water shows a steady decline between these two bands while there is a substantial increase for vegetation. The curve for soil shows a gradual increase. By dividing MSS band 7 by MSS band 5, a ratio results that can be used to identify water (ratio < 1), soil (ratio > 1), and vegetation (ratio $>> 1$) (23:189-191). Spectral reflectance curves for these types of land cover are shown in Figure 3[8].

A second example illustrates the reduction of topographic effects. If a sensor images a mountainous area, part of the scene is sunlit while the other part is in the shadow of the mountain. Given these varying illumination conditions, the gray values for the particular land type will be substantially lower in the shadowed area than in the sunlit area. "However, the ratio values for each land cover type are nearly identical, irrespective of the illumination conditions" (21:650). Thus, a ratioed image effectively reduces the gray value variation due to varying topography (21:650). In general, "spectral ratios negate the effect of any extraneous factors in sensor data that act equally well in all wavelengths of analysis" (21:392).

The mathematical expression of the ratio function is given by (15:135):

[7]Note: This same ratio, near-IR/red, can be applied to SPOT, Landsat Thematic Mapper (TM) and NOAA AVHRR imagery. The proper bands for each system are as follows: SPOT (XS3/XS2), Landsat TM (TM4/TM3), and NOAA AVHRR (AVHRR2/AVHRR1) (12:87-88).

[8]Note: These curves are not exact for each land cover type listed but are rather composites of many land covers of the same type. For example, the water curve gives the general reflectance response for lakes, rivers, oceans, etc.

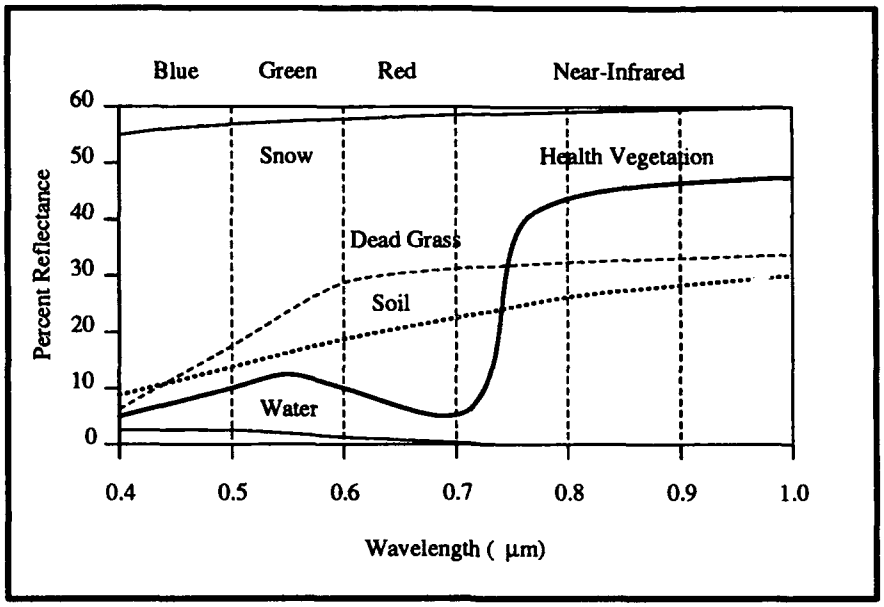

Figure 3. Reflectance Curves for Various Land Covers (13:19), (15:159)

$$GV_{ijr} = \frac{GV_{ijk}}{GV_{ijl}} \tag{1}$$

where GV_{ijr} is the ratioed gray value for the pixel located at row i, column j; GV_{ijk} is gray value at the same location in band k; and GV_{ijl} is the gray value in band l. Although this is a simple equation, the image analyst is cautioned against applying this methodology "blindly". There are several limitations both in the equation itself and in the application of the ratio. Given the above equation, one can see that the ratio can "blow up" when the gray value in the l band is zero. Another limitation of this equation is that ratios less than one are common and thus will compress much of the ratio data into a gray value of 0 or 1. A better algorithm to overcome these limitations is given below (12:86-87), (21:655):

$$GV_{ijr} = R \arctan\left(\frac{GV_{ijk}}{GV_{ijl}}\right) \tag{2}$$

where

GV_{ijr} = Gray value in ratioed image

R = Scaling factor to place ratioed data in correct range

$\arctan\left(\frac{GV_{ijk}}{GV_{ijl}}\right)$ = angle (in radians) whose tangent is the ratio of the gray values in bands k and l; if GV_{ijl} equals 0, this angle is set to 90 degrees.

"In the above equation the angle whose target is equal to the ratio of the two bands can range from 0 to 90 [degrees], or 0 to approximately 1.571 radians. Therefore, $[GV_{ijr}]$ can range from 0 to approximately 1.571R. If an 8-bit display is used, R is typically chosen to be 162.3, and thus $[GV_{ijr}]$ can range from 0 to 255" (21:655).

Additional caution is necessary when generating and interpreting ratioed images. The first limitation is that ratioed images are "intensity blind". "That is, dissimilar materials with different absolute radiances, but having similar slopes of their spectral reflectance curves, may appear identical. This problem is particularly troublesome when these materials are contiguous and of similar image texture"[9] (21:654). A second limitation is that noise should be removed before ratioing the images. The reason for this preprocessing is that "ratioing enhances noise patterns that are uncorrelated in the component images" (21:654). A final limitation of ratioed images is that they only compensate for multiplicative illumination effects. In other words, division of the gray values only cancels those factors that act equally well in the imaging bands and not those that are additive. (21:393, 654). For example, haze is an additive factor which should be removed prior to ratioing. Another solution to improve image pattern recognition is to use ratios of between-band differences and/or sums (21:393, 654).

One of the most common ratios used in vegetation studies is the ratio of the near-IR band to the red band. This ratio results in high values for healthy vege-

[9]One way to minimize this problem is to use a "hybrid color ratio composite" (21:654). This solution is beyond the scope of this thesis and the reader should review cited references for additional information.

tation, low values for bare areas (soil), and values less than one for water (12:87), (21:650), (23:189). This particular ratio is by no means the only one useful for image interpretation. In fact, the number of ratios possible from a multispectral image is $n = b(b-1)$ where n is the number of ratios possible and b is the number of bands (21:651), (24:159). The ideal ratio for the image analyst depends "upon the reflectance characteristics of the features involved and the application at hand" (21:651). Even when the application is known, some trial-and-error is often necessary to select the proper spectral ratio (21:654). In addition to the simple two band ratio explained in this section, there are many more combinations of multispectral data that can be used for spectral pattern recognition. These combinations are collectively called "vegetation indices" and will be discussed in detail in the next section.

2.2.2.2 Vegetation Indices One of the main applications of multispectral imagery is the monitoring of the world's food and fiber crops. The monitoring of these crops using in situ techniques is often not practical and thus, the monitoring has been accomplished by remote sensing spectral measurements. Much of the research in this area has focused on the analysis of Landsat Multispectral Scanner (MSS) and Thematic Mapper (TM) imagery. A common goal of this research has been to reduce the many bands of data down to a single value per pixel. This value is helpful in measuring "canopy characteristics [such] as biomass, productivity, leaf area, and/or percent [of] vegetative ground cover" (15:157-158). This section will identify several algorithms to extract this information from multispectral imagery.

While the Landsat satellite may be more commonly recognized, the National Oceanic and Atmospheric Administration (NOAA) meteorological satellites have also been used extensively for vegetation monitoring. With the advanced very high resolution radiometer (AVHRR) on board the NOAA satellite, two simple indices have developed to recognize vegetation in multispectral imagery. These indices are

called the *vegetation index* (VI) and the *normalized vegetation index* (NVI). Their respective equations are (21:597):

$$\mathrm{VI} = CH2 - CH1 \tag{3}$$

and

$$\mathrm{NVI} = \frac{CH2 - CH1}{CH2 + CH1} \tag{4}$$

where CH1 and CH2 represent AVHRR channels 1 and 2 respectively. In general, vegetation areas will result in high values for either index due to their relatively high near-IR (CH2) reflectance and low red (CH1) reflectance. In contrast, clouds, water, and snow have negative values due to their larger visible reflectance than near-IR reflectance. Rock and bare soil have similar reflectance values in both bands so their index value is close to zero (21:598). In operational use, the NVI is usually the preferred index since it compensates "for changing illumination conditions, surface slope, aspect, and other extraneous factors" (21:599). In highly vegetative areas, the NVI will vary from 0.1 to 0.6 depending on the vegetation density and greenness (21:599). The higher NVI numbers are associated with higher vegetation density and greenness. This same index can be used with Landsat MSS imagery if the appropriate substitutions are made.

Many vegetation indices have been developed over time to exploit the multispectral nature of Landsat imagery. In the following paragraphs, vegetation indices for the Landsat MSS and TM are explained.

- *Normalized Difference Vegetation Index* The NVI index used for AVHRR data can also be used with Landsat MSS imagery. The index is renamed as the Normalized Difference Vegetation Index (NDVI) and is defined as follows (23:192-193):

$$\text{NDVI} = \frac{MSS7 - MSS5}{MSS7 + MSS5} \tag{5}$$

where MSS 5 is MSS band number 5 (red) and MSS7 is MSS band number 7 (near-IR). Since the same bands are used (red and near-IR) in both the NVI and the NDVI, the results for the NDVI can be interpreted in the same manner as the NVI.

As we shall see, the number of different indices is considerable but many of the indices are redundant in information content. For this reason, they should be applied judiciously (15:158), (25:169-187). "If the principle of Occum's Razor ('don't increase the complexity of the problem without reasonable cause') is invoked we might conclude that the [near-IR/red ratio and NDVI] described above are sufficient for most causes" (23:193).

- *Transformed Vegetation Index* Deering et al modified the NDVI equation to produce the "transformed vegetation index" (TVI). In the TVI equations, 0.5 was added to eliminate negative values. They also extended the basic NDVI equation to create a similar algorithm using MSS5 and MSS6. The TVI algorithms are given below[10] (15:160):

$$\text{NDVI6} = \frac{\text{MSS6 - MSS5}}{\text{MSS6 + MSS5}} \tag{6}$$

$$\text{NDVI7} = \frac{\text{MSS7 - MSS5}}{\text{MSS7 + MSS5}} \tag{7}$$

$$\text{TVI6} = \sqrt{\text{NDVI6} + 0.5} \tag{8}$$

$$\text{TVI7} = \sqrt{\text{NDVI7} + 0.5} \tag{9}$$

[10]The writer of this thesis was not able to review the original source document for the TVI algorithms due to lack of availability. The cited reference gave the following information on the original source: Deering, D.W. et al. "Measuring Forage Production of Grazing Units from Landsat MSS Data," *Proceedings*, 10th International Symposium on Remote Sensing of Environment, Vol 2, 1975. pp. 1169-1178.

In their research, Perry and Lautenschlager found that the addition of 0.5 did not always eliminate the negative values. Therefore, they proposed new algorithms to correct for this deficiency (15:160):

$$\text{TVI6} = \frac{\text{NDVI6} + 0.5}{\text{Abs (NDVI6} + 0.5)} \times \sqrt{\text{Abs (NDVI6} + 0.5)} \tag{10}$$

$$\text{TVI7} = \frac{\text{NDVI7} + 0.5}{\text{Abs (NDVI7} + 0.5)} \times \sqrt{\text{Abs (NDVI7} + 0.5)} \tag{11}$$

where Abs is the absolute value and 0/0 is defined to equal one. The same TVI equations developed for Landsat MSS imagery can also be applied to Landsat TM imagery. The TVI equation for Landsat TM data is defined as (21:661, 663):

$$TVI = \left[\frac{TM4 - TM3}{TM4 + TM3} + 0.5\right]^{1/2}. \tag{12}$$

Like the NVI, the TVI value for Landsat MSS and TM imagery is "proportional to the amount of green biomass present within each pixel" (21:663). A Landsat image can be calibrated by relating ground-based biomass measurements to the TVI pixel values. "Separate calibration relationships must be established for each cover type and those become the quantitative basis upon which to depict the biomass present in similar [areas] throughout the scene" (21:663).

- *Perpendicular Vegetation Index* Another VI that can be used to recognize vegetation, soil, and water spectral patterns is the *Perpendicular Vegetation Index* (PVI). This index is based on the premise that there is a discrete axis in Landsat MSS four space on which the soil pixels lay. This soil axis contains the range of soils with low reflectance (wet) to those of high reflectance (dry) (15:167), (23:200-201). This index is perpendicular in the sense that the perpendicular distance from the soil axis, in the near-IR – red feature space, is computed for

each pixel[11] (13:88). In general, the greater the amount of vegetation in the pixel, the greater the value of the PVI (13:88), (27:307-312). Richardson and Weigand developed the PVI algorithms given below (27:307-312):

$$PVI7 = \sqrt{(0.355MSS7 - 0.149MSS5)^2 + (0.355MSS5 - 0.852MSS7)^2} \tag{13}$$

$$PVI6 = \sqrt{(0.498MSS6 - 0.457MSS5 - 2.507)^2 + (2.734 + 0.498MSS5 - 0.543MSS6)^2} \tag{14}$$

In general, interpretation of the PVI values are as follows: PVI $<$ 0, water; PVI = 0, soil; and PVI $>$ 0, vegetation (13:88), (23:202).

While these algorithms appear to be helpful in differentiating among vegetation, soil, and water, they should not be used with some forethought. The soil axis used in the PVI equations is based on 16 data points in Texas, from four dates in 1975. "It is unlikely that such a small and geographically-limited sample could adequately define the soil line on a universal basis" (23:201). To resolve this limitation, a locally-valid equation relating soil pixels in MSS7 – MSS5 (or MSS6 – MSS5) two space is needed. Secondly, the PVI equations are "based on the assumption that the maximum digital count in Landsat MSS bands 4, 5, and 6 is 127 with a maximum of 63 in Landsat MSS band 7" (23:201).

Perry and Lautergen also concluded that the PVI equations were computationally inefficient and that they did not distinguish between the left and right side (green vegetation and water) of the soil line. To overcome these deficiencies, they developed the following equations (25:169-187):

$$PVI6 = \frac{1.091MSS6 - MSS5 - 5.49}{\sqrt{1.091^2 + 1}} \tag{15}$$

$$PVI7 = \frac{2.4MSS7 - MSS5 - 0.01}{\sqrt{2.4^2 + 1}} \tag{16}$$

[11]This method is sort of an opposite of canonical components.

- *Tasseled Cap Transformation* Another VI that is based on soil occupying an axis in four dimensional Landsat MSS space is the "tasseled cap transformation" (23:200). With this transformation, the four MSS bands are used to create four new axes that are useful for agriculture monitoring. This transform creates the following new indices: the soil brightness index (SBI), the green vegetation index (GVI), the yellow vegetation index (YVI), and the no-such index (NSI). The SBI measures variations in soil background reflectance. The GVI measures variations in the vigor of green vegetation, while the YVI measures variations on yellowing of senescent vegetation. The NSI is related to atmospheric effects. Research has shown that the first two indices contain nearly all the scene information — 95 to 98 percent. This transformation has been so named, due to the cap region formed by vegetation in the brightness - greenness plane[12] (15:161), (21:660), (23:202-203).

  It is this cap region in the greenness - brightness plane that can be used to recognize agricultural vegetation on a Landsat MSS image. Specific regions outside this cap area have been identified as water, concrete and urban, and forest and natural vegetation. Overall, this transformation is a useful one for agriculture research since it has been rigorously tested and extensively used (15:162). It should be noted that the tasseled cap transformation has been extended to use Landsat TM data (21:660).

The list of vegetation indices goes on but Perry and Lautenschlager have shown that many of the most widely vegetation indices are equivalent. For example, they found that all TVIs were equivalent to their corresponding band ratios. Therefore,

[12]All three references cited a single Kauth and Thomas document as the original source for this information. The author of this thesis was not able to view the original source document due to lack of availability. The following information was given on this document: "The Tasseled Cap - A Graphic Description of Spectral-Temporal Development of Agricultural Crops as Seen by Landsat," *Proceedings: 2nd International Symposium on Machine Processing of Remotely Sensed Data.* West Lafayette, Indiana: Purdue University, 1976. pp. 41-51.

additional information may not be gained when two or more VIs are applied to the same image data (25:169-182).

2.2.3 Multispectral Classification "One of the most widely accepted applications of pattern recognition techniques to image processing has been the assignment off pixels to land use categories through multispectral classification" (21:96). This process requires that multiple images of a specific geographic region be taken in several spectral bands and that the images be in good registration (15:177). The "pattern" to be recognized is formed by the set of gray values for a single pixel on each of the spectral bands. Identification of the pattern results in a remapping of each pixel into a particular land cover type (23:276). This labeling of spectral patterns into land cover types may be performed using either of two methods: supervised or unsupervised classification.

2.2.3.1 Supervised Classification With this type of classification, *a priori* information about the land cover types and location is available. The image analyst "supervises" the classification process by specifying to the computer algorithm the ideal spectral patterns for each land cover type. Each pixel's spectral pattern is then compared to each of the ideal spectral templates and labeled with the name of the land cover it most resembles (21:668), (23:276). The first step involved in the supervised classification process is the mapping of these ideal spectral templates (sometimes known as training sets) into the n-space created by the n bands used to image the geographic region. For ease of explanation and understanding, only two imaging bands will be used in the discussion on supervised classification. However, it is rare that just two channels would be used in this analysis. The techniques discussed below can be applied to any number of channels (or bands) when implemented numerically (21:671). Three methods using the supervised classification process are the minimum-distance-to-means classifier, the parallelepiped classifier, and the maximum likelihood classifier.

- Minimum-Distance-to-Means Classifier This procedure begins with the training sets displayed graphically in two space (defined by two channels) as a scatter plot (i.e. a 2-D histogram).[13] Contrary to what one may think, "the pixels within each class do not have a single, repeated spectral value" (21:672). Therefore, for each land cover type "clouds of points" will form based on the spectral response patterns. These clouds "illustrate the natural centralizing tendency – yet variability – of the spectral properties found within each cover class" (21:672).

  An unknown pixel is classified by computing the distance between the pixel's location and each of the land cover means. After computing the distances, the unknown pixel is assigned to the land cover whose mean is closest.[14] This commonly used decision rule is mathematically simple and computationally efficient but it has one major limitation (15:209-211), (21:672-673). This classifier "is insensitive to different degrees of variance in the spectral response data" (21:673). Due to this limitation, this classifier is not used where spectral classes are close to one another and have high variance (21:675). However, this technique does give convenient linear boundaries between each land cover type.

  An example of this method of classification is shown in Figure 4. In this figure four types of agricultural crops are plotted in the 2-D space created by two imaging bands. The A's represent alfalfa, the B's are beans, the C's are corn and the W's represent wheat. The mean of each agricultural type is plotted with a solid dot.[15] Two unclassified points, X and Y are also plotted on this scattergram. Using the technique described above, the Euclidean distance from each unclassified point to each of the agricultural means is computed. In this

[13]Note this approach can also be applied in 1-space or n > 2 space.

[14]The computed distance can be Euclidean or "city block" (for computational efficiency with some loss of accuracy) distances.

[15]Note that the means are calculated by first classifying the pixels in some training set of data (which may be the data under consideration).

example, the X is classified as beans while the Y is classified as alfalfa. In Figure 5 the linear boundaries (based on minimum distance) for each crop type are shown.

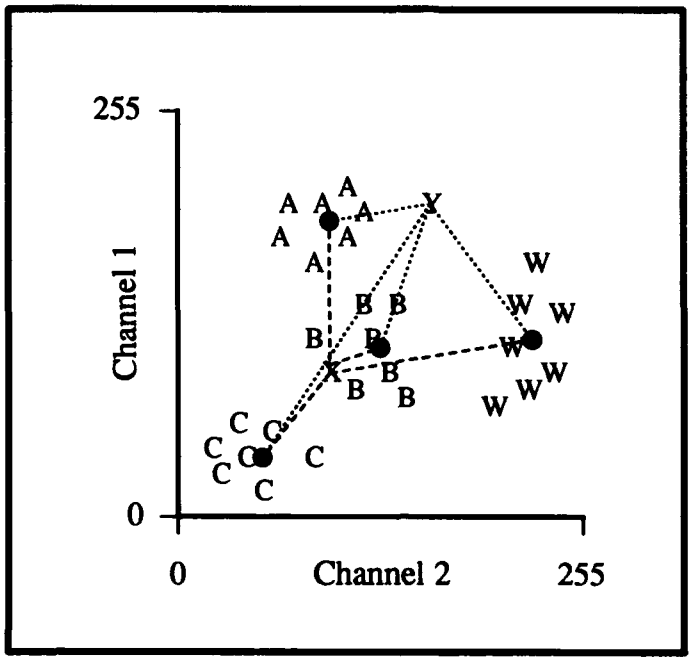

Figure 4. Example of Minimum Distance to Means Classifier. Adapted from Lillesand, p. 673, 1987.

- *Parallelepiped Classifier* One method for introducing sensitivity to category variance is to consider the range of gray values for a particular land cover. This range can be computed by taking the highest and lowest gray value for each training set (or land cover type) in each band. These boundaries form an n-dimensional parallelepiped in feature space (15:206), (21:674). For the 2-dimensional case, a rectangle would be formed around the cloud of points for each land cover type. An unknown pixel is assigned to a particular category if it falls within the specified ranges or assigned to an unclassified category if it does not fall within any of the land type ranges (15:206-207), (21:674).

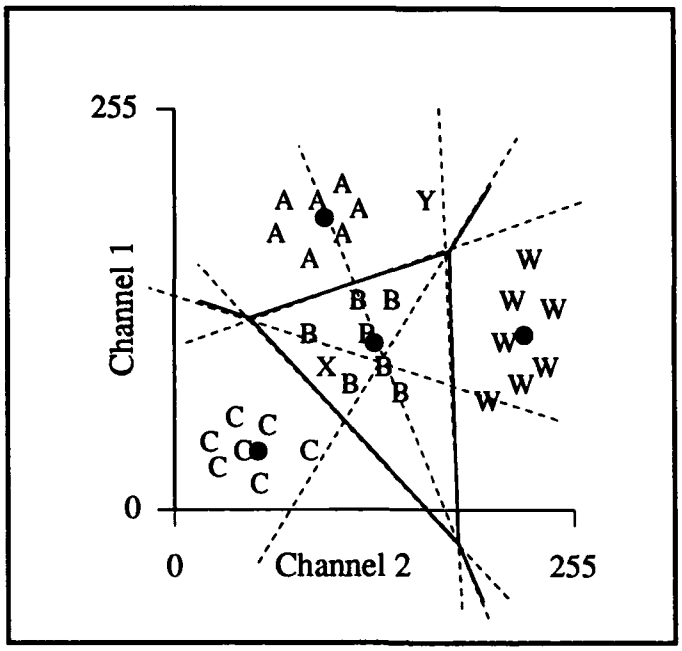

Figure 5. Linear Boundaries For Each Crop Type

This classifier is also very fast and computationally efficient but difficulties arise when category ranges overlap. When an unknown pixel lies in an overlap area, the computer algorithm may not know how to classify this pixel. This overlap is caused largely because categories with high covariance are poorly fit by rectangular decision regions (21:674-675). More importantly, the overlap may occur simply because two classes are just so similar that their variability inherently causes this problem. A method that accounts for this problem is discussed in the next section.

By using the same data set as the previous example, the parallelepiped classifier can be contrasted with the minimum distance to means classifier. In Figure 6, a rectangle encloses each cloud of points. Each rectangle effectively

bounds the spectral ranges for each agricultural crop. Using this classifier, the X is again classified as beans but the Y remains unclassified.[16]

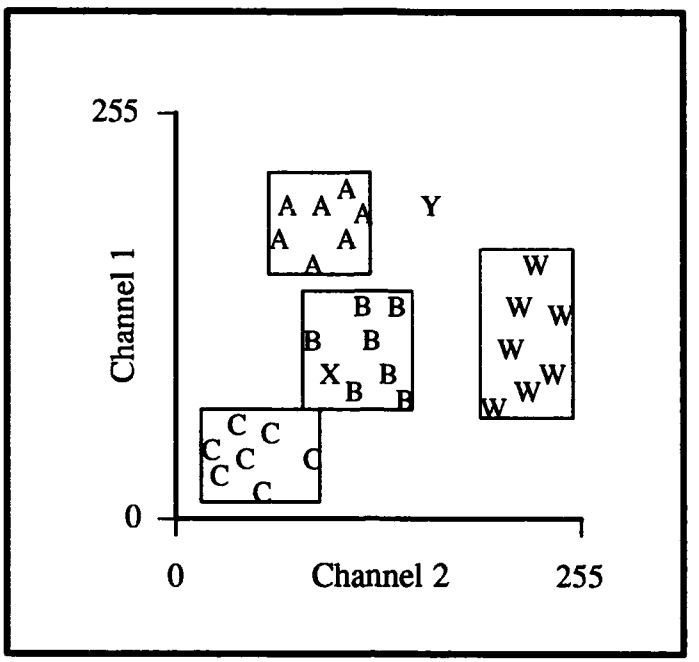

Figure 6. Example of Parallelepiped Classifier. Adapted from Lillesand, p. 674, 1987.

- *Maximum Likelihood Classifier* The maximum likelihood classifier takes into account the variance and covariance of the training data set when classifying an unknown pixel. This process is built on the assumption that the category training data set is Gaussian (or normally distributed). From the training data set, the probability density function for each category can be computed. These density functions are used to classify an unknown pixel by computing the probability that the pixel belongs to each category. The pixel is then assigned to the category with the highest probability or assigned to an unclassified category if all the probabilities are below an analyst specified threshold (15:212),

[16]Note that just because a pixel is labelled as a "B" here, it does not mean that it is not really a low probability-outlying "C", with respect to it's actual content. In fact, all classification schemes are based on statistical judgement calls based on various criteria for judging.

(21:675-676). Unlike the Minimum-Distance-to-Means Classifier, this method results in quadratic boundaries between land cover types.

In Figure 7, the probability density functions (PDF) are displayed for each agricultural crop. The PDF's value increases as the ellipses grow smaller. Using this classifier, the X is classified as beans but the Y remains unclassified.

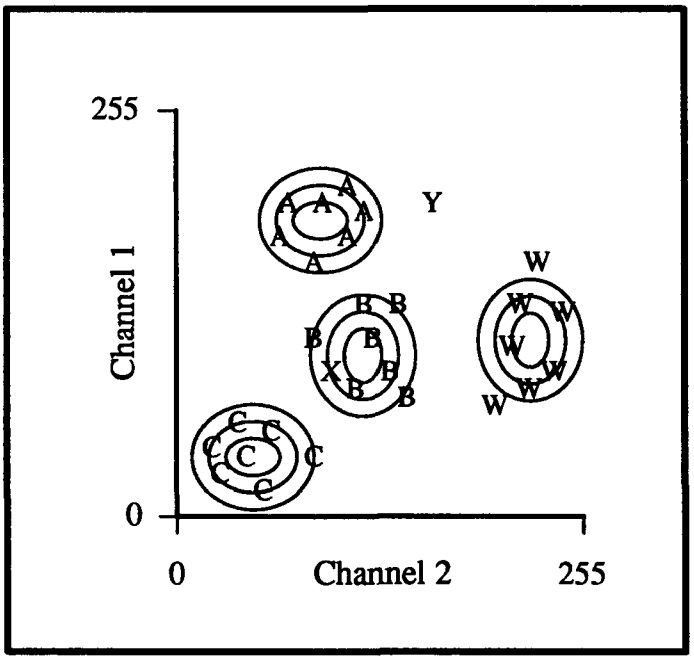

Figure 7. Example of Maximum Likelihood Classifier. Adapted from Lillesand, p. 677, 1987.

2.2.3.2 Unsupervised Classification Unsupervised classification differs from supervised classification in that the former does not use training data as the basis for classification. Rather, this classification method uses algorithms to group all the pixels in an image into their natural groupings based on their gray value patterns. The premise for this method is that gray values for a given land cover should be close together in measurement space, while the gray values for different land covers should be relatively well separated. Initially the identity of these groupings (or clusters) are

unknown and must be compared to ground reference data such as maps to determine the identity of the pixel groups (21:685). One of the primary advantages of this method is that the classifier identifies distinct spectral classes within the image. With supervised methods, this will not always hold true. There are hundreds of clustering algorithms for determining the natural groupings within a data set (15:215). One method of clustering that will be explained is the *K-Means* approach.

Similar to the supervised classifiers, *K-Means Clustering* begins with all the image pixels plotted in scattergram format in n-dimensional feature space. The analyst then dictates the number of clusters to be found in the image data.[17] The algorithm then arbitrarily locates that number of cluster centers. Each pixel is then assigned to the closest cluster mean. After all the pixels have been classified, the cluster centers are recomputed and the pixels are reclassified. This iterative approach continues until there is no significant change in the location of the cluster center. Once this step has been reached, the analyst determines the identify of each spectral class. One drawback to the K-Means clustering is that it is very computationally intensive (21:686).

2.3 Spatial Pattern Recognition

"Spatial pattern recognition involves the categorization of image pixels on the basis of their spatial relationships with pixels surrounding them" (21:668). With spatial pattern algorithms, patterns can be assigned to classes on the basis of image texture, pixel proximity, size, shape, orientation, location, repetition, or context (14:111), (21:668). Spatial pattern recognition is in direct contrast to spectral pattern recognition where the pixels are assigned to a class on a pixel-by-pixel basis. With spectral pattern recognition, the pixels can be entered in any order and the same classification results. This is not true for spatial pattern recognition. With this form

[17] A modification of the image analyst specifying the number of clusters is to allow the algorithm itself to choose the number of ***significant*** features.

of pattern recognition, pixels are classified on the basis of their neighboring pixels. Spatial pattern recognition tends to be more complex and computationally intensive (than spectral pattern recognition) since the classifiers attempt to mimic the spatial synthesis done by the analyst during image interpretation (21:668). This section will review several spatial pattern recognition techniques used with digital imagery.

2.3.1 Correlation Filtering A basic method for finding spatial patterns in digital imagery is through correlation filtering. With this method the target to be located is transformed into its frequency components through the application of the *Fourier transform.* The target can then be described by a combination of sine and cosine waves with various amplitudes, frequencies, and phases[18] (21:644). The image in which the target is to be located is also processed by the Fourier transform. The analyst then attempts to locate a frequency match in the image by scanning through the entire image. As the correlation between the target template and a location in the image rises, the probability that a target is present also increases. At a user specified threshold, the analyst indicates where the probable targets are located in the image. This method of spatial pattern recognition is implemented in the frequency domain[19] (21:644). The spatial pattern recognition methods in the following sections are implemented in the spatial domain.

2.3.2 Line Detection A common method for locating thin lines is to convolve the image with a directional derivative filter[20] (11:414), (28:113). Since line pixels will be located on the basis of shape, rather than their gray values of the line and background, "it is reasonable to use a filter function based on a derivative of the line pattern" (28:113). Using the 3×3 mask shown in Figure 8 below, the image is

[18]For a detailed explanation of the *Fourier transform* the reader should consult an electrical engineering textbook.

[19]The matched filtering approach described here is implemented in the frequency domain, but correlations can also be performed in the spatial domain.

[20]Convolution is defined as "digital image filtering usually performed using an $n \times n$ mask" (15:367). The value of n is usually odd.

convolved by computing the sum of the products of the mask weights and the pixels within the masks. Mathematically, the convolved value of the center pixel is given for a 3 × 3 window by:

$$
\begin{aligned}
C_{i,j} \quad = \quad & w_{1,1}z_{i+1,j+1} + w_{1,2}z_{i+1,j} + w_{1,3}z_{i+1,j-1} + \\
& w_{2,1}z_{i,j+1} + w_{2,2}z_{i,j} + w_{2,3}z_{i,j-1} + \\
& w_{3,1}z_{i-1,j+1} + w_{3,2}z_{i-1,j} + w_{3,3}z_{i-1,j-1}
\end{aligned} \tag{17}
$$

where

$C_{i,j}$ = Convolved value of the pixel in location (i, j),

$w_{i',j'}$ = Mask weight (or coefficient), and

$z_{i,j}$ = Gray value of pixel in location (i, j).

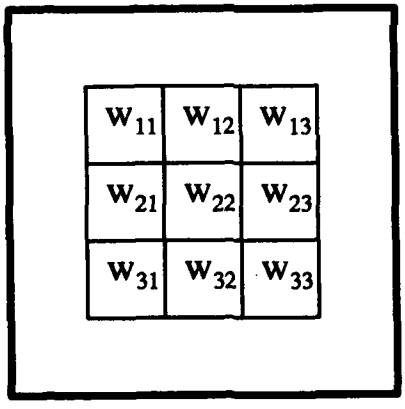

Figure 8. General 3 × 3 Convolution Mask

A mathematical example helps to illustrate this concept. If an analyst wants to remove spikes in imagery such as noise, a smoothing filter can be used. A simple smoothing filter is shown in Figure 9. In this example, the 3 × 3 smoothing filter has weights of one ninth in each cell. To convolve the center pixel of the middle 3 × 3 image, each pixel gray value is multiplied by the corresponding weight in the

smoothing filter. In this example, each pixel is multiplied by one ninth. The nine products are added together for the convolved value of the center pixel. The result of this filter is that the original gray value of 200 has been reduced to 96, which is more in line with the surrounding gray values. With this example, only the center pixel can be convolved because eight pixels need to surround the pixel being convolved. More generally, the mask is moved to place its center over each pixel to calculate the convolved value for each pixel.

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

Smoothing Filter

85	56	72
90	200	100
75	88	98

Gray Values

	96	

Convolved Pixel Value

Figure 9. Convolution Example

Using the masks shown in Figure 10, lines of one pixel thickness can be found. To locate the line, the center of the mask is placed around the first pixel in the first row and then systematically moved pixel by pixel through each row. With a constant background, the maximum response will result when a line passes through the middle row of the horizontal mask. A similar maximum response results for lines oriented in the manner of the other filter masks[21] (11:415).

2.3.3 Directional Edge Detection A more common method of spatial pattern recognition is edge detection. This is true because thin lines do not frequently occur

[21]There are limitations on these line masks but discussion of this topic is beyond the scope of this thesis. For more information, reference Rosenfeld and Kak, pp. 114-117.

-1	-1	-1
2	2	2
-1	-1	-1

Horizontal

-1	-1	2
-1	2	-1
2	-1	-1

+45

-1	2	-1
-1	2	-1
-1	2	-1

Vertical

2	-1	-1
-1	2	-1
-1	-1	2

-45

Figure 10. Line Detection Masks (11:416)

in digital imagery (11:415). "Edge detection is an image segmentation method based on the discontinuity of gray [values] or texture at the boundary between different objects. Such a discontinuity is called an edge. An edge separates two regions of relatively uniform but different gray [values] or texture" (24:225). Although there are many edge detection algorithms, only two methods will be explored below.

2.3.3.1 Running Difference One simple method for detecting edges is called the *running difference.* With this method, the image is shifted one pixel (either horizontally, vertically, or diagonally) and the corresponding pixels in the original and shifted image are differenced.[22] If there is no difference, a mid-gray value (e.g. 127) is substituted for the pixel. If the difference is positive, a darker gray value (e.g. 30) is substituted and a lighter value (e.g. 200) is substituted if there is a positive difference.[23] Thus, transitions from light to dark regions produce white lines and transitions from dark to light produce dark lines. The result of this operation is a gray image with white and dark lines surrounding the objects[24] (15:144). It should be noted that this processed image requires shifts in more than one direction.

[22]Note: Image is shifted in the direction of the edge orientation to be found (e.g. image is shifted horizontally to find horizontal lines). Since the image is limited in the directions it can shift, only edges in the orthogonal directions can be found.

[23]These gray values were assigned on the basis of eight bit data. Therefore, the range of gray values is from 0 to 255.

[24]The author of this thesis was not able to view the original source document due to lack of availability. Cited reference gave the following information on the original document: ***Handbook on Image Processing.*** Spatial Data Inc., Goleta, California 1975.

2.3.3.2 Sobel Operators Another method for recognizing edges is the *Sobel Operator* method. These operators are based on the mathematical principle of the gradient.[25] The gradient is calculated by taking the partial derivatives $\partial f/\partial x$ and $\partial f/\partial y$ at each pixel location. In digital imagery, derivatives may be implemented by differencing. Differencing alone is limited since it enhances noise. Sobel operators are optimal since they provide differencing as well as a smoothing effect on the image. Sobel operators convolve an image using the 3×3 masks shown in Figure 11 (11:418-419).

-1	-2	-1
0	0	0
1	2	1

Horizontal (X)

-1	0	1
-2	0	2
-1	0	1

Vertical (Y)

Figure 11. Sobel Operators (11:418-419)

As was mentioned earlier, the mask systematically operates upon each pixel in the image. The result is a gradient image with lines oriented in the direction for the particular Sobel operator.

2.3.4 Texture Enhancement Before the concept of *texture enhancement* can be explained, it is necessary to define the term *texture* in the context of digital imagery. Texture can be defined as "the spatial distribution of the gray [values] and discrete tonal feature" in the vicinity of each pixel (14:124, 245). In other words, texture is a measure of how much a pixel varies from its surrounding neighborhood. In the section on edge detection, location of the edge was based upon abrupt changes

[25] For a given location, the gradient calculates the direction of the maximum rate of change for a given function. For this thesis, this function is defined as the digital image.

in gray values. This same concept can be applied to changes in texture to locate region boundaries. This approach is valid since changes in texture usually indicate a different region (24:232). Thus, the goal of texture enhancement is to emphasize the boundaries between regions of homogeneous texture (35:32).

There are a variety of enhanced algorithms to recognize changes in texture patterns. In general, they all compare the gray value of the pixel of interest to the average of the surrounding neighborhood. Any departure from the average is exaggerated and remapped onto the output data set. One of the simplest texture enhancement methods is the Laplacian texture enhancement. This process also uses a filter to convolve the image, pixel by pixel. This mask is shown below in Figure 12[26] (35:32-33). Filters of this sort are in the category of non-directional, high pass, high boost filters. The overall effect of this process is "to 'sharpen' all boundaries, and to increase the apparent resolution" (35:33). For this reason, texture enhancement is particularly effective on blurred images (23:258), (24:137).

0	-1	0
-1	5	-1
0	-1	0

Figure 12. Laplacian Texture Enhancement Mask (35:32-33)

[26]Note: This mask is not the one for computation of the Laplacian. The digital Laplacian at location (x, y) of an image f is given by $(\nabla^2 f)(x, y) \equiv [f(x+1, y) + f(x-1, y) + f(x, y+1) + f(x, y-1)] - 4f(x, y)$ (28:89). The given mask subtracts the Laplacian from image f (23:258-259), (24:137).

2.3.5 Region Oriented Segmentation This method of pattern recognition consist "of sequentially extracting solid objects or regions, rather than edges or thin curves, from [an image]" (28:138). The goal of this process is to extract objects of interest from the image rather than to partition the complete image (28:138). This is in contrast to the edge detection algorithms which only outlined the regions of interest. Two methods of region oriented segmentation will be discussed: region growing and split and merge.

2.3.5.1 Region Growing As the name implies, *region growing* is a process of grouping pixels together to form regions. One of the simplest region growing methods is by pixel aggregation. This method starts with "seed" pixels and then adds (or grows) additional pixels that meet the decision criteria. The decision criteria may be based on texture, relative position to seed pixel, size and shape of object to be found, and gray value. When new pixels are accepted, the decision process is repeated for the new neighboring pixels. The growth of the region will terminate when no acceptable neighbors exist (11:458-461), (28:140).

An important step in this process is the selection of the seed pixels since different seed pixels can cause different results. To begin the growing process, a pixel should be chosen which is very typical of the object (or region) of interest. There are two ways to identify these seed points. The first way is to have *a priori* knowledge about the objects to be found. If this is the case, these ideal values should be selected as the seed points. A second method is used if *a priori* knowledge is not available. The property values of each pixel are first measured and then plotted in n-dimensional space, where n equals the number of property values measured per pixel. A scatter plot of these values will show natural groupings of clusters. A pixel which is located near the centroid of a cluster can be selected as a seed pixel. Once the growing begins, the aggregation of additional pixels may depend on their property values relative to the growing object (28:61, 141). This statement leads to the second important step in this process — selection of decision criteria. An accurate

description of the object of interest must be found in the decision criteria. If not, the region may not be found. For example, criteria based on gray value alone may be inadequate if connectivity and adjacency are also important (11:460).

2.3.5.2 Split and Merge At the opposite extreme, the digital image is taken as a single region that is partitioned by repeated splitting. One method of dividing the image is by bisection. If the image does not meet the decision criteria, the image is divided into quadrants.[27] If a quadrant does not meet the decision criteria, we divide it into subquadrants and so on. As the image is split into various sized regions, adjacent regions that meet the decision criteria can be merged. This splitting and merging continues until no further merging or splitting is possible (11:461-463), (28:148-149). The end result is the objects of interest identified in the image.

2.4 Conclusion

In this chapter many of the common methods used for spectral and spatial pattern recognition were discussed. For spectral pattern recognition, the three general approaches explained were single band classification, algebraic combinations, and multispectral classification. In the next chapter, three of the methods explained will be used to combine two bands of imagery: Vegetation Index, Normalized Vegetation Index, and band ratioing. For spatial pattern recognition, five approaches were explained: correlation filtering, line detection, directional edge detection, texture enhancement, and region oriented segmentation. In spectral pattern recognition, pixels are assigned to a class on a pixel-by-pixel basis. This is in direct contrast to spatial pattern recognition where pixels are classified based on their neighboring pixels. Of the two recognition approaches, spatial pattern recognition tends to be the more complex and computationally intensive. Each method has its advantages

[27]The decision criteria mentioned under the split and merge method are the same type as those mentioned in the region growing section.

as well as it limitations. The *best* method for an image analyst depends on the type of image data and the specific image exploitation problem. In the following chapter, a method for combining both spectral and spatial information (of a particular land cover type) will be explored. By combining the two types of information, a model can be developed to find land cover types specified by the image analyst.

III. Methodology

The basis of my model, to recognize spectral and spatial patterns, is a multiobjective discrete programming model developed by Benabdullah and Wright (B&W) (1). This model was created to bound multiple areas of land based on the user's inputs. The model is limited in its capacity to finding/bounding rectangular areas (1:1). As such, the model was not originally designed to process digital imagery. The following paragraphs explain the principles of the B&W model as applied to digital imagery. The equations of this model are first explained, followed by several example problems, and finally an analysis of the results. Following this analysis of the B&W model, the model is examined from the perspective of an imagery analyst. Significant shortfalls are identified and changes to the model are proposed. If this model is applied to digital imagery, the analyst is required to know a priori the following target characteristics: size, border length, and spectral range. The last section of this chapter explains a method for processing more than one channel at a time. With multispectral imagery several channels are available so a method is needed to process more than one channel.

3.1 Benabdullah & Wright Model

For the bounding of multiple subregions within a digital image, the following equations are given (1:9-12):

$$\text{Minimize } Z = \sum_{k=1}^{K} w_k \left[\sum_{i=1}^{n} c_i x_{ik} \right] \tag{18}$$

subject to:

$$\sum_{i=1}^{n} x_{ik} = M_k \quad \forall k \tag{19}$$

$$x_{ik} - x_{jk} - P_{ijk} + N_{ijk} = 0 \quad \forall i, j \in T_i, k \tag{20}$$

$$\sum_{k=1}^{K} x_{ik} \leq 1 \quad \forall i \tag{21}$$

$$\sum_{i=1}^{n} \sum_{j \in T_i} s_{ij}(P_{ijk} + N_{ijk}) = L_k \quad \forall k \tag{22}$$

$$x_{ik}, P_{ijk}, N_{ijk} = 0, 1 \quad \forall i, j \in T_i, k. \tag{23}$$

Each of the variables in the above equations need to be explained before the model can be appreciated and also run. In Equations 18, 19, 20, 21, and 23, x_{ik} is a binary variable that indicates whether a given pixel, i, is assigned to subregion k:

$$x_{ik} = \begin{cases} 1 & \text{if pixel is assigned to subregion } k \\ 0 & \text{otherwise.} \end{cases} \tag{24}$$

The range of i is from 1 to n where n is the total number of pixels to be considered for incorporation into a subregion. In the equations listed above, k is the number of subregions to be found in an image. In the cost objective function (Equation 18), w_k is the weight for subregion k, and c_i is the cost for incorporating pixel x_{ik} into a subregion. The objective function seeks to minimize the cost of the k subregions.

In Equation 19 (first constraint), M_k represents the size of the kth subregion. For a digital satellite image, this would equate to the number of pixels in the target subregion. This interpretation can be seen in the equation — the sum of the incorporated areas equals the size of the kth subregion.

The second constraint (Equation 20) is necessary to ensure nonoverlapping of pixels between subregions (1:9). This constraint also defines the binary variables P_{ijk} and N_{ijk}.

> For any adjacent [pixels] i and j, if only one of [the pixels] is selected ($x_i = 1$, $x_j = 0$ or $x_i = 0$, $x_j = 1$), then either P_{ij} or N_{ij} must equal 1 and the other must equal 0. For example, if at optimality [pixel] i is acquired ($x_i = 1$) and [pixel] j is not ($x_j = 0$), then Equation [20] would be satisfied if $P_{ij} = 1$ and $N_{ij} = 0$. If both [pixels] i and j are acquired

($x_i = x_j = 1$) or neither is acquired ($x_i = x_j = 0$), then P_{ij} and N_{ij} must both equal 1 or both equal 0. Because the external border function is being minimized, then the smallest values assigned to P_{ij} and N_{ij} that would satisfy Equation [20] would be 0 ($P_{ij} = N_{ij} = 0$) at optimality[1] (1:6).

In this constraint, T_i is the first order neighbors adjacent to pixel i. See Figure 13 for a graphical depiction of first order neighbors. In this figure, the first order neighbors for pixel number 10 are pixels 6, 9, 11, and 14. For pixel number 16, the first order neighbors are 12 and 15.

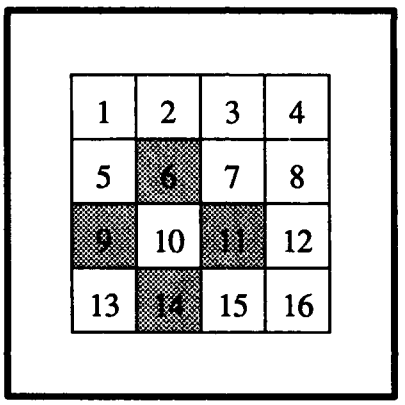

Figure 13. First Order Neighbors For Pixel #10

The third constraint (Equation 21) is similar to the second constraint and ensures that each pixel i is only assigned at most to one subregion. The fourth constraint (Equation 22), sets the border length for the subregions to be found. In this equation, the variable s_{ij} is the length of the border between pixels i and j (1:5). For the application of this constraint to digital imagery, I will assume pixels lengths and widths of one.[2] Variables P_{ijk} and N_{ijk} are "mutually exclusive binary decision

[1]Note: In the explanation given for variables N and P, the general case of only finding one subregion in the image is discussed. For this reason, the k subscript is not used since $k = 1$. In Equation 20 the k subscript is used to find multiple subregions in the image. The general explanation given for the one subregion also applies to the multiple subregion search.

[2]In general, this is not a bad assumption since most satellite remote sensing systems have equal resolution in both length and width. However, the border length of the object must now be measured in terms of pixel units.

variables that sum to [one] if the border separating [pixels] i and j in the final solution is an external border ... and [zero] otherwise" (1:5-6). In the Benabdullah and Wright model, the range for feasible external border lengths for a specified M value (subregion size) is given. This range consists of the even values between B^M_{min} and B^M_{max} where (1:7-8):

$$B^M_{max} = 4M \tag{25}$$

$$B^M_{min} = 4(< \sqrt{M} >) + 2t. \tag{26}$$

In Equation 26, t is defined as follows:

$$t = \begin{cases} 0 & \text{if } M - (< \sqrt{M} >)^2 = 0 \\ 1 & \text{if } M < (< \sqrt{M} >)^2 + (< \sqrt{M} >) + 1 \\ 2 & \text{if } M \geq (< \sqrt{M} >)^2 + (< \sqrt{M} >) + 1 \end{cases} \tag{27}$$

$$\text{and } < \sqrt{M} > = \text{the integer part of } \sqrt{M}.$$

With this border length constraint, the B&W model "guarantees contiguity of the resulting land area for values of L near B^M_{min}" (1:8). If the border length, L, is set to B^M_{min}, a square region will result.

3.1.1 Example Problems Before this model is run on an actual digital image, it is necessary to run the model on some example problems. The results of these problems will highlight the capabilities and limitations of the model. These example problems are ones in which the analyst can visually determine the correct answer before running the model. Validation that this model can solve the example problems will give the model more creditability in solving the real spectral and spatial pattern recognition problems. Several problems are created using the 3×3 pixel image shown in Figure 14.

In this digital image, the pixels are numbered sequentially from one to nine, starting in the upper left hand corner and finishing in the lower right hand corner.

11	5	6
8	13	2
5	1	1

Figure 14. 3 × 3 Pixel Image With Associated Costs

The numbers shown represent the *cost* to acquire a pixel into a subregion[3]. Therefore, the cost to acquire pixel number four is eight[4]. These example problems were run using the **L**inear, **I**nteractive a**N**d **D**iscrete **O**ptimizer (LINDO)[5] software on a Zenith 248 personal computer.

One of the easiest problems to solve would be the location of two subregions, both of one-pixel size. Remembering that the B&W model has a minimize objective, the solution is easily identified as pixels eight and nine. When the problems were run with border length constraints of four and four, and subregion size constraints of one and one, an incorrect answer resulted. The B&W model selected pixels seven and nine with a corresponding cost of six.

To further test the validity of this model, 45 other problems were run on this same data set. The results from these problems were significant. Out of the 46 problems run, only 25 had the right objective function cost and out of these 25, only

[3]In the context of applying the B&W model to digital imagery analysis, the term *cost* is synonymous with the pixel gray value.

[4]The actual numbers used in this 3 × 3 image are not representative of an actual image, but are rather used for illustrative purposes. The numbers were arbitrarily chosen to create a necessary data set

[5]LINDO is "an interactive linear, quadratic, and integer programming system" (31:1).

ten had the correct pixels selected. Given an accuracy rate of only a little over 20 percent, further research was required.

Further investigation on the B&W model revealed some insight into the border length constraint (Equation 22). For non-corner pixels, s_{ij} is the length of the side of a pixel and for a corner pixel, s_{ij} is twice the length of a pixel side. I have set the length of a pixel side to one but any value can be used without affecting the solution (1:12). In addition to these border length accounting rules, one should note that the pixel sides on the edge of the image have no value. Therefore, they (side of the pixels which form the image edge) are not included in the border length accounting. Additional research on this part of the B&W model has been conducted by Dr. Chan at the Air Force Institute of Technology. He states that the border length, L, is twice the perimeter length of the desired subregion (6:32).

With this new information, several more runs were made on the selection of two subregions both of size one. The results of the runs as well as the constraint inputs are shown in Table 2. The complete listing of the LINDO input and a summary of the output for each completed problem is given in Appendix A.

Table 2. Results for Location of Two Unitary Subregions

Border Constraint			*Pixels Selected*		
Run	*Length 1*	*Length 2*	*Region 1*	*Region 2*	*Cost*
#1	4	4	7	9	6
#2	3	2	-	-	*
#3	4	3	-	-	*
#4	6	8	8	9	2
#5	4	6	9	8	2
#6	8	8	8	9	2
*No solution after 2,000+ iterations					

3.1.2 Analysis of Results With the results of the six computer runs, the intricacies of the model can be explained.

- *Run #1* In run number one, border lengths of four and four were selected. Given the additional information on how border lengths are calculated, the feasible pixels can be determined. The feasible set is the corner pixels and the middle pixel. With a minimize objective function, the optimal solution is reduced to pixels seven and nine.
- *Run #2* In the second run, the lengths were changed to three and two. This problem was run to test if pixel number nine actually had a border four as opposed to two. Although no solution resulted, this run proved the border length of pixel number nine was not two.
- *Run #3* In run three, the border length of three was kept but switched from region one to two. The first border length was set at four. With this combination, one would expect pixels number nine and number eight to be selected. Pixel number eight only has a border of three since only three pixels are adjacent to it.[6] The B&W model failed to produce a solution after 2,000 iterations.
- *Run #4* In run number four, the border lengths were doubled and switched between the subregions. With this combination of border lengths, the model finally selected the correct pixels. With this combination, both border lengths are twice the actual perimeter.
- *Run #5* In run five, the border length of eight from run four was divided by two. However, the border length of six was retained. With these constraints, pixel number nine must be selected for a border length of four and pixel number eight must be selected for a border length of six. These are the minimum border lengths required for the model to correctly select pixels eight and nine.
- *Run #6* In the final run, both border lengths were set to eight. Pixels eight and nine were again selected. One may question why two corner pixels were not selected. It appears that if the border length is above the minimum required

[6] Remember, border length is computed on the basis of adjacent pixels.

for a pixel, the objective function drives the solution. In other words, given a choice between two cells where the border length constraint is either an integer multiple of the pixel border or the border length constraint is greater than the pixel border, the model will pick the lower cost pixel.

3.1.3 Analysis of Model Given the results from the example problems, two limitations about the B&W model are evident. The first noticeable limitation is that the accounting of the boundary lengths is not straight forward. Benabdullah and Wright stated that the only accounting difference between pixels was between corner and non-corner pixels. However, if this was true, the example problem with border constraints of three and four should have been solvable. The minimum border length constraints for the example problem was six and four. This observation supports the statement by Dr. Chan that the border length, L, is twice the subregion perimeter.[7] The second limitation is that the model will only find the lowest cost pixels for a given area size. In processing digital imagery, this would correlate to the lowest gray value total for a given subregion size. It is rare that the imagery analyst will only want to find the darkest (lowest gray value) pixels in an image.

For the B&W model to become a digital processing tool, both the border length accounting and the pixel selection must be improved. With the present formulation, the analyst is required to know if a given pixel is on the image border or not. In the real world, the analyst will not know where his target is located (in the image), and he or she will expect the computer to find it. With reformulations to the basic model, these inherent weaknesses can be overcome. Improvements to the basic B&W model are examined in the next section.

[7]Note: The subregion perimeter does not include the external sides of the pixels on the image edge. Therefore, a single pixel subregion will have a minimum border length of six (edge pixel) or four (corner pixel).

3.2 *Examination and Modification of the Benabdullah & Wright Model*

The B&W model as originally designed is driven by a minimization objective function. This function is constrained by limits on border length and subregion size, contiguity requirements, nonoverlapping requirements and by limiting pixel assignment to only one subregion. As presently written, this model is of limited used to an imagery analyst processing multispectral imagery.[8] To increase the utility of this model, the constraints need to be examined and their relevance to digital imagery explained. Once this is determined, modifications to the model can be made. As will be seen, some of the constraints are of limited value and thus can be elimanted from the problem. This removal has the dual benefit of simplifying the problem (reducing the number of equations) and also increasing the fidelity of the model.

3.2.1 Examination of Constraints The B&W model is composed of two types of equations: the objective function and the constraints. The objective function drives the model but is limited by the constraints. It is important to understand each type of equation, in the context of digital imagery analysis, and what its implication is for the location of subregions within the imagery. In the B&W model, the constraints fine tune the objective function. For this reason, the constraints will be examined first and their significance explained. This will be followed by an examination of the objective function.

3.2.1.1 Border Constraint One of the constraints in the B&W model is a border constraint. In the description of the B&W model, this constraint was defined as shown in Equation 28.[9]

$$\sum_{i=1}^{n} \sum_{j \in T_i} s_{ij}(P_{ijk} + N_{ijk}) = L_k \quad \forall k \tag{28}$$

[8]Recall that this model was not originally designed to process digital imagery.
[9]For an explanation of each of these variables, refer back to Section 3.1.

This constraint allows the analyst to fix the border length for a given object size. At first, this would appear to be a very powerful constraint since the border length is fixed for a given area configuration. For example, a subregion of four pixels arranged in a square pattern would have a border length of 16.[10] The combination of a subregion of 4 pixels along with a border length of 16 will efficiently differentiate between this region and a subregion of 3 pixels also with a border length of 16. This combination of constraints will also differentiate between this square subregion of 4 pixels and any other subregion of four pixels not arranged in a square fashion. These examples are shown graphically in Figure 15.

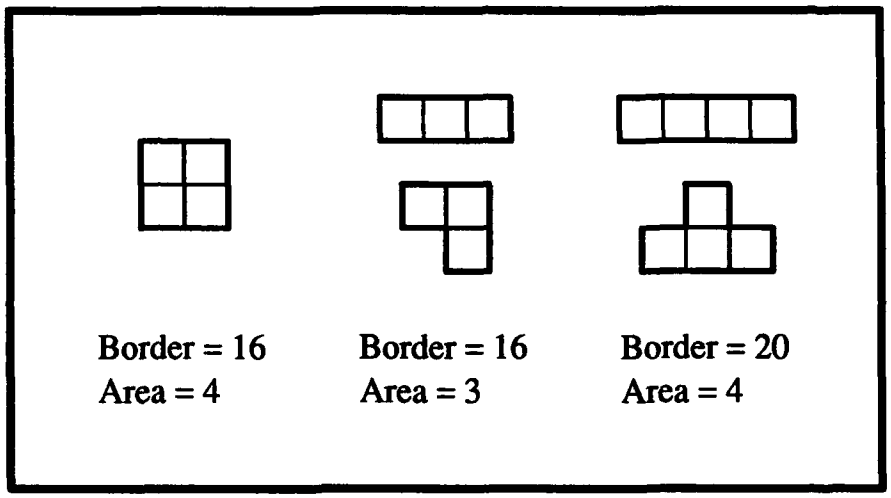

Figure 15. Border Constraint Examples

In the report on the B&W model, the writers only call out two methods to account for the border length (1:12). In reality, three methods are actually used to account for the border length of a pixel. If the pixel is a corner pixel, the length is four. If the pixel is an edge pixel, the length is six. Finally, if the pixel is an interior pixel, the length is eight. All of the above rules apply if the subregion size is one. If

[10] Recall, that the border length for this model is defined as twice the actual pixel border length. In this example, the actual pixel border length is 8, thus the subregion border length is defined as 16.

the subregion is greater than one, the border length is defined as twice the interior perimeter of the subregion. Since the analyst will not know *a priori* where the pixel is located on the image, a standard method for accounting of the border length is required.

In 1991 Captain Reed completed a Master's thesis entitled, *Binary Programming Models of Spatial Pattern Recognition: Applications in Remote Sensing Image Analysis.* In this thesis, he added a frame of pixels to the image data to help solve a multiple subregion allocation problem (26:35). Although he did not use this technique to help resolve the border length accounting problem, this method will work. With a frame of pixels around the image data, all the potential pixels of interest (all the pixels in the image) will now have a border of eight. With this standard method of accounting, the image analyst is not required to know where the target is located in the image. See Figure 16.

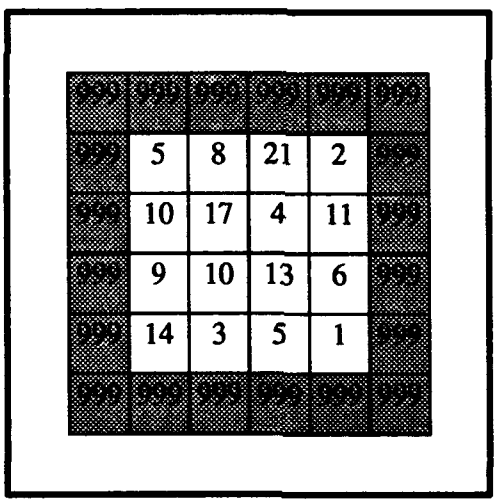

Figure 16. 4 × 4 Image Data With Frame

In this figure each of the image pixels is surrounded by four first order neighbors. To prevent the B&W model from selecting a frame pixel, the value of the pixel is set high. With a minimize objective function, the model will then bypass these

"high cost" frame pixels. In Figure 17, a frame has been added to the 3 × 3 pixel image that was used in previous examples.

A total of seven runs were made to test whether the frame pixels provided a standard accounting method for the border lengths. The results of these runs are shown in Table 3 below. A partial listing of the General Algebraic Modeling System (GAMS) output from these computer runs is included in Appendix B.[11]

Pixel Grey Values

999	999	999	999	999
999	11	5	6	999
999	8	13	2	999
999	5	1	1	999
999	999	999	999	999

Pixel Positions

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20
21	22	23	24	25

Figure 17. 3 × 3 Image Data With Frame

These results show that the addition of the frame results in a standard methodology for the accounting of subregion border lengths. Each image pixel now has a border length of eight.[12] With the accounting of the border length standardized, the image analyst no longer needs to know where the target is located in the image. If the analyst knows the size and shape of the target, the analyst can now input

[11] All the computer programs from this point in the thesis on were run on a VAX mainframe computer using GAMS. This switch from LINDO to GAMS was necessitated by the increased complexity of the problems. "GAMS is designed to make the construction of and solution of large complex mathematical programming models more straightforward for programmers and more comprehensible to users of models from other disciplines" (2:xiii).

[12] Recall that the border length is defined as twice the actual border.

Table 3. Results of Seven Runs With Frame

Model	*Input Constraints*				*Output*	
Run	*Subregion 1*		*Subregion 2*		*Pixels Selected*	
No.	*Size*	*Border*	*Size*	*Border*	*Subregion 1*	*Subregion 2*
1	1	8	1	8	18	19
2	1	8	3	16	14	17,18,19
3	2	12	5	24	12,17	8,9,14,18,19
4	3	16	3	16	8,9,14	17,18,19
5	1	8	4	20	8	14,17,18,19
6	1	8	5	24	8	9,14,17,18,19
7	1	8	6	20	17	8,9,13,14,18,19

these constraints into the model.[13] With the minimize objective function, the model will select the subregion with the lowest gray values that satisfy the user-defined constraints. In the above runs, the minimum subregions are selected (that meet the user-defined constraints) and this can be verified by inspection. Although the model now has a standard methodology to account for the border lengths, the model is far from operational. The model only selects the lowest gray values pixels due to the minimize objective function. It would be rare that the analyst only wanted to find the lowest *cost* pixels.

When the border length constraint is applied to digital imagery, it has limited utility. The model's underlying assumption tied to this constraint is that the object or subregion of interest is aligned with the grid lines. As long as the region is aligned with the the grid lines, the border length remains fixed. However, if the object is rotated, the border length will increase. For a rectangular subregion, this increase in border length will reach a maximum at 45 degrees. For an example of this phenomena, see Figure 18. In this figure, the grid represents a 10×25 digital image with an individual pixel depicted by a single small square.

[13]The size and shape of the object must first be converted into pixel size and shape before this data can be input into the model.

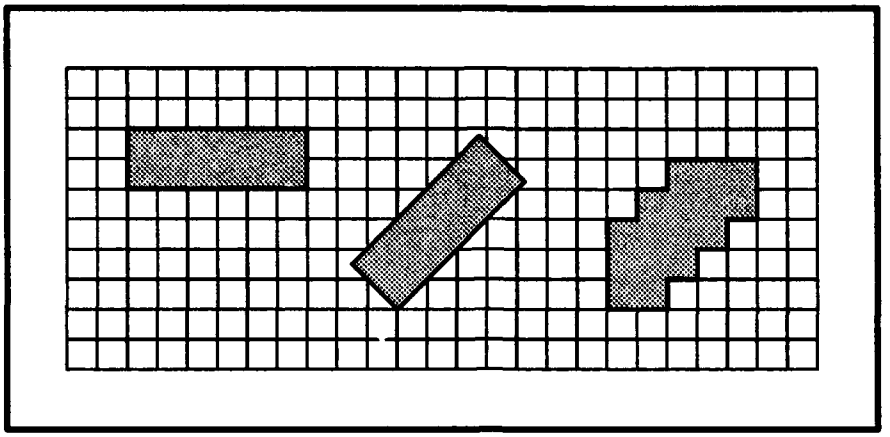

Figure 18. Effect of Rotation on Border Length

In Figure 18 the rectangular subregion of 12 pixels has a border length of 32 when it is aligned with the grid lines. This same subregion is rotated 45 degrees in the middle figure. Since the pixels are square in dimension, they can not replicate the diagonal lines. Instead, a stair step effect occurs as shown in the third subregion. In this subregion, the area and border has been approximated using the following rule. If a pixel is 50 percent or more covered by the subregion area, then it is included in the subregion area. The result of the rotation is that the area increased to 16 pixels and the border length increased to 40. It is important to note this change was caused only by the rotation of the object and not by a change in the subregion itself.

One may see the obvious applicability to moving targets, but may not see how this applies to stationary targets such as buildings. An object can appear to be rotated if the pixel grid itself is rotated. This is certainly possible with the orbital viewing capabilities of a satellite sensor. Usually the pixel grid (formed by the sensor array) will not rotate and the building (or any other object) will not be aligned with the grid. If the object of interest and the grid are not aligned, the object will usually display this characteristic stair step border.[14]

[14]The occurrence of the stair step border will depend on the size of the object relative to the pixel size.

When remote sensing is accomplished by satellites, the objects of interest will rarely be aligned with the pixel grid. For this reason, the use of the border length constraint is of little utility. If this constraint is used, a range of border lengths would increase the utility of the model. However, even this extension is limited in its application.

In the example used above, a pixel was assigned to the subregion if it was 50 percent or more covered by the subregion. This *rule of thumb* is generally accurate, but it is not entirely true. A *mixel* is formed when a pixel spans the boundary between two or more types of subregions. A mixel thus contains the weighted spectral response from two or more different objects. For this reason, mixels are difficult to classify based on their spectral response alone. The sensor does not differentiate between these different areas, but rather records the combined spectral response from this area. As one object begins to dominate the mixel, the spectral response will begin to approach the characteristic spectral range of the dominating object. If the border between an object and its surroundings is difficult to distinguish, then the border length will also be difficult to specify for a given area. In Figure 18, one can see eight mixels that are 50/50 in composition between the object of interest and the surrounding area. Therefore, the border outlined on the rotated object may not be entirely correct. Given this analysis, one begins to appreciate the lack of utility for the border constraint. If the border constraint is used, one should use a range of lengths for a given size target.[15] The analyst should then systematically try each of these values in the model until the object is found. If no answer results, the size of the object may be incorrect or the object may not be in the image.

3.2.1.2 Area Constraint Another constraint in the B&W model is the area constraint. With this constraint, the analyst is able to specify the a priori size

[15] Recall earlier in this chapter the discussion of the acceptable range of border lengths for a given sized object.

of the subregion to be found in the digital image. The mathematical formulation of this constraint is given in Equation 29 below.[16]

$$\sum_{i=1}^{n} x_{ik} = M_k \quad \forall k \tag{29}$$

At first, this constraint also appears to be a powerful one in helping the analyst to find the area of interest. However, further examination of this constraint shows that it is limited in its digital imagery application. It appears that this constraint allows the analyst to bypass all those *like-type* subregions that are either smaller or larger than the specified area size.[17] In fact, this observation is only partially true. If the area constraint is set at five pixels, no area in size under five pixels will be found. On the other hand, this constraint does not restrict the model from finding a portion of a like-type region of larger size. This fact is illustrated in Figure 19.

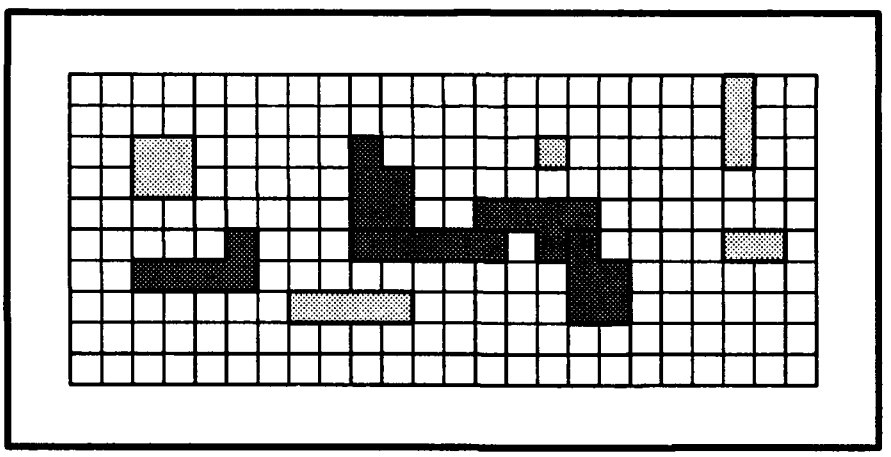

Figure 19. Area Constraint Examples

[16]For a explanation of each of these variables, refer back to Section 3.1.

[17]By *like-type*, I mean the same land cover type or same type of object. For example, if the analyst is looking for lakes, not only will lakes be found, but also all bodies of water such as rivers, oceans, swimming pools, etc. What can actually be found will depend on what water features are in the image and the spatial and spectral resolution of the image.

In Figure 19 all the subregions represent the same type of object. For the sake of illustration, suppose they are all bodies of water. The lighter bodies of water will all be passed over by the model since they do not meet the size constraint.[18] If the analyst was looking for the L-shaped body of water, he or she would not be able to find it based on size alone. Consider the large body of water in the middle. This region illustrates how the 20-pixel-sized body of water could be broken into four acceptable answers. The result of dividing the area as described is that there are five possible solutions. The division of the 20-pixel-sized body of water shown is not the only acceptable division. There are more ways to divide the region into other subregions of 5-pixel size and thus, more possible solutions will result. Even if the border length constraint is included, this will not allow the model to uniquely identify the L-shaped subregion. The border length of the L-shaped subregion is 24, but this is the same border length for two of the 5-pixel area subregions shown in the larger body of water. Thus, the possible number of solutions has been reduced to three. With this reduction in the number of possible solutions, one may be lulled into reconsidering the border length constraint. However, one should recall how the border length is not an effective constraint with satellite digital imagery.

The above example illustrates an important subtlety that the analyst needs to consider when using the area constraint. The area constraint allows the analyst to find subregions equal to or greater than the size constraint. In other words, a subregion equal in area to the size constraint can be found, but this is not necessarily the natural size of the subregion. For example, if the model selected a subregion of five pixels out of the major body of water, this does not tell the analyst the true size of the 20-pixel-sized body of water.

[18]In using the phrase "lighter bodies of water," I am not referring to the spectral response of the water, but rather the shading of the graphical depiction.

Given this examination of the area constraint, we see that it is not very effective.[19] This constraint will eliminate all subregions of size smaller than the desired area but it will not eliminate subregions larger than the constraint. Analysis of the area constraint shows that it must be used with some forethought.

3.2.1.3 Contiguity Constraint An important constraint in the B&W model is the contiguity constraint. With this constraint, the analyst ensures that the pixels selected to form one defined area are all connected together. Mathematically, this constraint is enforced with two equations:[20]

$$\sum_{i=1}^{n} x_{ik} = M_k \quad \forall k \tag{30}$$

$$\sum_{i=1}^{n} \sum_{j \in T_i} s_{ij}(P_{ijk} + N_{ijk}) = L_k \quad \forall k \tag{31}$$

With this constraint, the B&W model is prevented from randomly selecting pixels in the image. Rather, the pixels selected must all be connected together and also meet all of the other constraints. By the term *connected*, I mean first order neighbor connected. While this is a valuable constraint, it also has its limitations. Consider the following. For each interior pixel, there are a total of eight neighbors.[21] Four of these neighbors are first order and thus, are possible candidates for selection into the subregion. However, the second order pixels (diagonally connected pixels) are not candidates for selection. The second order neighbors will not be incorporated into the subregion unless one or more of the first order neighbors are first selected. For an example of this, look at Figure 20.

[19]This measure of less than 50 percent effectiveness is based on the following assumption: the number of subregions larger and smaller than the area constraint are equal. The magnitude of this assumption depends on what the analyst is looking for and what the image contains. Thus, the magnitude of this assumption will vary from case to case.

[20]For an explanation on each of these variables, refer back to Section 3.1.

[21]By the phrase *interior pixel*, I mean a pixel not located on the border of the image.

2	1	2
1	X	1
2	1	2

1	2	3
4	X	6
7	8	9

Figure 20. Illustration of First and Second Order Neighbors

In this figure, the subregion of nine pixels on the left illustrates the concept of first and second order neighbors. The first order neighbors are denoted by 1's and the second order neighbors are denoted by 2's. Whether a pixel is a first or second order neighbor depends on its location with respect to the pixel currently under consideration. The pixel under consideration is marked with an X in this figure. The limitation of the model is illustrated in the second subregion of nine pixels. Pixels 1, 3, 7, and 9 can not be added to the X pixel unless pixel 2, 4, 6, or 8 is first added to the middle pixel. For example, if pixel 6 is added to the middle pixel, then pixels 2, 3, 4, 8, and 9 are candidates for incorporation into the subregion. Notice that pixels 1 and 7 were excluded since they are still second order neighbors even with the incorporation of pixel 6. If pixel 4 is added to the X pixel, then pixels 1 and 7 are candidates for incorporation. With this model, diagonal appendages will not be considered for incorporation into a subregion if they are all second order neighbors. For example, see Figure 21 below.

In this figure, the model can not select any of the diagonal appendages of the subregions. The reason for this nonselection is that they are all second order neighbors. An extreme example is the right subregion. The B&W model will not even detect this subregion if the area constraint is set greater than one pixel. Notice

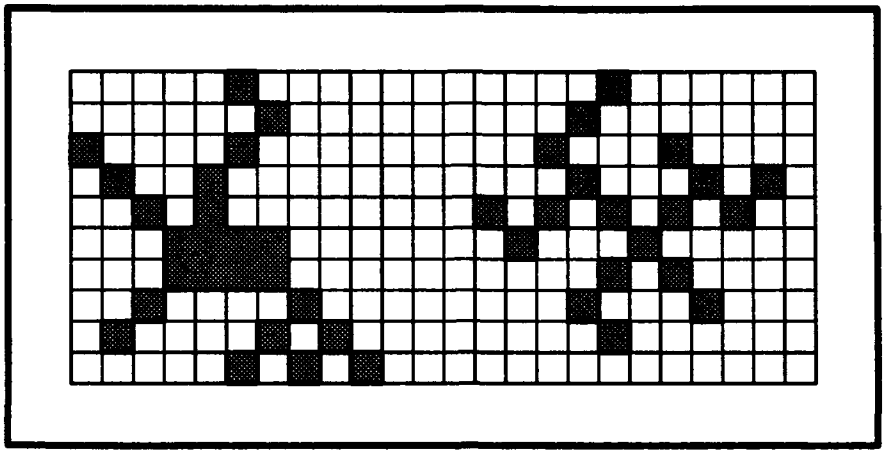

Figure 21. Subregions with Diagonal Appendages

that all of the diagonals are of one pixel width. If the diagonal appendage is greater than one pixel width, then all the diagonal pixels are candidates for incorporation into the subregion. They are possible candidates because they have now become first order neighbors. See Figure 22 for an illustration of this concept. All of the diagonal appendages of the central subregion are possibilities for incorporation in this figure.

The significance of this limitation to the imagery analyst will depend on two factors. The first factor is the spatial resolution of the sensor. As the spatial resolution increases, the pixel size decreases. As the pixel size decreases, the number of single width diagonal appendages will decrease. Thus, as the spatial resolution improves, the ability of the B&W model to find diagonal type subregions and diagonal appendages also improves. There is a direct correlation between the two. The second factor to consider is what the analyst is looking for. For example, if the analyst is attempting to locate river systems, this model would not be very effective. The reason for this is twofold: 1) river systems have characteristic branching resulting in thin lines and 2) they typically do not travel in straight lines. When the river is relatively large in comparison to the pixel size this is not a problem. However, when

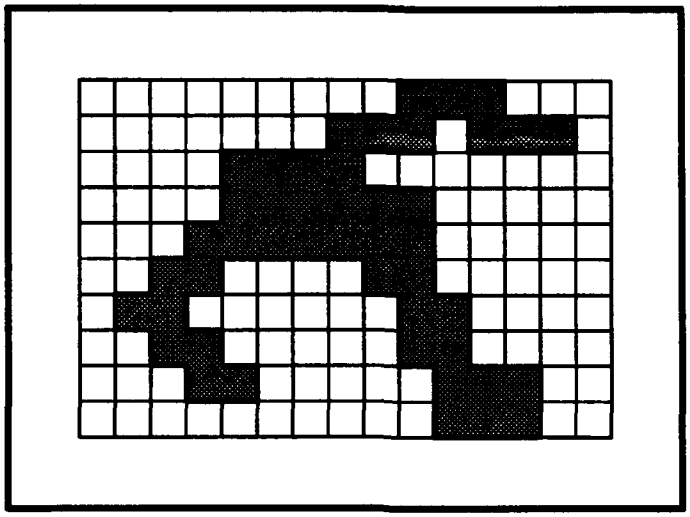

Figure 22. Subregions with Diagonal Appendages of Width Greater Than One

the river branches approach the size of an individual pixel in width, this limitation needs to be seriously considered. This limitation applies not only to river systems but also to any other objects or subregions that display the same characteristics as described above (e.g. a road at 45 degrees). If the analyst is not looking for objects or subregions that branch or have diagonal appendages, then the limitation is of little concern. One should recall that a straight region can appear diagonal when not aligned with the pixel grid. For this reason, the analyst needs to be careful not to arbitrarily disregard this limitation when using this constraint.

3.2.1.4 One Pixel - One Subregion Constraint Another constraint in the B&W model is the one pixel, one subregion constraint. This constraint is mathematically defined as follows:[22]

$$\sum_{k=1}^{K} x_{ik} \leq 1 \quad \forall i \tag{32}$$

[22]For an explanation on each of these variables, refer back to Section 3.1.

The model limits the sum of a particular pixel to one, so that it can only be used for one subregion. With this constraint, there is no double accounting. In other words, a pixel will not be assigned to two or more subregions at the same time. This constraint is straight forward and requires no further explanation.

This constraint is directly applicable to the analysis of digital imagery. It performs the same function as above when looking for subregions in digital imagery. This is a necessary constraint that will be retained in my implementation of the model.

3.2.1.5 Nonoverlapping Constraint A constraint related to the *One Pixel - One Subregion Constraint* is the nonoverlapping constraint. This constraint is mathematically defined as follows:[23]

$$x_{ik} - x_{jk} - P_{ijk} + N_{ijk} = 0 \quad \forall i, j \in T_i, k \tag{33}$$

This constraint prevents overlapping of pixels between subregions. It also defines the binary variables P_{ijk} and N_{ijk} which are necessary to account for the border length. Therefore, this constraint is needed for digital image processing if more than one subregion is being located or if the border length constraint is used.

3.2.2 Examination of Objective Function The driving equation for the whole B&W model is the minimize objective function. With this equation, pixels are selected on the basis of least cost. In addition to least cost, the pixels selected must also meet the constraints of border length and size. With this formulation, the model is of little value to an imagery analyst. In the real world, the analyst does not always attempt to locate the lowest *cost* pixels.[24] Rather, the analyst may look for targets that encompass the entire range of gray values. If the B&W model is run on the data

[23]For an explanation on each of these variables, refer back to Section 3.1.

[24]For this report, the *cost* of the pixel is equivalent to the gray value of the pixel. In eight-bit digital imagery, the range of gray values is from 0 to 255.

shown in Figure 23, the expected result, for a subregion of size four or smaller, would be the combination of pixels eight through eleven.[25] The results of this computer run are shown in Table 4. More detailed results are shown in Appendix C.

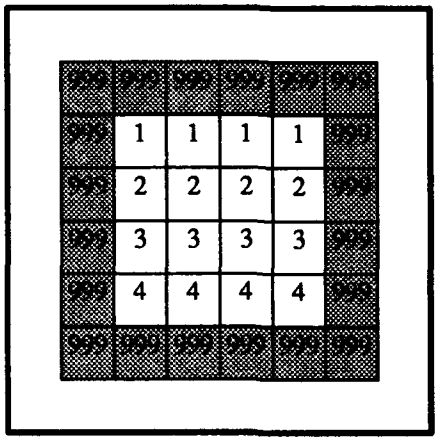

Figure 23. Stratified Digital Image

Table 4. Location of a Four Pixel Subregion With Min Objective

Input Constraints		*Output*
Size	*Border*	*Pixels Selected*
4	20	8,9,10,11

If the objective function is changed to a "maximize" objective function and the same problem is rerun, the expected result would be the combination of pixels 26 through 29.[26] The results for the location of a four pixel region is shown in Table 5. More detailed results are shown in Appendix C.

Although the data used in these two problems are not representative of what would normally occur in nature, the limitations of the present model are clearly

[25]Recall that the basic B&W model is driven by an minimization objective function.

[26]If the objective function is changed to maximize, the frame surrounding the image pixels must change. If the arbitrarily large pixel values are not changed, the model will attempt to incorporate these nonexistent image pixels into a subregion. For this computer run, the frame pixels were changed to a value of zero.

Table 5. Location of a Four Pixel Subregion With Max Objective

Input Constraints		*Output*
Size	*Border*	*Pixels Selected*
4	20	26,27,28,29

shown. The model can be used to find subregions of the lowest gray values or the highest gray values. A modification is needed to find subregions composed of pixels at the gray value limits (0 and 255) and everywhere in between. One method that would allow for this flexibility in the model would be to set ranges on the sum of the gray values for a subregion. A similar concept was used by Benabdullah and Wright in the dividing of areas according to populations (1:19). While this concept was mentioned in the cited reference, it was not fully developed nor applied to digital imagery.

The present model with either a maximize or minimize objective function is of little value to the imagery analyst. With this implementation of the model, the analyst would only be able to find the highest or lowest cost pixels. Unless the analyst is only interested in finding the lightest or darkest areas, the objective function and thus the whole model is of little utility. Modification to the objective function as well as to the constraints must be made to transform the B&W model into one that can be effectively used by an imagery analyst. These modifications will be explained in the next section.

3.2.3 Modification of Constraints Given the above analysis, it is evident that modifications to the B&W model are necessary. Without these changes, the imagery analyst can not find the subregions of interest. Modifications to the constraints are first explored followed by changes to the objective function. The overall goal of these modifications is to increase the utility of the model for the imagery analyst.

3.2.3.1 Pixel Selection Constraint One of the first modifications needed by the B&W model is a method which allows the selection of pixels with any gray value. Currently, the model can only select the highest or lowest gray value subregions. One method to improve the selection of pixels is to set an upper and lower bound (in terms of gray value total for the subregion) on the subregion of interest. A second method for improving the selection of pixels is to set limits on the acceptable gray value for each pixel considered for incorporation into the subregion. Both methods are explored in depth in the following sections.

Subregion Bounds If the image analyst has *a priori* knowledge about what he or she is looking for, the spectral signature of the target will probably also be known. The spectral signature of a target is the reflected and emitted radiation from the target. Each object has a characteristic response that can be used to differentiate one object from another. It is this feature that creates different gray values on the image after it has been developed. For this reason, homogeneous regions tend to have pixels of similar gray value. Therefore, a corn field may have pixel values centered around 140 and an oak forest may have pixel values centered around 100.[27]

Using this knowledge, the B&W model could be modified to find a 100-pixel-sized oak forest and 100-pixel-sized corn field. If the oak forest is designated as Subregion Number One, an additional constraint can be added for the sum of this region. Assuming a totally homogeneous oak forest, the sum of the 100 pixels would be 10,000. Likewise, the sum of the 100 pixels for the corn field would be 14,000. If these two constraints are added to the model, both areas should be found (provided there is a 100 pixel subregion of both a corn field and a oak forest).

[27]These gray values are not the actual values of a corn field and oak forest. These values are used for illustrative purposes. The actual value would depend on the spectral band in which the two targets were imaged, the imaging conditions, and condition of the targets.

In reality, the gray value for a homogeneous region is not an exact value but rather a small range of gray values. Using the corn field example, the range of gray values may be from 135 to 145. Thus, a better constraint for the location of the 100-pixel-sized corn field would be to set a lower and upper bound for the subregion of 13,500 and 14,500.[28]

Computer Runs & Analysis To test if this method works, several computer runs of the model were made. With the data shown in Figure 23, both the border length improvement and the sum of the subregion constraints were used. The first run attempted to locate two subregions. One subregion consisted of three pixels and the sum of the gray values was set to three. The second region consisted of four pixels and the sum of the gray values was set to 16. The equations for this problem are given below.[29] Note Equations 42 and 43 are the new subregion total constraints.

$$\text{Minimize } Z = \sum_{k=1}^{2} w_k \left[\sum_{i=1}^{36} c_i x_{ik} \right] \tag{34}$$

subject to:

Subregion size constraints

$$\sum_{i=1}^{36} x_{i1} = 3 \quad \forall k = 1 \tag{35}$$

$$\sum_{i=1}^{36} x_{i2} = 4 \quad \forall k = 2 \tag{36}$$

[28]Once again, these are not the actual gray values for a corn field. These values are used for illustrative purposes.

[29]In the first equation, the weight for each subregion, w_k, is set to one. The significance of setting the weight to one is that each subregion is equally weighted in the objective function. In the border length constraint, the variable s_{ij} is also set to one. This is a reasonable initialization since most pixels have equal lengths and widths. With this setting, however, the target will need to be measured in pixel units rather than meters or feet. To illustrate this point, consider a 100m × 200m target area. If the pixels measure 20m on a side, the size of the object will be 5 × 10 pixels in dimension and the border length will be 60 (2 × a subregion border of 30).

Contiguity constraints

$$x_{ik} - x_{jk} - P_{ijk} + N_{ijk} = 0 \quad \forall i, j \in T_i, k \tag{37}$$

Each pixel only assigned to one subregion

$$\sum_{k=1}^{2} x_{ik} \leq 1 \quad \forall i \tag{38}$$

Border length constraints

$$\sum_{i=1}^{36} \sum_{j \in T_i} s_{ij}(P_{ij1} + N_{ij1}) = 16 \quad \forall k = 1 \tag{39}$$

$$\sum_{i=1}^{36} \sum_{j \in T_i} s_{ij}(P_{ij2} + N_{ij2}) = 20 \quad \forall k = 2 \tag{40}$$

Setting x, P, and N to binary variables

$$x_{ik}, P_{ijk}, N_{ijk} = 0, 1 \quad \forall i, j \in T_i, k. \tag{41}$$

Grey value totals per subregion

$$\sum_{i=1}^{n} c_i x_{i1} = 3 \tag{42}$$

$$\sum_{i=1}^{n} c_i x_{i2} = 16 \tag{43}$$

In the second run two subregions will again be found, but the input will change. In this run, the first subregion will consist of eight pixels with an upper gray value sum of 32 and a lower bound of 24. The second region will consist of two cells and have an upper bound of four and a lower bound of two. Instead of two addition equations, as in the previous problem, four additional equations are needed. These four equations are given below. The results of these two runs are shown in Figure 24.

Grey value bounds per subregion

$$\sum_{i=1}^{n} c_i x_{i1} \leq 32 \tag{44}$$

$$\sum_{i=1}^{n} c_i x_{i1} \geq 24 \tag{45}$$

$$\sum_{i=1}^{n} c_i x_{i2} \leq 4 \tag{46}$$

$$\sum_{i=1}^{n} c_i x_{i2} \geq 2 \tag{47}$$

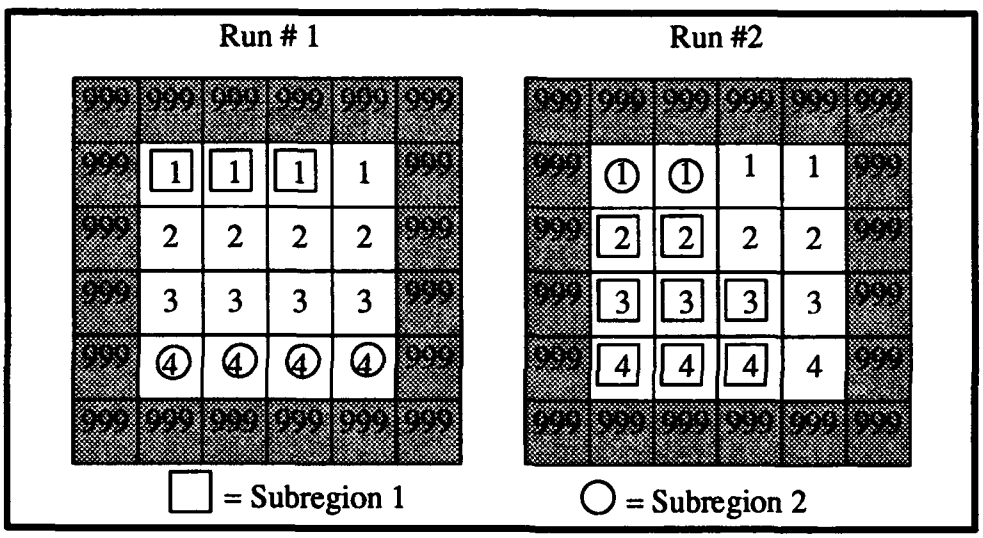

Figure 24. Results of Runs #1 and #2

With these new constraints, the analyst can dictate to the model what type of pixels to consider for incorporation into a subregion. The tighter the upper and lower bounds for a subregion, the more the model is restricted. To illustrate this point, consider the two runs. In the first run, the sum of each subregion was set. With this constraint, the pixel values are not allowed to vary. If there is a subregion of four pixels that sum to 16 (with a border length of 20), the model will find it. For the given data set, the model only has one option and that is to select the four

pixels each with a value of four.[30] As was mentioned earlier, homogeneous regions usually vary somewhat in their spectral response. Therefore, while a specific required total of a subregion will cause the model to be very exact in what subregion it will find, it is not always operationally valid. In the second run, this variance in the spectral response is incorporated into the model by allowing each subregion to have an upper and lower bound. See Equations 44-47. With these constraints, the model selected a subregion composed of the one-valued pixels and the other subregion was composed of two, three, and four-valued pixels. Seeing this result, one could suggest that the model is not restrictive enough. As was mentioned earlier, the model can be made more restrictive by narrowing the upper and lower bounds. This premise was validated by rerunning the second run and raising the lower bound to 26. With this run, the subregion of one-valued pixels was still selected but the other subregion was composed exclusively of three and four-valued pixels. This method of changing the upper and lower bounds cannot be arbitrarily applied to digital imagery. The reason for this is that the spectral response for a given land cover type is set. If the spectral range is shortened, pixels of a given land cover type will be excluded from consideration for incorporation into a subregion. For more details on all three GAMS runs, see Appendix D.

Pixel Bounds A similar but better constraint to improve the pixel selection is to set bounds on the individual pixels. With this method, undesired pixel types are not allowed into a subregion. In other words if the analyst is looking for water, he or she can set an upper and lower bound for each pixel equal to the spectral range for water. Mathematically, this constraint is given as

[30]This line of thinking also applies to the other subregion specified in Run Number One — subregion of size three, border length of 16, and pixel gray value sum of three.

$$x_{ik} = \begin{cases} \text{may be 1}, & \text{if pixel gray value is between spectral bounds} \\ 0, & \text{otherwise.} \end{cases} \tag{48}$$

where x_{ik} is a binary variable that indicates if a given pixel is considered for assignment into subregion k. Another mathematical representation of this constraint is as follows:

$$\mathbf{lsb} \leq \quad \text{if } c_i \quad \leq \mathbf{usb} \tag{49}$$

$$\text{then, } x_{ik} \text{ can be selected}$$

$$\mathbf{lsb} \geq \quad \text{if } c_i \quad \geq \mathbf{usb} \tag{50}$$

$$\text{then, } x_{ik} = 0$$

where **lsb** is the lower spectral bound and **usb** is the upper spectral bound for a particular land cover type.

With these equations, only pixels of the desired land cover type are considered for incorporation into a subregion. For example if the analyst is looking for water, only pixels that fall within the water spectral range are candidates for incorporation into the water subregion.[31] All the other land covers are set to zero and thus are not considered for incorporation into the subregion. This method is superior to the subregion bounds method because that method only ensures that the mean gray value falls within the required range. The subregion bounds constraint does not prevent an undesired pixel type from being included in the subregion as long as the total sum for the subregion falls within the designated bounds. Another advantage of the pixel bounds method is that non-candidate pixels are assigned a value of zero. This means that these pixels will not be considered for incorporation into a

[31] Note: Just because a pixel is a candidate for incorporation does not mean it will be selected by the model. The candidate pixel has to meet all the other constraints and also maximize or minimize the objective function.

subregion. This constraint should assist the model when it performs the branch and bound to find a noninferior solution. The model is assisted by eliminating the majority of the pixels for pixel selection. Only the pixels with a gray value in the specified spectral range will be considered for selection.

With the pixel bounds constraint, the model is now becoming a tool the imagery analyst can use to find specific types of land cover. Each subregion to be found will have a separate constraint specifying the type of pixels to be considered for incorporation into the different subregions. Based on the analysis of the two pixel selection methods, I will incorporate the pixels bounds method into my formulation of the model.

3.2.4 Modification of Objective Function The original objective function of the B&W model is a minimization one. In Section 3.2.2, it was shown that the model can also be run with a maximize objective function. The implications of combining the minimization or maximization objective function with the other modifications can now be discussed. One new formulation of the model is to combine the objective function with the pixel bounds, contiguity, and one-pixel-one-subregion constraints and delete the border length and area constraint. With this new model, subregions of desired land cover type could be found, but this would not tell the whole story. With a maximization function, the largest contiguous subregion of desired land cover type will be found. The reason why the largest subregion is found is that the size constraint is deleted. Recall that using a size constraint is of limited use because it can only find areas of an exact size. Even though this formulation will give a *correct answer*, the answer will be biased.[32] The answer is biased in the sense that the lighter (higher gray value) pixels are preferred over the darker (lower gray value) pixels. A good assumption for the gray values of a particular land cover type is that they are Gaussian (or normally) distributed (3). If this is a given, then the model

[32] By the phrase *correct answer*, I mean a result where all the selected pixels are contiguous and each subregion is composed of correct pixel type.

is not picking those pixels that best represent the desired land cover type. Rather, the model is biased to select those pixels that lie on the fringes of the designated spectral range.

If the objective function is changed to a minimization one, the model will operate differently. The most noticeable difference is that the model will select no pixels if the same constraints are used as described above. Without an area constraint, the minimize objective function will minimize the **Z** value by selecting no pixels.[33] Therefore, a size constraint is needed to run this model with a minimize objective function. With this additional constraint, a correct answer can be found but the answer will again be biased. This time, the model is biased to pick the darker (lower valued) pixels over the lighter (higher valued) pixels. While the answer is correct, it is not the best representation of the particular land cover type. A method to correct for both the upper and lower biases is needed to improve the pixel selection process. One method for accomplishing this correction is discussed below.

If a Gaussian distribution for the gray values of a particular land type is assumed, a method to optimize the selection of pixels with the gray value mean is needed. In other words, a method to bias the selection of pixels that are most representative of the particular land cover type is needed. If a penalty is imposed for deviations from the mean gray value then this would correct for the upper and lower bound selection biases. A method to accomplish this goal is defined mathematically in Equation 51

$$\mathbf{Z} = \text{Max} \sum_{k=1}^{K} \sum_{i=1}^{n} \frac{1}{|(\overline{x}_k - c_i x_{ik})| + 1} \tag{51}$$

where k indicates the number of subregions to be found, $\overline{x}_k$ is the mean gray value for subregion k, c_i is the gray value for pixel number i, and x_{ik} is a binary variable that indicates if a given pixel, i, is assigned to subregion k.[34] The one is added (to

[33]In operations research programming, the result of the objective function is typically designated by **Z**.

[34]Note another objective function which increases the penalty for deviations from zero is $\mathbf{Z} = \text{Max} \sum_{k=1}^{K} \sum_{i=1}^{n} \frac{1}{(\overline{x}_k - c_i x_{ik})^2 + 1}$. With this objective function, the penalty is increased geometrically

the absolute value of the differences) in the denominator to prevent division by zero. This equation along with two others will be discussed next.

One may question why the equation is not $\mathbf{Z} = \text{Max} \sum_{k=1}^{K} \sum_{i=1}^{n} |(\overline{x}_k - c_i x_{ik})|$. If this equation is the objective function, then the model will attempt to maximize the difference between the gray value of the pixels selected and the mean gray value. Therefore, the model would again be biased to select pixels at both extremes of the subregion spectral range. A correct answer will result with this objective function but the model is biased not to select the best representative pixels.

If the objective function described in the above paragraph is changed to a minimization one, the model will select no pixels. The $\mathbf{Z}$ value would indeed be minimized since the value would be zero. However, a correct answer can be obtained if the area constraint is added. With this formulation, the model will attempt to select pixels closest to the mean gray value for the desired subregion. This result supports the goal stated above but the selected subregion may or may not be the natural size of the subregion.

The objective function in Equation 51 also supports the goal of implementing a selection bias that favors pixels close to the mean gray value. Pixels at the upper and lower bounds contribute significantly less to the objective function than those that are close to the mean. The relative contribution of various pixels to the objective function is shown in Table 6.

Looking at the values in Table 6 shows that the model is biased to select the pixels that are closest to the mean gray value for a given land cover type. The model will select the area most representative of the specified type if an area constraint is used in conjunction with this objective function.

In summary, all of the three modified objective functions described have varying degrees of utility. The model is biased to select pixels at the extremes of the

as a pixel gray value varies from the mean. I did not use this objective function since the scope of this thesis is confined to linear program techniques.

Table 6. Relative Contribution of Pixels to Objective Function

Pixel Difference $(\overline{x}_k - c_i x_{ik})$	*Absolute Value* $\lvert(\overline{x}_k - c_i x_{ik})\rvert$	*Contribution to Objective Function*
0	0	1
$\pm$ 1	1	1/2
$\pm$ 2	2	1/3
$\pm$ 3	3	1/4
$\pm$ 4	4	1/5
$\pm$ 5	5	1/6
$\vdots$	$\vdots$	$\vdots$

designated spectral range with the first objective function, $\mathbf{Z} = \text{Max} \sum_{k=1}^{K} \sum_{i=1}^{n} \lvert(\overline{x}_k - c_i x_{ik})\rvert$. In general if an area constraint is used with this objective function, the model will find the area least representative of the designated land cover type. With the second objective function, $\mathbf{Z} = \text{Min} \sum_{k=1}^{K} \sum_{i=1}^{n} \lvert(\overline{x}_k - c_i x_{ik})\rvert$, the model is biased to select pixels close to the mean gray value for a specified land cover type. No pixels will be found with this objective function unless an area constraint is added. An area most representative of the specified land cover type will result with this addition. The model is again biased to select pixels close to the mean gray value for a specified land cover type with the third objective function, $\mathbf{Z} = \text{Max} \sum_{k=1}^{K} \sum_{i=1}^{n} \frac{1}{\lvert(\overline{x}_k - c_i x_{ik})\rvert + 1}$. An area most representative of the specified land cover type will be found if an area constraint is used with this objective function. Of these three objective functions, the second and third have the most utility when applied to digital imagery.

3.3 Combining of Single Channel Images

Once a single channel has been processed (spectral and spatial patterns identified), a methodology is needed to combine the single channels into a composite image. By combining the images, a better segmentation of the object of interest should result. The reason for this increase in accuracy is that more information has

been used to identify (or recognize) the object of interest. It would not be uncommon for the model to select a different subregion size and border length because of the different spectral response of the object of interest in each imaging band. One method that can be use to combine processed single channel images is *multicriteria optimization.* The concept of multicriteria optimization will first be explained and then its application to digital image processing will follow.

3.3.1 Multicriteria Optimization Description When a single channel image has been processed, it is necessary to combine it with the other single channels images. Each channel is processed against a different objective. Therefore, the answer in one image may not be the same as in another image, although the images are of the same geographic area. If the images were simultaneously processed with all the different objectives included, it is possible to calculate a composite result. This method also requires that the constraints for each image also be the same. In short, the focus of multicriteria optimization is to eliminate the inferior solutions rather than determine a single best solution. The result of this process is a small subset of feasible solutions that require further analysis. This subset is formally known as the set of *noninferior solutions*[35] (10:174). This subset is sometimes called the *N-set* (4). One method of multicriteria optimization is by multiple objective linear programming (MOLP).

The general formulation for a multiple objective linear program is as follows (33:138):

$$\max \quad (\mathbf{c}^1\mathbf{x} = z_1)$$

[35]The term *noninferior solutions* is sometimes referred to as dominate solutions, Pareto optimal solutions, nondominated solutions, admissible solutions, or the production feasibility frontier. Although the term *noninferior solutions* is not elegant, it is preferred since it is both descriptive and accurate. The term *dominate solutions* is misleading since no solution in the noninferior dominates over any other in the set (5:4-5), (10:175).

$$\begin{aligned} \max \quad & (\mathbf{c}^2\mathbf{x} = z_2) \\ & \vdots \\ \max \quad & (\mathbf{c}^k\mathbf{x} = z_k) \\ \text{subject to} \quad & \mathbf{x} \in S \end{aligned} \tag{52}$$

or

$$\text{“max”}(\mathbf{C}\mathbf{x} = \mathbf{z} | \mathbf{x} \in S) \tag{53}$$

where

k is the number of objectives,

$\mathbf{c}^i$ is the *gradient*; objective function coefficients of the ith objective,

z_i is the *criterion value*; objective function z-value of the ith objective,

S is the feasible region,

"max" indicates maximization of all objective simultaneously,

$\mathbf{C}$ is the $k \times n$ criterion matrix; matrix of objective function coefficients whose rows are the gradients of the gradients $\mathbf{c}^i$ of the k objective functions, and

$\mathbf{z}$ is the *criterion vector*; objective function vector, $\mathbf{z}$-vector.

Multiple objective problems rarely have a single point which maximizes all objectives simultaneously. For this reason, we want to maximize each objective to the "greatest extent possible" (33:138-139). To better understand the process of multicriteria optimization, it is necessary to introduce the concept of **X**, **Y**, and **Z** space. **X** is commonly referred to as the *decision space*. **Y** is called the criterion or *outcome space* and **Z** is the *preference structure* (5:1).

3.3.1.1 Multicriteria Optimization Example One of the best ways to explain the **X**, **Y**, and **Z** spaces is by way of an example. These spaces are used

to arrive at a multicriteria optimization solution. Given the multicriteria problem below, the utilization of the three spaces can be shown (4).

$$\begin{aligned} \text{Max } f_1(x_1, x_2) &= 4x_1 - x_2 \\ \text{Max } f_2(x_1, x_2) &= -2x_1 + 5x_2 \\ \text{subject to } 2x_1 + 3x_2 &\leq 12 \\ x_2 &\leq 3 \\ 3x_1 - x_2 &\geq 0 \\ x_1, x_2 &\geq 0 \end{aligned} \tag{54}$$

A proven method for solving this MOLP is to subordinate one of the objectives (or criterion functions) into the constraints (5:6). An equivalent representation of Problem 54 is given below (4).

$$\begin{aligned} \text{Max } f_1(x_1, x_2) &= 4x_1 - x_2 \\ \text{subject to } \mathbf{x} &\in \mathbf{X} \\ f_2(x_1, x_2) &= -2x_1 + 5x_2 leqr_2 \end{aligned} \tag{55}$$

where r_2 is the satisficing level for f_2. "By minimizing and maximizing f_2 over $\mathbf{X}$, the feasible region defined by the original constraint set, we find" that f_2 is bounded as follows (5:6):

$$-12 \leq f_2(x) \leq 13 \tag{56}$$

If the constraints are plotted along with f_1 and f_2, the resultant graph will display the noninferior solutions. This graph is shown in Figure 25 below. The five constraints

form a region of points that may be feasible. This region is further restricted by the bounding of f_2 between r_2 max and min. When f_1 is also plotted, the *N-set* (or efficient frontier) can be found. These noninferior solutions are depicted by the line segment A-B-C-D-E. The graphing of the decision variables, x_1 and x_2, was done in the **X** space. The next step in solving the MOLP is to transform the noninferior solution into the **Y** or criterion space.

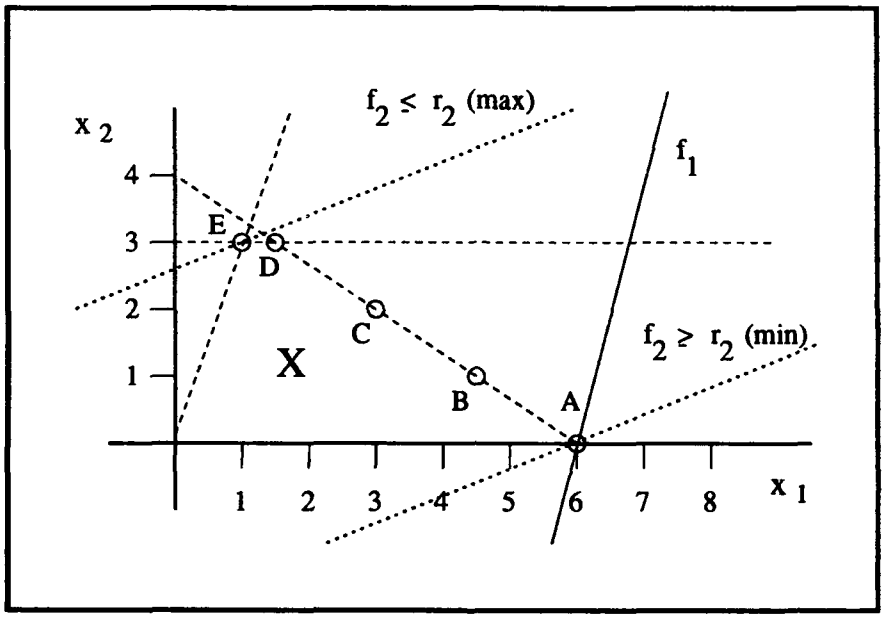

Figure 25. Graphical Display of Constraints and Objectives in X Space

If the noninferior points are substituted into the two objectives, these points can be remapped into the **Y** space. Using the points A-E, the remapped points, A'-E', are displayed in Table 7 below. Plotting these new points graphically displays the remapping into **Y** space. This graph is shown in Figure 26 below.

The goal of multiobjective optimization is to optimize all objectives at the same time. While the **Y** space graphically depicts the points that optimize the objectives, it does not show the relative weights of each objective. In other words, if the objectives are combined into one objective function, what is the relative contribution of

Table 7. Mapping of Points From X Space to Y Space

X Space		*Y Space*	
Point	(x_1, x_2)	*Point*	(f_1, f_2)
A	(6,0)	A'	(24,-12)
B	(4.5,1)	B'	(17,-5)
C	(3,2)	C'	(10,3)
D	(1.5,3)	D'	(3,12)
E	(1,3)	E'	(1,13)

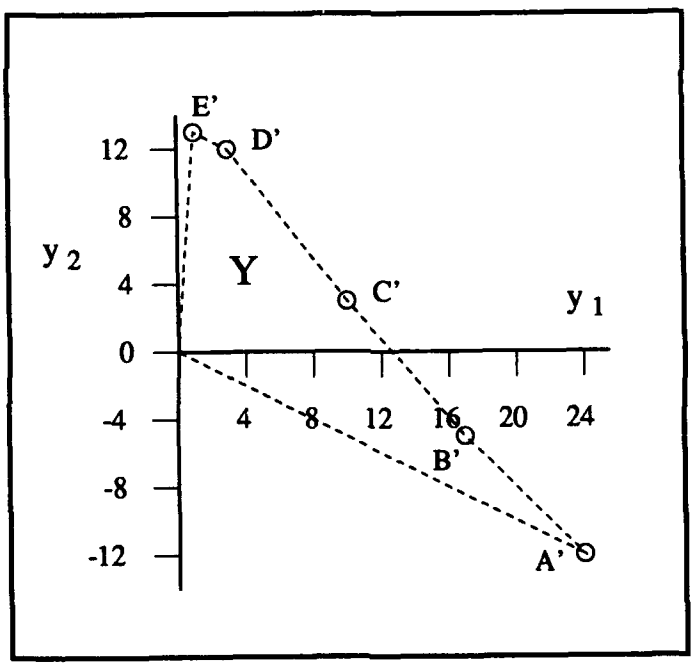

Figure 26. Graphical Display of Constraints and Objectives in Y Space

each objective at each noninferior point? One method for determining these relative weights is called the *weighted-sums approach.*

The weighted sums approach has appeal since a MOLP can be converted into a single criterion LP, which can be solved with most commonly available LP software. The general formulation of the weighted sums approach is as follows (33:166):

$$\max \{\boldsymbol{\lambda}^T \mathbf{C}\mathbf{x} | \mathbf{x} \in S\} \tag{57}$$

In this equation, $\boldsymbol{\lambda}$ is the weighting vector which is normalized so that all its elements sum to one. The set of all weight vectors is denoted by Λ and is defined in Equation 58 below. $\mathbf{C}$ is the $k \times n$ criterion matrix whose rows are the $\mathbf{c}^i$'s, and S is feasible region. This method attempts to optimize the composite objective function for a given $\boldsymbol{\lambda}$. Thus, this method only "experiments with convex combinations of the objectives" (33:165).

$$\Lambda = \left\{ \boldsymbol{\lambda} \in R^k | \lambda_i > 0, \quad \sum_{i=1}^{k} \lambda_i = 1 \right\}. \tag{58}$$

Given the multiobjective problem that was converted from $\mathbf{X}$ space to $\mathbf{Y}$ space, one may ask,"What combination of (λ_1, λ_2) would make the A', B', ..., or E' a maximum point of $\lambda_1 f_1 + \lambda_2 f_2$ over $\mathbf{Y}$?" Considering only point D', it can be shown at which lambdas this point remains optimal and at what lambda combinations the optimal solution will shift from this point. For the following equation, changes in the λ weights can shift the maximum from D' to E'.

$$(\lambda_1, \lambda_2) \begin{bmatrix} y_1 - 3 \\ y_2 - 12 \end{bmatrix} \leq 0 \tag{59}$$

For a shift from D' to E' to occur, the λ weights must satisfy the following equation:

$$(y_2 - 12)/(y_1 - 3) \leq -\lambda_1/\lambda_2 \tag{60}$$

where y is the set of points in the feasible region Y (5:7), (4). By varying the λ weights, the shift of the optimal point D' can be shown. Consider the examples of Λ(D) and Λ(E) shown in Table 8 below.

Table 8. Λ(D) and Λ(E) Mapping in **Y**-Space. Adapted from (5:7).

Lambda Weights		*Optimal Point(s)*	*Y Space Mapping*	
λ_1	λ_2	x^*	f_1	f_2
0.5	0.5	D'	3	12
.333	0.667	E'-D'	1-3	12-13
0	1.0	E'	1	13

Another way to think of this concept is to consider the multiple objective function, $z = \lambda_1 f_1 + \lambda_2 f_2$, which is rotated around point D'. Table 8 records the break point at which D' is no longer optimal as the lambda weights change (5:7). The results of this break point are shown in Figure 27, which is graphed in the **Z**-space.

3.3.2 Application of Multicriteria Optimization In the B&W model, the objective function attempts to maximize the total gray value subject to the given constraints. The objective function for one channel is given as follows:

$$\text{Maximize } Z = \sum_{k=1}^{K} w_k \left[\sum_{i=1}^{n} c_i x_{ik} \right] \tag{61}$$

In this equation, c_i corresponds to the gray value of the pixel at location x_i. The variable w_k can be assigned a value of one if the number of subregions to be found in the image is one or if the relative importance (weight) of finding one subregion over another is equal (for two or more subregions). In the following objective equations, I will assign a value of one to w_k. This equation as presently written does not allow

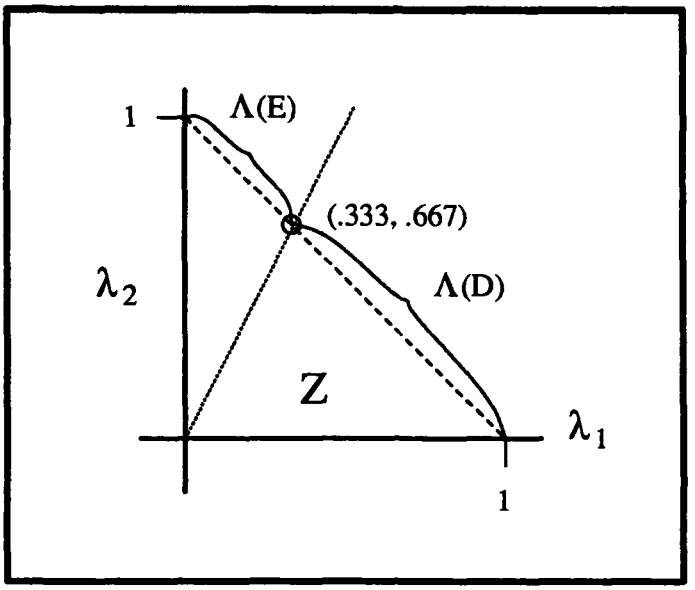

Figure 27. Mapping of D' Breakpoint in Z-Space

for the simultaneous processing of two or more images. For the processing of two channels, the MOLP can be written similarly to the example described above.

$$\begin{aligned} \text{Max } f_1(x_{ik}) &= \sum_{k=1}^{K}\left[\sum_{i=1}^{n} c_{i1}x_{ik}\right] \\ \text{Max } f_2(x_{ik}) &= \sum_{k=1}^{K}\left[\sum_{i=1}^{n} c_{i2}x_{ik}\right] \\ \text{subject to constraints} & \end{aligned} \tag{62}$$

Using the multicriteria optimization techniques described in the previous section, a new objective function can be formulated. The goal of processing two or more images is to maximize the combined gray value total from each channel, subject to the constraints. Applying the weighted sums methodology, the new objective function can be written as:

$$\text{Maximize } Z = \sum_{k=1}^{K} w_g \left[\sum_{i=1}^{n} (c_{i1}x_{ik} + c_{i2}x_{ik} + \ldots + c_{ig}x_{ik}) \right] \tag{63}$$

or

$$\text{Maximize } Z = \sum_{k=1}^{K} w_g \left[\sum_{i=1}^{n} \sum_{g=1}^{g} (c_{ig}x_{ik}) \right] \tag{64}$$

In these equations, c_{ig} represents the gray value of the x_{ik} pixel in the gth channel. Therefore, if three channels are available for processing, then there will be three objective functions in the combined objective function. The variable w_g is the placeholder for the lambda weights of the individual channels. Notice that the variables in each objective function are the same, as required in multicriteria optimization. Only the coefficients change as the channel changes. This method of multicriteria optimization could not be used if the variables were different. If they were, we could only optimize over the common variables. It is the coefficients that change as the channel changes. The goal of this combined objective function is to maximize the gray value total of the pixels selected in the g channels for a given set of lambda weights.

In the image processing objective function above (Equation 64), the weight for each channel varies from 0 to 1. In other words, if two channels are being processed, the relative importance of selecting a pixel in one channel over another pixel in another channel can be changed. Consider the following example. For a user specified object, the spectral range is set in each channel via the constraints. If the analyst is looking for a forest, the spectral range may be 50 – 75 in channel 1 and 100 – 200 in channel 2. Given this information, the analyst is able to determine an upper and lower gray value total for an object of specified size. In Figure 28, the gray value bounds are shown for each channel (with lambda weights both set to one). However, the combined objective function maximizes both channels at the same time, so the feasible range would be the addition of the two ranges. The addition of the two channels is shown in Figure 29 (with the lambda weights both set to one).

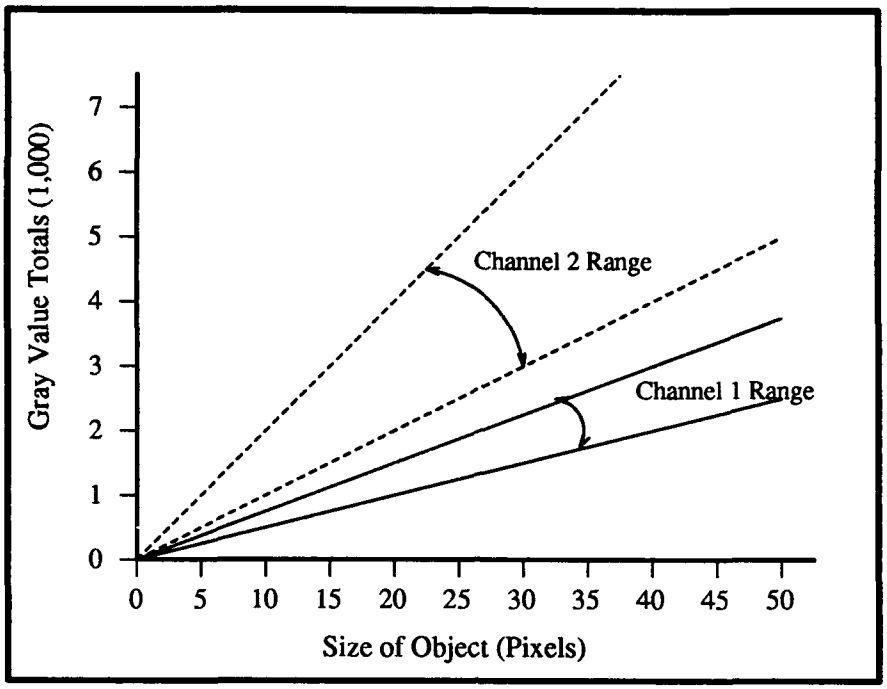

Figure 28. Graphical Display of Gray Value Total Bounds By Channel

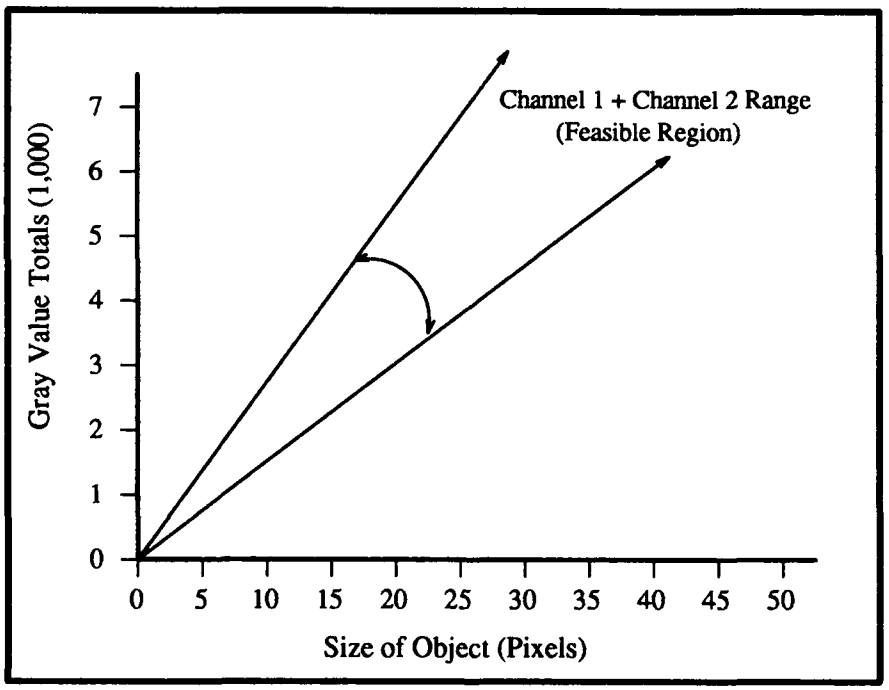

Figure 29. Graphical Display of Combined Gray Value Total Bounds

In Figures 28 and 29, the efficient frontier is not shown since individual lambda weights have not been applied to the objective function. If the lambda weights are applied for a subregion of a given size, the efficient frontier can be plotted. Consider the two-channel 2×2 image in Figure 30. In this figure, the location of the pixels as well as the gray values for the two channels are given. In Table 9, the lambda weights from 0 to 1 are applied to the two channels, for a subregion of two pixels. In Figure 31, the efficient frontier for this subregion is plotted. Although not shown, the plotting of the efficient frontier begins at the origin (0,0), goes to a maximum on the Channel 1 axis (0,135), next to the maximum on the Channel 2 axis (350,0), and finally, back to the origin. This characteristic start and finish of the efficient frontier is shown more clearly in the previous example (Figure 26).

1	2
3	4

Pixel Location

60	75
50	55

Channel 1

190	100
160	140

Channel 2

Figure 30. Two-Channel Image with Gray Values and Locations

In Figure 31 the **Y**-space for two channels being processed with a maximize objective function lies between the two axes and the dotted line. The closer the **Z** value approaches to the dotted line (efficient frontier), the closer the objective function comes in reaching a noninferior solution (for a given set of lambda weights). The application of the lambda weights in digital image processing has an effect that is not shown in Figure 31. With lambda weights of (1,0), only Channel 1 is processed. With lambda weights of (0,1), only Channel 2 is processed. With any other lambda weights, both channels are processed with relative weights. The implication of this is that the subregion found with one set of lambda weights may not be the same as the

Table 9. Mapping of Efficient Frontier in **Y**-Space

Optimal Points	*Lambda Weights*		*Y Space Mapping*		*Selected Pixel Locations*
x^*	λ_1	λ_2	f_1	f_2	
A'	1.0	0	135.0	0	1,2
B'	0.9	0.1	121.5	29.0	1,2
C'	0.8	0.2	92.0	66.0	1,4
D'	0.7	0.3	77.0	105.0	1,3
E'	0.6	0.4	66.0	140.0	1,3
F'	0.5	0.5	55.0	165.0	1,3
G'	0.4	0.6	44.0	210.0	1,3
H'	0.3	0.7	33.0	245.0	1,3
I'	0.2	0.8	22.0	280.0	1,3
J'	0.1	0.9	11.0	315.0	1,3
K'	0	1.0	0	350.0	1,3

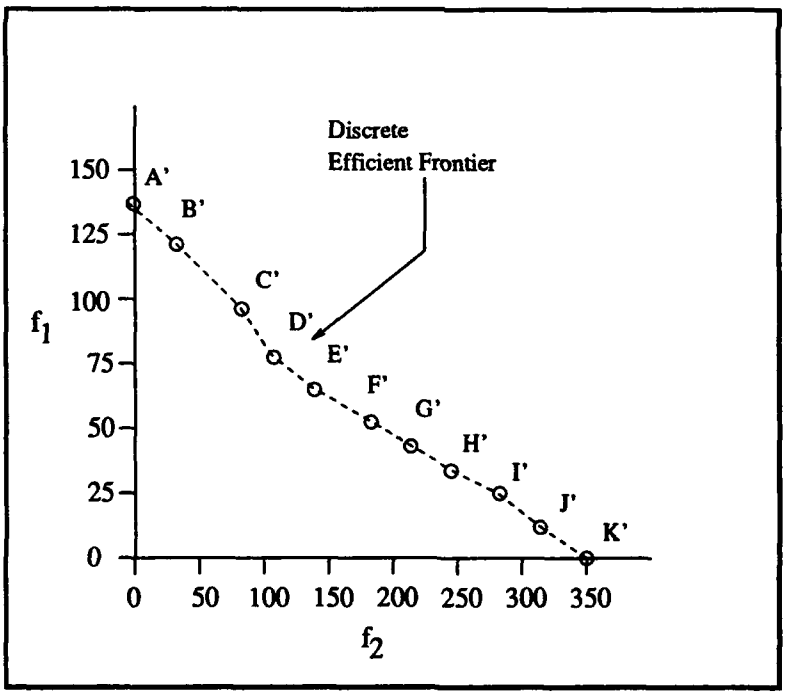

Figure 31. Efficient Frontier For Two Pixel Sized Subregion

subregion found with another set of lambda weights. This result was shown is the above example. For digital image processing, one can not assume that the optimal point lies on the efficient frontier. The reason for this is that the subregion will rarely be composed of only the upper-spectral-bound pixels.[36] While the efficient frontier is defined by the dotted line in Figure 31, it is not the best representation of the forest type pixels.

Recall in the example for this figure that the spectral range was 50 - 75 for Channel 1 and 100 - 200 for Channel 2. In most cases, the most representative spectral value would not be the upper bound, but rather the Gaussian spectral mean value. Suppose the mean spectral values for this example were 60 and 140, respectively. If this was true, the efficient frontier would have to be redrawn. The new efficient frontier would move inside the feasible region. Once again the reason for this change is that the new efficient frontier line now reflects the combined spectral values that are most representative of the forest-type vegetation. This new line is much more applicable and reasonable for digital imagery processing than to the original efficient frontier line in Figure 31. Therefore, any regions found on this line are noninferior solutions for a defined subregion size.

For a given size target, an efficient frontier line can be calculated using the method described above. If multiple solutions result from changes in the lambda weights, the best or *optimal* solution will be the one located closest to the efficient frontier line.[37]

3.4 Conclusion

In this chapter, the B&W model was explained and several runs were made. Analysis of the model and results revealed two shortcomings: lack of consistency in

[36] In this example, a subregion on the efficient frontier line would only be composed of pixels with a gray value of 75 for Channel 1 and a gray value of 200 for Channel 2.

[37] How close a solution is located to the efficient frontier line is measure in terms of perpendicular distance.

border length accounting and lack of control over what pixels are selected. The model was then examined from the perspective on an imagery analyst. Additional short comings were identified and modifications were suggested to correct these problems. In the last section, a method for combining the processing of more than one channel was explained — multicriteria optimization. In this chapter, the groundwork was laid for the real computer runs to be made in Chapter 4.

IV. Spectral and Spatial Pattern Recognition in SPOT Imagery

This chapter will investigate analysis of SPOT digital imagery on a channel by channel basis and on a composite basis using the methods discussed in Chapters 2 and 3. The findings should indicate whether one method is better than the other for exploitation of multispectral imagery. If a suitable technique can be found for the composite method, this may be the preferred approach. This hypothesis is based on the premise that fewer calculations (composite method) are usually preferred over more calculations (individual method on each channel).

In this chapter, the SPOT imagery data set will first be described. This will include a brief description of the SPOT satellite system and also a physical description of the image data that will be analyzed. Following this description, the assumptions used in analyzing the data set will be identified and their significance explained. The data set will then be analyzed visually to report what appears in that portion of the image under analysis. The three channels will then be spectrally examined. My analysis will next compare and contrast the findings in each channel. Following this portion of the analysis, methods for combining the data will be explored. Concepts such as vegetation index and band ratios will be used. The individual channel spectral analysis and the combined spectral data will be used as an input to the modified B&W model. With the spectral analysis knowledge, I will attempt to find spatial patterns in the image based on the users inputs. This combined spectral and spatial pattern recognition will be performed on the single channels as well as the combined channels. The results from both of these image processing methods will be then be compared to determine the most efficient manner to process multispectral imagery.

4.1 Description of Imagery Data

For this thesis, a SPOT image of the Washington D.C. area was used as the source of multispectral data.[1] The coverage of the image is approximately 50 km by 50 km and centers around the city of Washington D.C. Thc actual image is composed of four SPOT images acquired on March 19 and 24, 1987[2] (19). Rather than attempting to analyze the whole image, the scope of this thesis will be limited to a 48 × 18 pixel subimage.[3] The area selected is a portion of the Washington D.C. mall located between the Lincoln Memorial and the Washington Monument. The actual area used in this thesis is outlined in Figure 32. The subimage can be viewed on a personal computer using the Training System – Image Processor (TS-IP) software developed by Dr. Thomas S. Kelso at the Air Force Institute of Technology.[4]

All three multispectral channels of this subimage will be used in the analysis contained in this chapter. The imaging bands for the SPOT system are listed in Table 10 below. In addition to the bands, the resolutions are also given. In Table 11 additional characteristics of the SPOT system are given.

In each channel, the individual pixels are allowed to take on one of 256 shades of gray. The number of gray levels is due to the 8-bit nature of the pixels.[5] The values of the pixels will range from zero (black) to 255 (white). The actual numerical value of each pixel by channel is shown in Appendix E.[6]

---

[1] Although the SPOT system is also capable of imaging in the panchromatic mode, this type of data was not used in this thesis.

[2] According to Glenn Kelly of the EROS Data Center (organization responsible for merging the four images), three of the four images were taken in the multispectral mode and one in the panchromatic mode (18).

[3] The actual dimensions of this subimage are 480 m × 180 m.

[4] The complete Washington D.C. area image is contained on a 60 megabyte CD-ROM which was prepared by the EROS Data Center.

[5] The number of gray levels in eight-bit data is calculated as follows: $2^8 = 256$.

[6] Note each channel subimage is surrounded with a border of 999s. These values are not part of the subimage. The first line of the Channel 1 image begins on the second line with the numerical sequence: 224 222 214 .... This border assists the model program in locating specific spectral and spatial patterns. The number 999 was chosen since it was visually distinct and because it was not in the eight-bit spectral range. Since it is not in the spectral range, it will not be selected as part of the image.

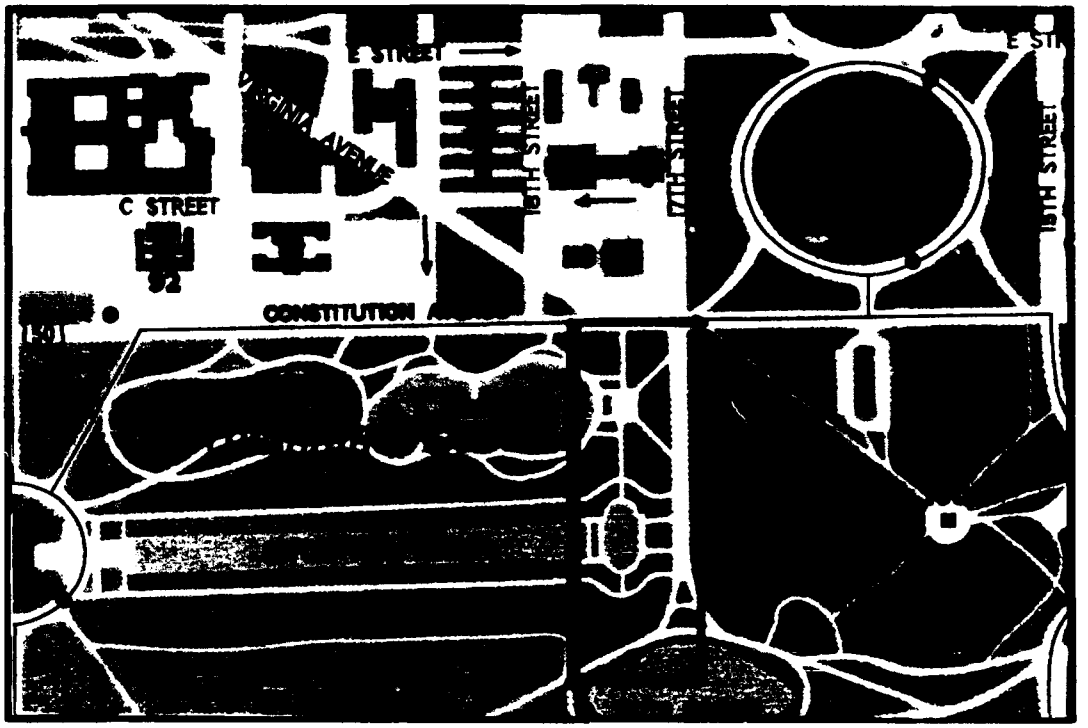

Figure 32. Portion of Washington D.C. Mall Under Analysis (37).

Table 10. SPOT Imaging Bands (34:26)

Mode	*Wavelength (μm)*	*Resolution* (m)*
Panchromatic	0.51 - 0.73	10
Multispectral		
Channel 1	0.50 - 0.59	20
Channel 2	0.61 - 0.68	20
Channel 3	0.79 - 0.89	20
* For nominal altitude of 832 km.		

Table 11. SPOT Satellite Characteristics (8:495-500), (21:585), (23:65)

Orbit:	
Shape	Circular
Altitude	832 km
Inclination	98.7° (Sun-Synchronous)
Descending Node	10:30 A.M. LST*
Orbital (Repeat) Cycle	26 Days
High-Resolution-Visible (HRV) Sensor:	
Number of Sensors	2/SPOT Satellite
Off-Nadir Viewing	±27° (±475 km from nadir)
Ground Swath Width (at nadir)	60 km/HRV 117 km/SPOT Satellite** 80 km/HRV at ±27°
Angular Field of View	4.13°
Radiometric Resolution:	
Multispectral	8 bits (256 gray levels)
Panchromatic	6 bits (64 gray levels)
* Local Solar Time	** 3 km overlap

4.2 Analysis of Data

In this section, the visible and spectral characteristics of the SPOT image data will be discussed. Before proceeding into the analysis, it is necessary to first state the assumptions used in this thesis.

4.2.1 Data Assumptions Several assumptions were made in the analysis of the SPOT data. Each assumption and its significance is called out below.

- *Rectification Assumption:* I assumed that no rectification was needed among each of the multispectral images. In layman's terms, this means that a pixel located at position (20,45) in Channel 1 is the same pixel located at (20,45) in Channels 2 and 3. By the phrase "same pixel," I mean that each pixel (with the same location, row and column position) contains the same area when projected onto the imaging plane. This assumption, simplifies the task of analyzing the

data, is not a bad one. That is because even though each channel was imaged at a different satellite position, the imaging cells of each channel are practically in the same location when compared to the distance from the SPOT satellite to the Earth. For this reason, the amount of geometric correction needed, due to the different location of the imaging cells, is negligible. This assumption will aid in comparing like pixels in each image based on location alone. This concept of rectification is shown in Figure 33.

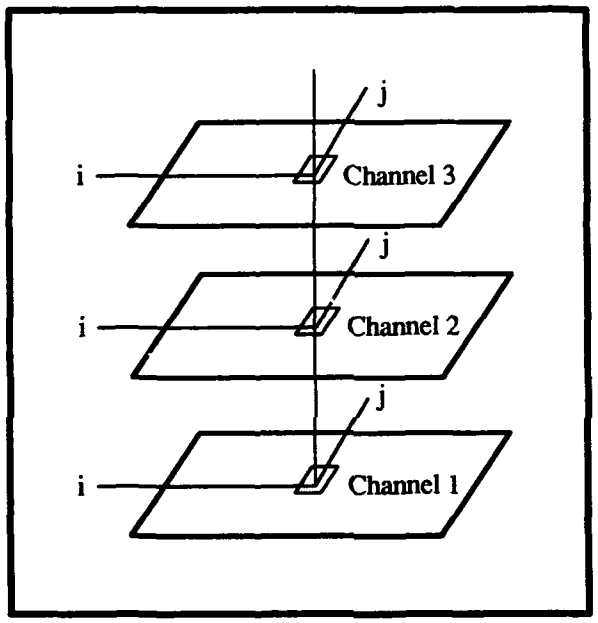

Figure 33. Multi-Channel Rectification Example. Adapted from (24:188).

- *Uniformity of Pixel Size:* A second assumption used in this chapter is that the pixels remain uniform in size (10 meters × 10 meters) throughout the entire image.[7] This assumption is greater in significance than the first assumption. The actual pixel size will vary more from the 10 × 10 meter size the further

[7]SPOT multispectral imagery has a 20m resolution. This statement is in direct conflict with the assumption made that the SPOT multispectral imagery is 10m in resolution. This increase in resolution was made possible by geometrically rectifying and spatially resampling the multispectral data to 10 meter resolution to match the panchromatic data (18). More information on this resampling process is contained in the *Minimal Image Processing Effects* paragraph in Section 4.2.1.

the pixel is located from the ground track of the SPOT satellite. The further the pixels are located from the ground track, the larger the area each pixel will cover. This increase in coverage (per pixel) is due to the viewing geometries of off-nadir imaging. In this thesis, the actual size of the pixels will not vary much (over the subimage) due to the small size of the data set under analysis. I will assume that each pixel, no matter where it is located in the image, will be 10×10 meters in size. This will be of great help when attempting to define the size of a target area or when searching for a spectral pattern of a particular size.

- *Minimal Atmospheric Effects:* This assumes the effects of the atmosphere on the true gray value of each pixel are negligible. If this assumption was not used, the effects of the atmosphere would have to be taken into account. The atmospheric effects that can change the values recorded from the actual gray values include: clouds, weather, and illumination conditions. Without the actual weather data for the dates the image was taken, the magnitude of this assumption is not known.
- *Minimal Image Processing Effects:* This assumption is similar to the atmospheric effects one. I have assumed that the processing effects on the image are minimal. This assumption is made because the scope of this thesis does not include the image processing techniques used by the EROS Data Center. In reality, the SPOT image was been processed quite a bit. This processing will affect the analysis process described in this chapter, but the magnitude of the effect is unknown. The analysis approach described in this chapter is general enough in nature that it could be applied to raw image data (no image processing). A detailed description of how the raw image data was manipulated is given below.

According to Dr. Kelso, the combined image of the Washington D.C. area was created by merging the red, green, and blue multispectral data with the panchro-

matic data from the four images taken on 19 and 24 March 1987. This merge was completed by the EROS Data Center in Sioux Falls, South Dakota. The following is a description of the processing completed by the EROS Data Center on the SPOT imagery.

> The merge of panchromatic and multispectral data was completed prior to mosaicking. To accomplish the merge, an image-to-image registration was first performed between each scene set. Approximately 60 control points were selected for each set using an autocorrelation technique. The multispectral data were geometrically rectified and spatially resampled to 10-meter pixel ground resolution during this procedure to match the panchromatic data using a second-order polynomial and cubic convolution resampler. Once the panchromatic and multispectral data sets were geometrically identical, the merge was performed using a hue-intensity-saturation (HIS) method. The data merge was effected by transforming a contrast-enhanced spectral image into HIS color space. The panchromatic data were stretched to match the distribution histogram of the spectral intensity channel, then substituted for the intensity channel to create a hybrid HIS data set. The hybrid HIS data was then transformed back into the red, green, and blue components. The four merged scenes were mosaicked together, each scene was registered to 1:24,000 scale maps by selecting approximately 10 control points per image. These control points were combined with tie points selected by Large Area Mosaicking Software (LAMS) along adjacent scene boundaries and a transformation grid was created. The grid was used to geometrically correct each image in relation to the Universal Transverse Mercator (UTM) projection and to each other. Once each scene was in proper geometric configuration, the radiometric matching between scenes was accomplished using LAMS. Final processing of the data included a 3 X 3 edge enhancement and multipoint linear stretch to increase contrast. The digital data were then output on a recorder in the form of a color transparency. The transparency was scanned on a graphic arts scanner and the digital color separates were generated. These separates are presented sequentially in the file EDC_SPOT.IMG on this disc. The number of lines and samples for this file are 15,045 and 5,650 respectively; this is an 8 bit image file. An IMDISP label file has been provided in this subdirectory for your convenience (WASH_DC). The top left-hand lines/samples coordinates for display of the respective color-band images are:

red	160	48
green	5157	48
blue	10190	48

The color separates were used to produce a color 1:50,000 scale map of the Washington, D.C. area available from the Geological Survey offices in Reston, VA, and Denver, CO, and from various commercial vendors that sell USGS maps and publications. Further information on the procedures described above may be obtained from the U.S. Geological Survey, EROS Data Center, Sioux Falls, South Dakota, 57198. Further information regarding the acquisition of SPOT data scenes can be obtained from Robert Lees, SPOT Image Corporation, Reston, VA (703-620-2200).

Be advised that the SPOT data presented here are copyrighted and were obtained under license from the SPOT Image Corporation, Reston, VA. This data may not be copied (whole or in part) from this disc onto any medium for distribution without the prior written permission of the SPOT Image Corporation (19).

4.2.2 Visual Analysis Before spectral analysis of the subimage is begun, it is necessary to describe the SPOT subimage. The portion of the SPOT image used for this thesis is a section of the Washington D.C. mall. The physical dimensions of the subimage is 980 m × 360 m. The subimage is bounded on the eastern side by the western edge of the Lincoln Memorial reflection pool and bounded on the west just past 17th Street. It is bounded on the northern edge, by Constitution Avenue and on the southern edge by the Tidal Basin. Four bodies of water are contained within the subimage. The subimage also contains numerous pathways and the two roads called out previously. The remainder of the subimage is covered by grass-type vegetation. Additional visual analysis of this subimage is given below.

As was mentioned earlier, the three SPOT subimages were viewed on the TS-IP software. In order to match the individual pixels with a gray value number, it was necessary to perform two steps. The first step was to blow up the area of interest (the 48 × 18 subimage) 800 percent. This was accomplished by using the zoom feature in

the TS-IP software three times.[8] With the picture blown up to this magnification, it was easy to see the individual pixels.

With the individual pixels resolved, there appeared to be several *themes* within the scene. These *themes* were apparent by viewing the images at 800 percent magnification with different palettes. The most helpful palettes were the 256-gray, the blue-green-red-yellow (BGRY), and the topo. After viewing these subimages in the different palettes, the themes that appeared to be present were roads, pathways, grass-type vegetation, and water. It should be noted that these apparent themes were verified using a National Geographic map for ground truth (37).[9] Normally, after an image has been segmented into themes, these areas are then compared with the actual land cover for that area. With a ground truth standard, one can verify if the initial classification of the segmented areas is correct. The methodology discussed in this paragraph is referred to as *unsupervised classification*. This method is contrasted with *supervised classification* where the analyst knows what is on the ground (in selected areas) before the analysis actually begins.

The second step in the analysis of the three subimages entailed matching the segmented themes with individual pixel gray values. This process is described in detail after a more thorough description of the target of interest is given.

4.2.3 Description of Target In the previous section, it was noted that four bodies of water existed. These bodies of water are portions of the Lincoln Memorial reflecting pool, the Constitution Gardens lake, the Tidal Basin, and the complete reflecting pool east of the Lincoln Memorial reflecting pool. It is this last body of

[8]Within the zoom feature, the *nearest neighbor* option was selected rather than the straight *blow up* option. The latter option was able to blow up the image data, but the output was not very useful to work with.

[9]Ideally, the initial segementation would be verified by actually viewing the area under observation on the day of imaging. This is ideal because a scene will change as time passes. This change in image scene is most obvious with the annual growth and death of vegetation. Since it is not possible to travel back in time (to the data of the imaging, 19 and 24 March 1987), I will focus on locating those types of landcover that are relatively permanent and unaffected by the seasonal changes.

water that I will attempt to find with the modified B&W model. Water in general was chosen for one main reason. Water is spectrally distinctive enough to be visually distinguishable from the background when viewing the subimage on the TS-IP system or when viewing the actual gray value numbers. The contrast of the other themes against the grassy-vegetation background is not as high as water.[10] Of the four bodies of water, the reflecting pool east of the Lincoln Memorial reflecting pool was chosen for two reasons. The first and most important reason is that since it is manmade, it is relatively permanent over time. This is contrasted with the Tidal Basin which will vary monthly with the tides. If a target is constantly varying, then the size and border length will also be constantly varying. Any variation in the target makes it more difficult to accurately locate. Since the reflection pool of interest varies little over time, an approxiamte size and border length can be estimated. Another reason this reflecting pool was chosen is that the target area is not rectangluar in shape. If an object is rectangular, the object can be easily found with previously existing implementations of the B&W model[11] (7). In the following section, all four bodies of water are spectrally analyzed in all three channels.

4.2.4 Spectral Analysis Before the data could be spectrally analyzed, it was necessary to break each channel data set down into its individual gray values. Using a FORTRAN program, I read in the three subimages and recorded the gray values in each image. In Appendix F, the gray value count and gray value histogram for each channel is recorded.[12] Reviewing this data reveals several interesting observations. The first is the range of the spectral values. In Channel 1, the values ranged from 0 to 227, while in Channel 2, the range was 0 to 216 and in Channel 3, from 0 to

[10]These statements about the spectral distinctiveness of water only apply to Channel 1. While the water regions can be seen in the other channels, it is difficult to tell where the borders are located.

[11]A GAMS implementation of the B&W model to find rectangular subregions was created in 1991 by Dan Burke, Research Assistant at the Air Force Institute of Technology.

[12]Also contained in this appendix is the FORTRAN program created to tabulate the gray value counts.

212. These different ranges were expected since the spectral signature for an object changes as the spectral band changes. Another interesting point was that the range of gray values was not continuous (between the upper and lower bounds) as one might expect. Rather, most of the image data had spectral values that were not continuous. Upon viewing the data visually in the TS-IP software, clues were found to suggest why the ranges are not continuous from 0 to 255. As was mentioned earlier in the visual analysis section, only four types of land cover appear to be present in the subimage. Because the image does not vary much in content, the subimages will not contain the whole spectral range of 0 to 255.[13] The most significant reason why the spectral values are not continuous is because of the image processing performed on the SPOT imagery. Recall in Section 4.2.1 that the imagery was processed with a multipoint linear stretch to increase the contrast. The result of this stretching will cause gaps to appear in otherwise continuous data. The magnitude of the stretch and the points of the stretch are unknown. According to Glenn Kelly, of the EROS Data center, the original records for this work no longer exist, so it is difficult to determine the magnitude of this linear stretch (18). In the following sections, spectral analysis will be confined to four bodies of water since that is what I am attemping to find. The process of spectral analysis in each channel is not unique to water and could be applied to any other type of land cover in the subimage.

4.2.4.1 Channel 1 Spectral Analysis When viewing the histogram of pixel gray values, it is not obvious where the natural break in the image gray values occurs for the water subregions. For this reason it was necessary to look at the ground truth map and compare that with the visual image on the TS-IP system and also the actual gray values. In this channel, all the water subregions showed up as navy blue areas on the TS-IP system when using the BGRY palette. The

[13]With only four major land cover types, the spectral signatures will be discrete rather than continuous. There are many other features not present which would yield different spectral signatures. Some of these features include: forest-type vegetation, buildings, mountains, snow, bare rock, sand, ocean, etc.

size and shape of these areas were then compared to the ground truth map. Using control points in both the map and the visual image, the water subregions were identified.[14] The TS-IP system, has the ability to display the pixel gray value for a given location.[15] With the water subregions located, all that was necessary was to record the gray values. In these four subregions, the spectral range for water was from 0 - 22.

I also scanned past the subimage borders to see it there were any other subregions in the image with this characteristic spectral range. Two major subregions were found with this same spectral range in the vicinity of the Mall. The first subregion was the Potomac River, which was expected. The second subregion was more of a surprise. The shadow of the Washington Monument also had gray values in this characteristic spectral range. Also contained in the subimage was what appeared to be noise with the same spectral range. All of these noise-type subregions were one or two pixels in size. I was not able to definitively identify what these areas were with the ground truth map. In Appendix G, these water subregions are identified with the letter W. The subimage is surrounded by a border of 999s. This border serves two functions: 1) all the subimage pixels are now interior pixels and 2) it is easier to locate the position of a given pixel. The numbering of the pixels begins with the first row of 999s. The first subimage pixel is thus in position number 22. I have converted the rest of pixels in the subimage to a blank space.

From the processed subimage in Appendix G, it is obvious that this spectral range identifies water, but not uniquely.[16] In Table 12 the gray values for water and the combined number of these pixels from the four subregions are charted. The total number of pixels in the subimage within this gray value range is also given. The overall accuracy in indentifying the water contained in the four subregions

[14]In digital imagery, control points are locations/features that readily apparent in both the map and image.

[15]This is the examine point option on the TS-IP system.

[16]Compare the processed image in Appendix G with the ground truth map and the similarity can be seen.

(with this spectral range) is 94 percent. More spectral information is contained in Channels 2 and 3, so it is necessary to also process these channels. By acquiring more information, the sizing and locating of these water subregions should become more accurate. Channel 2 is spectrally analyzed in the next section.

Table 12. Classification of Water in Channel 1

Gray Value	*Pixel Count*		*Classification Accuracy*
	Water Areas	*Total*	
0	57	58	98%
3	2	2	100%
6	3	4	75%
9	4	4	100%
13	2	2	100%
16	3	5	60%
19	2	2	100%
22	3	6	50%

4.2.4.2 Channel 2 Spectral Analysis In Channel 2, the water regions were not as spectrally distinct. Therefore, the same spectral analysis methods (visual inspection) used in Channel 1 could not be used for Channel 2. It was first necessary to locate the water subregions so the gray values could be recorded. With these values recorded, the characteristic gray value range for water in Channel 2 can be identified. By using the Rectification Assumption described in Section 4.2.1, the water subregion can be easily located. In other words the location of the water pixels in Channel 2 have the same location as in Channel 1. Using this methodology, the spectral range of water is 5 – 166. This wide range for water shows why the water is not as spectrally distinct as in Channel 1. Even the *color* of the water (a multiband concept) varied over this range.[17] This range of colors for the water spectral range is shown in Table 13.

[17] I do not mean the actual color of the water but rather the false color of the water as shown on the TS-IP system with the BGRY palette.

Table 13. False Color of Water in Channel 2

Gray Value Interval	*False Color*
5 - 39	Navy Blue
48	Royal Blue
58	Blue
67	Light Blue
77 - 86	Turquoise
96 - 105	Light Green
111 - 116	Green
127	Olive
138 - 166	Red

From Table 13 one can see that the spectral range for water is considerably larger in Channel 2. This range is more significant when the spectral range for the entire subimage is considered: 0 – 216. Thus, the spectral range for water nearly encompassed the entire spectral range for the subimage. I also scanned the rest of the subimage to see what other subregions were in the water spectral range. In general, grassy vegetation and the shadow of the Washington Monument were also navy blue in color. The pathways and roads were several shades of green in color. With this overlap in spectral signatures, it is difficult to uniquely identify the water subregions with Channel 2 spectral information alone.

In Table 14 the accuracy in identifing the water subregions is shown. There is a marked decrease in the overall accuracy when compared to Channel 1. When viewing the *Pixel Count Total* column, one can see that there are many other regions (in the subimage) composed of the same spectral values as water. This observation is shown graphically in Appendix G. In this processed image, the pixels with a gray value between 5 and 166 are labelled with the letter W. With this processed image, it is difficult to distinguish the four regions of water.

Table 14. Classification of Water in Channel 2

Gray Value	*Pixel Count*		*Classification Accuracy*
	Water Areas	*Total*	
5	3	5	60%
6	1	8	12.5%
7	4	16	25%
9	3	28	11%
10	3	33	9.1%
20	2	26	7.7%
29	5	38	13%
39	5	41	12%
48	10	45	22%
58	6	59	10%
67	4	58	6.9%
77	7	39	18%
96	3	48	6.3%
105	5	47	11%
111	3	28	5.2%
116	3	35	5.7%
122	1	31	3.2%
127	2	12	17%
133	1	22	4.6%
138	5	26	19%
144	2	11	18%
155	2	24	8.3%
166	1	9	11.1%

4.2.4.3 Channel 3 Spectral Analysis In Channel 3, the water regions were again spectrally non-distinct. In fact, the problem in locating the water was similar to Channel 2 but worse. Therefore, the Rectification Assumption was again used to find the spectral range for water. Using the same methodology as applied to Channel 2, the spectral range of water is 5 – 200. This wide range for water shows why the water is not visually apparent when veiwed on the TS-IP system. Once again, the false color of the water varied greatly over this spectral range. This range of colors for the water spectral range is shown in Table 15. The range for the entire subimage is 0 - 212. Given that the range of water nearly matches this subimage range, the classification accuracy will be very low as shown in Table 16.

In Appendix G, the Channel 3 subimage has been processed so that any pixels within the 5 – 200 range are labelled with the letter W. This processed image graphically shows the poor classification accuracy of this spectral range for water. In scanning the rest of the subimage, other land cover themes had the same spectral values. In general, the grassy areas and the Washington Monument shadow were navy blue in color. The roads were green in color, and the pathways were a combination of red and orange colored pixels. Based on this data, classification accuracy of the water subregions with Channnel 3 spectral data alone is very low.

In summary, each of the three channels were spectrally analyzed to determine the gray value range for the four bodies of water in the subimage. Of the three channels, Channel 1 has the most distinctive spectral range for water. The consequence of this fact is that the water regions are easily resolvable from the background. Due to the extended range for the other two channels, the water regions were not easily resolved. When Channels 2 or 3 were combined with Channel 1, two *noise-type* pixels were removed from the processed image. These pixels were located at positions 594 and 665. The information gained from this analysis will be used to find the reflection pool in the subimage. The spectral analysis information serves as an input into the

Table 15. False Color of Water in Channel 3

Gray Value Interval	*False Color*
5 - 19	Navy Blue
37	Royal Blue
46 - 55	Blue
64 - 67	Light Blue
73 - 82	Turquoise
91 - 106	Light Green
111 - 117	Green
128	Olive
133 - 138	Red
161 - 178	Orange
183 - 200	Lt. Orange

computer model, namely the pixel bounds constraint. The running of the modified B&W model will be explained in the next section.

4.3 Computer Model Runs

In this section, several modified versions of the B&W model will be run to find the reflection pool of interest. The model will be run with different combinations of channels and also a combined image. The combined image is the result of merging two channels with the NVI process. These different sets of data will be compared to see which one gives the most accurate answer and which one is the most efficient in terms of computer processing time.

4.3.1 Selection of Computer Model Equations In Chapter 3 many modifications were suggested for the B&W model. In this section, the various improvements will be tested on small problems before attempting to use them to solve the SPOT subimage problem. One of the image processing methods explained in Chapter 3 was the creation of a new objective function that would bias the selection of pixels

Table 16. Classification of Water in Channel 3

Gray Value	*Pixel Count*		*Classification Accuracy*
	Water Areas	*Total*	
5	1	2	50%
6	1	4	25%
8	2	27	7.4%
10	3	21	14%
19	3	28	11%
28	2	32	6.3%
37	1	28	3.6%
46	3	36	8.3%
55	2	45	4.4%
64	6	38	16%
73	10	41	24%
82	5	47	11%
91	6	39	15%
100	2	41	4.9%
106	3	33	9.1%
111	3	44	6.8%
117	3	34	8.8%
122	4	32	13%
128	1	20	5%
133	3	26	12%
156	1	16	6.2%
161	3	15	20%
167	2	14	14%
172	1	12	8.3%
178	2	10	20%
183	2	16	13%
189	1	7	14%
200	1	5	20%

to the mean pixel gray value. In order to process more than one channel of data, it is necessary to incorporate the process of multicriteria optimization. One objective function which merges these two concepts is shown below. Note that this equation is only valid for the combining of two channels. It can be extended to more than two channels as long as the number of channels to be processed matches the number of lamda weights.

$$\mathbf{Z} = \text{Max} \left[\lambda_1 \left(\sum_{i=1}^{n} \frac{1}{|(\overline{x}_1 - c_i x_{i1})| + 1} \right) + \lambda_2 \left(\sum_{i=1}^{n} \frac{1}{|(\overline{x}_2 - c_i x_{i2})| + 1} \right) \right] \tag{65}$$

I used this objective function on a small 7×7 matrix. When run on the GAMS system, the following error resulted:

51 IN EQUATION OBJ .. ENDOG ARGUMENT(S) IN FUNCTION

Before attempting to rework and rerun the program, I called the GAMS Development Center in Washington DC. According to Erwin Kalvelagen, two operations were causing this problem: the absolute value function and the presence of a variable ($x_{i,k}$) in the denominator.[18] Although the objective function appears to be linear, these two operations (absolute value and division with a variable) make it nonlinear for GAMS. Erwin added that the absolute value function makes the derivatives noncontinuous when GAMS attempts to solve the problem. He said there was a possibility that the problem could be solved with the MINOS option in GAMS, but that the integer solution would be forfeited. He confirmed that if the problem is to be solved as a mixed integer problem, the absolute value and the division operation need to be removed (16). For a partial listing of this GAMS run, see Appendix H.

[18]In GAMS, an endogenous variable is a variable which the program is attempting to solve for.

In light of this new knowledge on the limitations of GAMS, it was necessary to reformulate the objective function. Although the three objective functions in Section 3.2.4 are mathematically correct, they can not be solved as mixed integer problems in GAMS. The reason for this limitation is that all of the three new objective functions have an endogenous variable (x_{ik}) in a mathematical function. Given that the new objective functions are not compatible with GAMS, the original objective function with lambda weights will be used. An equation that will work as either a maximization or minimization objective function is given below.

$$\mathbf{Z} = \text{Max/Min} \left[\lambda_1 \left(\sum_{i=1}^{n} c_{i1} x_{ik} \right) + \lambda_2 \left(\sum_{i=1}^{n} c_{i2} x_{ik} \right) \right] \tag{66}$$

While this equation will work, it was pointed out in Chapter 3 that it will be biased. If the maximization objective function is used, the model is biased to pick the spectrally lighter gray values. If the minimization objective function is used, just the opposite will occur. Thus, the model can be run with this objective function, but the analyst needs to remember the implications of a maximization or minimization objective function.

Another modification that I attempted to implement was an improvement to the contiguity constraint. For a given sized object I attempted to enforce contiguity with the following equation.

If $X(I, K) = 1$ then

$$X(I-1, K) + X(I+1, K) + X(I-20, K) - X(I+20, K) \geq 1 \tag{67}$$

With this equation, each pixel selected ($X(I, K) = 1$) is required to have at least two first order neighbors. The efficiency of the model should be improved since the equations for the P_{ijk} and N_{ijk} variables could be removed. When I attempted to run this improvement in the GAMS program, the following error resulted.

57 IN EQUATION C6 .. ENDOG OPERANDS REL OR AND XOR

$X(I, K)$ is a variable and this error indicated that GAMS does not allow an equation to be complied based on the status of a variable. In this instance I was not allowed to use the EQ (equals) operator with the endogenous variable. I attempted the program again but this time I removed the conditional "If $X(I, K) = 1$ then" statement from the constraint. This modified contiguity constraint successfully ran in GAMS, but the region found was not contiguous. Rather, several subregions were found that added up to the given sized area. However, the objective function takes precedence over this partial contiguity constraint to select pixels which will maximize or minimize the objective function as required.

Given these results, I was forced to use original size and border length constraints called out in Section 3.1. However, two major improvements can be made to the model. These improvements are the pixel bounds constraint and the multi-objective function. With this model I will attempt to locate the reflection pool of interest using two channel combinations.

4.3.2 Running of the Model In this section, the model will be run with different channel combinations and with different lambda weights. The goal of these programs is to locate the reflection pool identified earlier and shown in Appendices G and I. The spectral analysis performed earlier in this chapter will be used to set pixel gray value bounds for the model. Before running the model to find this one region, I performed a three-channel filter to identify all the water subregions. These subregions represent the possible pixels for selection by the model. Following this run, three sets of two-channel runs will be run and their results analyzed. If different solutions result, they can be compared to the efficient frontier to select the noninferior solution. None of the solutions will be on the efficient frontier since none of the

water subregions were homogeneous.[19] This statement can be verified by reviewing Appendix E.

4.3.2.1 Three-Channel Filter To identify the water regions in the SPOT subimage, I modified the objective function to incorporate all three channels. No lambda weights were used in this objective function since all the channels were of equal weight. In the three-channel filter I eliminated the border length and area constraints. With this representation of the model, all water regions will be identified based on their spectral values alone. This program is similar to the parallelepiped method discussed in Chapter 2. In Appendix I a character representation of this three-channel filter is shown. In addition, a partial listing of this program with the pixels selected is included. The solution in Appendix I shows the four main bodies of water along with random pixels scattered in the subimage. These random pixels may be noise.

4.3.2.2 Channel 1 - Channel 2 Programs In this section Channel 1 - Channel 2 programs will be run to locate the reflection pool identified earlier. The objective function is a maximization one and will incorporate the concept of lambda weights (See Equation 66). All the constraints identified in Chapter 3 will be used. The water region of interest has an area of 24 pixels. The corresponding border length is 44. Both of these values can be verified by reviewing Appendices G and I. Recall in Chapter 3 that a range of border lengths can be calculated for a given size area. For this problem, the border length range consists of the even values from 36 to 192.[20] In reality, this upper border length is too high. The B&W model stated that the upper border length was four times the size of the area. This number must then be multiplied by two since the actual border length is double accounted. In order

[19]In this context, *homogeneous* means pixels of the same gray value. If the objective function was a maximization, all the pixels would need to be at the upper spectral bound. If the objective fuction was a minimization one, just the opposite would be true.

[20]This range of border lengths was calculated using the border length equations in Section 3.1.

for each pixel to have a border length of four, they would have to be non-contiguous. Because of the contiguity constraints, this condition would not occur. The border length constraint is exact in this problem since the target is aligned with the pixel grid. Recall that if this were not the case, a stair step border would result. If this were the case, the analyst could start with the lower bound of the border length for a given area. Then the analyst could systematically increase the border length by two until the desired subregion was found.

In addition to the constraints identified in the B&W model, two additional constraints were added to this program. These constraints set the spectral bounds for water in each channel and set the value of x_{ik} to zero if the pixel gray value is out of this range. With these constraints, only the water-type pixels are considered for selection into a water region. These constraints were implemented in GAMS with the following lines of code:

```
X.FX(I,"1")$(C(I,"A") LT 22) = 0;
X.FX(I,"1")$(C(I,"B") LT 5 OR C(I,"B") GT 166) = 0;
```

where the A variable represents Channel 1 and the B variable, Channel 2.

A total of eleven runs were made with the Channel 1 - Channel 2 combined objective function. In Equation 66, the lambda weights were each allowed to vary from 0.0 - 1.0. A partial listing of the second run (lambda weights of 0.1 and 0.9) is shown in Appendix J.[21] In Table 17 the results of varying the lambda weights are given.

In these GAMS runs, a total of 8723 equations were generated for each program. Considering that the objective function is a maximize one, one may be initially inclined to select the second run as the optimal solution. However, one should recall

[21]Only a partial listing of the GAMS output is included in Appendix J, since the length of the output was typically 450+ pages. The FORTRAN programs to generate the GAMS adjacency and cost matrix are also included in Appendix J.

Table 17. Results of Channel 1 - Channel 2 Runs

GAMS Run	*Lambda Weights*		*Objective Value*	*Iteration Count*	*Execution Time (Sec)*		*Resource Usage*
1	0.0	1.0		*			
2	0.1	0.9	1706	1544	119.43	37.81	544.89
3	0.2	0.8	1524	1425	123.88	38.18	513.58
4	0.3	0.7	1342	1405	122.16	38.27	509.36
5	0.4	0.6	1160	1365	125.65	39.30	510.61
6	0.5	0.5	978	1634	118.75	39.15	587.86
7	0.6	0.4	796	1629	122.29	39.38	589.96
8	0.7	0.3	614	1542	123.53	43.36	577.38
9	0.8	0.2	432	1727	126.02	37.39	632.02
10	0.9	0.1	250	1517	126.12	34.52	548.37
11	1.0	0.0	68	1353	132.36	41.06	566.04

* No final solution after 5000 iterations.

that the efficient frontier value will change as the lambda weights are changed. All of these runs selected the same 24 pixels. Their numerical positions are 528, 529, 547–550, 567–570, 587–590, 607–610, 627–630, 648, and 649. All of these pixels were fully allocated. In other words, the x_{ik} values for each of these 24 pixels were one. These pixels exactly match the size and the location of the reflection pool shown in Appendix I. Since there is no variance in the pixels selected, an optimal solution can not be chosen. Rather, they are equal answers in their optimality. They are all equally spaced from the efficient frontier. If different 24-pixel-area subregions were selected, the efficient frontier would need to be drawn and the solutions plotted. The solution(s) closest to the efficient frontier would be the optimal solution. Recall that the term *optimal* in this context is speaking strictly in an operations research sense. It was explained earlier that the optimal solution in digital imagery processing may not be the same. In other words if different subregions were found by changing the lambda weights, the solutions may or may not be correct for the imagery analyst. The analyst will need to perform a ground truth check to see if the ***optimal solutions***

are valid. This problem will be minimized if the target of interest has a characteristic spectral signature that is different from all other objects in the image.

An obvious entry in Table 17 is run number one. With this run, only Channel 2 is being processed. By reviewing the feasible pixel regions in Appendix G, one begins to understand why the iteration count has suddenly exploded. Suddenly the program has to check nearly every pixel in the picture. With this increase in the number of possible solutions, the branch and bounding will also increase. Additional runs were made, but program continued to abort on the 5001st iteration[22].

It is difficult to measure the efficiency of the programs when they were run with different lambda weights. The GAMS output gives several measures of performance to compare the runs. The iterations and computer time usage (execution time and resource usage) varied as the weights changed. However there was not a direct correlation among these four measures of performance. For example, Run 5 had the lowest iteration count (for an objective function with two weights), but it did not have the lowest execution time or resource usage value. It is not known which execution time is a better measure of performance for the GAMS runs. The first execution time was typically given early in the GAMS output and the second execution time was the last piece of information given in the output. No units were given to the *Resource Usage* values so they can not be directly compared to the execution times. When running these programs, the wait for an answer was sometimes several minutes and at other times over an hour. The main factor in this wide range of waiting times is the load on VAX. As the number of users and jobs increase, the performance seen by the individual user will decrease. This variance in performance is characteristic of mainframes that are operated in a time sharing mode.

[22]Note the iteration limit for this program was set to 5,000. I ran several of the programs with higher iteration limits (7,000 – 8,500), but the program still failed to produce a noninferior solution.

Although the objective values and all other parameters measured in Table 17 had different values, there was no clear run which one was optimal in terms of computer performance. In terms of comparing all the solutions to the efficient frontier, all the solutions were equally optimal. The next set of GAMS runs will consist of Channels 1 and 3.

4.3.2.3 Channel 1 - Channel 3 Programs In this section, eleven Channel 1 - Channel 3 programs were run to locate the reflection pool identified earlier. The program remains the same as the Channel 1 - Channel 2 programs, but the Channel 3 gray values have been substituted for the Channel 2 gray values. The new water spectral bounds were set with two lines of GAMS code:

```
X.FX(I,"1")$(C(I,"A") LT 22) = 0;
X.FX(I,"1")$(C(I,"C") LT 5 OR C(I,"C") GT 200) = 0;
```

where the variable A represents Channel 1 and the C variable, Channel 3. The water region of interest remains set at a size of 24 pixels and a border length of 44. In Table 18 the results of varying the lambda weights are given.

The results of this program is comparable to the first set of runs. The same set of 24 pixel were again selected by all the runs except the first one. All these pixels were fully allocated in the solution. Once again since different subregions of 24-pixel-size were not selected, there is no variance in the optimal solution. Even though the objective function value changes, the last ten programs are all equally optimal.

4.3.2.4 Channel 2 - Channel 3 Programs The Channel 2 - Channel 3 GAMS programs were coded just like the other two channel combinations, but the output was very different. In fact, none of the eleven lambda combinations resulted in an final answer. Rather, each program stopped after 5001 iterations. In my

Table 18. Results of Channel 1 – Channel 3 Runs

GAMS Run	*Lambda Weights*		*Objective Value*	*Iteration Count*	*Execution Time (Sec)*		*Resource Usage*
1	0.0	1.0		*			
2	0.1	0.9	2253.2	1380	121.32	41.10	511.53
3	0.2	0.8	2010.4	1373	120.94	38.15	503.94
4	0.3	0.7	1767.6	1383	125.98	38.92	515.04
5	0.4	0.6	1524.8	1363	125.53	38.22	509.57
6	0.5	0.5	1282.0	1435	124.74	37.73	522.79
7	0.6	0.4	1039.2	1603	124.94	39.80	586.00
8	0.7	0.3	796.4	1495	124.06	39.57	522.00
9	0.8	0.2	553.6	1656	125.00	37.25	597.13
10	0.9	0.1	310.8	1292	124.93	39.74	476.45
11	1.0	0.0	68.0	1353	132.36	41.06	566.04

* No final solution after 5000 iterations.

GAMS programs, the iteration limit is set to 5000. On the 5001 first iteration, the programs aborted and output the current solution file. Many times the objective function was negative in value and the total number on pixels did not add up to 24. This high iteration count is the direct result of water not being spectrally distinct in both channels. In the other two-channel combinations, Channel 1 serves as a filter to reduce the number of possible solutions. Therefore, even though water is not spectrally distinct in Channels 2 or 3, Channel 1 filters out the majority of the undesired data. The difference between the feasible pixel regions is easily seen in Appendix G. In this appendix, the four major bodies of water are easily in Channel 1, but this is not true for the other channels.

4.3.3 Efficient Frontier Programs In the Channel 1–2 and 1–3 combinations there were no variations in the solution.[23] In each program, the same set of 24 pixels were selected even though the lambda weights were allowed to vary. It appears

[23]This statement holds true for all the lambda weight combinations except when Channel 2 or 3 were given a weight of one.

that with an area constraint of 24 pixels and a border length of 44, the model is not allowed to select any other solution of 24 pixels.[24] To see if other pixels can be selected and if an efficient frontier can be plotted, I will vary the border length of the target. The original B&W Model is a multicriteria objective function in itself. The area and border length constraints were originally in the combined objective function and subsequently placed in with the constraints. This method of subordinating an objective function into the constraints was described in Section 3.3.1.1. By varying the border constraint, other areas should be selected and an efficient frontier should result. To test if this hypothesis is correct, I will run programs where the border length and the lambda weights are allowed to vary.

In the following sections, I will attempt to still find an area of 24 pixels, but the border length will be allowed to vary. Recall in Chapter 3 that for a given size area, there is an acceptable range of border lengths. For an area of 24 pixels, the acceptable border length range is 36 – 192. Not every number in this range is acceptable, but rather every fourth number after the number 36 (i.e. 36, 40, 44, ..., 192). Therefore, there are only 41 possible border lengths for an area of 24 pixels. Rather than attempt to run all 41 border lengths and all lambda weights (451 programs), I will run several border lengths to get an idea of what the efficient frontier looks like. As the border length decreases from the maximum, I would expect the number of contiguous pixels to increase.

4.3.3.1 Border Length Of 192 In these first set of runs, the border length has been set to the maximum of 192. With this border length, each pixel has a border length of eight. The implication of this maximum border length is that the selected pixels are no longer required to be contiguous. Rather, 24 non-contiguous pixels could be selected if this combination maximized the objective function. The

[24]It should be noted that there are 26 contiguous pixels available for selection in that portion of the Tidal Basin contained in the subimage.

results of my model with a maximize objective function, area of 24, border length of 192, and varying lambda weights are shown in Table 19.

Table 19. Results of Channel 1 - Channel 3 Runs with 192 Border Length

Selected Pixels (x_{sk})	GAMS Computer Runs: Channel 1 - 3 Combination Lambda Weights (Measured in Tenths)										
	0/1.0	1/9	2/8	3/7	4/6	5/5	6/4	7/3	8/2	9/1	1.0/0
142†	¶	X	X	X	X	X		X			
163†		X	X	X	X	X	X	X	X	X	
184†		X	X	X	X	X	X	X	X	X	X
244†		X	X	X	X	X	X	X	X	X	X
262†		X	X	X	X	X	X	X			
264†		X	X	X	X	X	X	X	X	X	X
283†		X	X	X	X	X	X				
529*		X	X	X	X	X	X	X	X		
542§		X		X	X	X	X	X	X		X
547*											X
550*		X	X	X	X	X	X	X	X	X	
567*									X	X	X
568*		X	X	X	X	X	X	X	X		
570*		X	X	X	X	X	X	X	X	X	
587*									X	X	X
588*		X	X	X	X	X	X	X	X	X	X
589*		X	X	X	X	X	X	X	X	X	
590*		X	X	X	X	X	X	X	X	X	
594⋆											X
608*		X	X	X	X	X	X	X	X	X	X
609*			X								
610*		X	X	X	X	X	X	X	X		
622§										X	X
628*											X
630*			X				X	X	X		
648*											X
665⋆										X	X
686⋆										X	X
706⋆										X	X
710⋆											X
879⋆										X	X
935‡		X	X	X	X	X	X	X	X	X	X
936‡		X	X	X	X	X	X	X	X	X	X
937‡		X	X	X	X	X	X	X	X	X	
938‡		X	X	X	X	X	X	X	X	X	X
939‡		X	X	X	X	X	X	X	X	X	X
951‡		X	[illegible]	X	X	X	X	X	X	X	X
968‡						X	X	X	X	X	X
974‡		X		X	X						
Obj Value		3174.0	2838.4	2502.0	2166.0	1831.0	1499.2	1167.4	844.4	554.4	339.0

¶No final solution after 5000 iterations. †= Constitution Gardens Lake. * = Reflection Pool.
§= Lincoln Reflection Pool. ⋆ = Noise Type Pixels. ‡= Tidal Basin.

In Table 19, the lambda weights are designated with a #/# label. The first number represents the lambda weight measured in tenths for objective function 1 (Channel 1) and the second number represents the lambda weight for the second objective function (Channel 3). This labelling holds true for all the computer runs except the first and the last. On these two runs, the fully weighted channel is labeled with a 1.0. This distinction is made to avoid confusion between 0.1 and 1.0. In all except the first run, the pixels selected were fully allocated. The label

Obj Value represents the combined objective value for Channels 1 and 3. All of the computer programs were successfully run except the first one (0/1.0). On this run, the program aborted after 5,000 iterations. This problem was the result of the feasible region being too large, as was previously explained. This problem continued to exist in all the other border length programs for the Channel 1–3 combination.

As the lambda weight for Channel 1 increases from 0 to 1.0, a trend is seen. In general the selected pixels are migrating away from the Constitution Gardens Lake and are reselected in the Reflection Pool and Noise-Type pixels. From computer runs 1/9 to 6/4, the pixels selected stay fairly constant, but in the subsequent lambda weights the change in pixel selection is more noticeable. The combined objective value changed as the lambda weights changes. The concept of an efficient frontier is beginning to come into play since the 24 pixels selected change as the lambda weights change. One final observation about these solutions is that the pixels in the Tidal Basin area remained fairly fixed. In the next set of runs, the border length is changed to 100 while the area remains fixed.

4.3.3.2 Border Length Of 100 In this second set of runs, the border length has been set to 100. With this border length, the pixels are still not forced to be all contiguous. The results of my model with a maximize objective function, area of 24, border length of 100, and varying lambda weights are shown in Table 20.

In these computer runs, the most noticeable difference from previous runs is the noninteger selection of the x_{ik}'s. In Table 20 the noninteger pixels are allocated with a decimal number. This number indicates what percentage of that pixel was allocated to the selected subregion. Up until this point, all the solutions have been integer. These results are significant since all of the GAMS programs in this thesis were run as a mixed integer program. In addition, the x_{ik}'s are assigned as binary variables in the code. Even though this was true, this did not prevent the noninteger solutions from occurring. Additional runs will be made to see if this trend continues

Table 20. Results of Channel 1 – Channel 3 Runs with 100 Border Length

Selected Pixels (x_{ijk})	GAMS Computer Runs: Channel 1 – 3 Combination Lambda Weights (Measured in Tenths)										
	0/1.0	1/9	2/8	3/7	4/6	5/5	6/4	7/3	8/2	9/1	1.0/0
163†	¶	X	X	X	X	X	X	X			
184†		X	X	X	X	X	X	X	X	X	0.556
244†		X	X	X	X	X	X	X	X	X	
264†		X	X	X	X	X	X	X	X	X	X
528*		X	X	X	X	0.7	0.3	X	0.222		
529*		X	X	X	X	0.7	0.3	X	0.222		
542§			X	X	X	0.5	X	X	X	X	X
547*			X	X	X	X			0.222	0.333	X
548*		X	X	X	X	0.7	0.3		0.222	0.333	X
549*		0.571	0.571	0.571	0.571	0.7	0.3	0.222		0.333	
550*		X	X	X	X	X	X	X	X	X	
567*								X	X	X	X
568*		X	X	X	X	X	X	X	X	X	X
569*		0.571	0.571	0.571	0.571	0.7	0.3	X	X	X	
570*		X	X	X	X	X	X	X	X	X	
587*								X	X	X	X
588*		X	X	X	X	X	X	X	X	X	X
589*		X	X	X	X	X	X	X	X	X	
590*		X	X	X	X	X	X	X	X	X	
607*											X
608*		X	X	X	X	X	X	X	X	X	X
609*		X	X	X	X	X	X	X	X	X	
610*		X	X	X	X	X	X	X	X	X	
628*		X	X	X	X	X	X	0.4			
629*		X	X	X	X	X	X	0.4			
630*		X	X	X	X	X	X	0.4			
665⋆											X
686⋆											X
706⋆											X
879⋆											X
935‡		X	X	X	X	X	X	X	X	X	X
936‡		X	X	X	X	X	X	X	X	X	X
937‡		X	X	X	X	X	X	X	X	X	X
938‡		X	X	X	X	X	X	X	X	X	X
939‡		X	X	X	X	X	X	X	X	X	X
951‡		0.857	0.857	0.857	0.857	X	0.5	0.8	.889	X	
955‡											0.444
956‡											0.444
957‡											0.444
958‡											0.444
959‡											0.444
968‡									X	X	
975‡											0.444
976‡											0.444
977‡											0.444
978‡											0.444
979‡											0.444
Obj Value		3284.4	2933.8	2583.2	2232.7	1882.6	1456.4	1131.5	817.4	512.0	.0

¶No final solution after 5000 iterations. † = Constitution Gardens Lake. * = Reflection Pool.
§ = Lincoln Reflection Pool. ⋆ = Noise Type Pixels. ‡ = Tidal Basin.

and if it does a hypothesis will be offered as to why these noninteger results are occurring.

In general, the pixels selected in these runs remained fairly constant with the pixels divided over the Constitution Lake, the Reflection Pool, and the Tidal Basin. The most noticeable change in pixel selection occurred on the last run (1.0/0). In this solution, many more Tidal Basin pixels were partially allocated to the water subregion. Due to the relatively loose border, the pixels are not required to be contiguous. For this reason, the pixels are mainly allocated in the three areas mentioned previously.

4.3.3.3 Border Length Of 64 In this third set of runs, the border length has been set to 64. With this border length, more of the pixels should be forced to be contiguous since the border length is decreasing. The results of my model with a maximize objective function, area of 24, border length of 64, and varying lambda weights are shown in Table 21.

In these solutions, more of the pixels were contiguous as was hypothesized before the runs. Since the border length has decreased, the selected pixels are forced to be more contiguous than previous runs. It is possible that the border length shortening has also caused the majority of the solutions to once again be integer. Another possibility is that there is a natural geometric shapes in the two images for this area of 24 and border length of 64. In the next run, the border length will again be shorten to see what the affect is on the solution.

4.3.3.4 Border Length Of 60 In this fourth set of runs, the border length has been set to 60. I would expect more pixels to be contiguous in these runs. The results of my model with a maximize objective function, area of 24, border length of 60, and varying lambda weights are shown in Table 22.

Table 21. Results of Channel 1 – Channel 3 Runs with 64 Border Length

Selected Pixels (x_{ik})	GAMS Computer Runs: Channel 1 – 3 Combination Lambda Weights (Measured in Tenths)										
	0/1.0	1/9	2/8	3/7	4/6	5/5	6/4	7/3	8/2	9/1	1.0/0
244†	¶	0.143	0.143								
264†		0.143	0.143								
528*		X	X	X	X	X	X	X	X		
529*		X	X	X	X	X	X	X	X		
542§											X
547*		0.571	0.571					X	X	X	X
548*		X	X	X	X	X	X	X	X	X	X
549*		X	X	X	X	X	X	X	X	X	X
550*		X	X	X	X	X	X	X	X	X	X
567*		0.571	0.571					X	X	X	X
568*		X	X	X	X	X	X	X	X	X	X
569*		X	X	X	X	X	X	X	X	X	X
570*		X	X	X	X	X	X	X	X	X	X
587*		0.571	0.571					X	X	X	X
588*		X	X	X	X	X	X	X	X	X	X
589*		X	X	X	X	X	X	X	X	X	X
590*		X	X	X	X	X	X	X	X	X	X
607*											X
608*		X	X	X	X	X	X	X	X	X	X
609*		X	X	X	X	X	X	X	X	X	X
610*		X	X	X	X	X	X	X	X	X	X
627*											X
628*		X	X	X	X	X	X	X	X	X	X
629*		X	X	X	X	X	X	X	X	X	X
630*		X	X	X	X	X	X	X	X	X	X
686⋆											X
706⋆											X
879⋆											X
935‡		X	X	X	X	X	X	X	X	X	
936‡		X	X	X	X	X	X	X	X	X	
937‡		X	X	X	X	X	X			X	
938‡		X	X	X	X	X	X	X	X	X	
939‡		X	X	X	X	X	X	X	X	X	
958‡				X	X	X	X				
959‡				X	X	X	X			X	
Obj Value		2839.1	2534.9	2197.8	1894.4	1591.0	1287.6	993.1	699.4	406.8	144.0

¶No final solution after 5000 iterations. †= Constitution Gardens Lake. * = Reflection Pool.
§= Lincoln Reflection Pool. ⋆ = Noise Type Pixels. ‡= Tidal Basin.

Table 22. Results of Channel 1 – Channel 3 Runs with 60 Border Length

Selected Pixels (x_{ik})	GAMS Computer Runs: Channel 1 – 3 Combination Lambda Weights (Measured in Tenths)										
	0/1.0	1/9	2/8	3/7	4/6	5/5	6/4	7/3	8/2	9/1	1.0/0
528*	¶	X		X	X	X	X	X	X		
529*		X		X	X	X	X	X	X		
542§											X
547*				0.923	0.846	0.846	0.846	X	X	0.947	X
548*		X	X	X	X	X	X	X	X	0.947	X
549*		X	X	X	X	X	X	X	X	0.947	X
550*		X	X	X	X	X	X	X	X	0.947	X
567*			X	0.923	0.846	0.846	0.846	X	X	X	X
568*		X	X	X	X	X	X	X	X	X	X
569*		X	X	X	X	X	X	X	X	X	X
570*		X	X	X	X	X	X	X	X	X	X
587*			X	0.923	0.846	0.846	0.846	X	X	X	X
588*		X	X	X	X	X	X	X	X	X	X
589*		X	X	X	X	X	X	X	X	X	X
590*		X	X	X	X	X	X	X	X	X	X
607*			X								X
608*		X	X	X	X	X	X	X	X	X	X
609*		X	X	X	X	X	X	X	X	X	X
610*		X	X	X	X	X	X	0.5	0.5	X	X
627*			X								X
628*		X	X	X	X	X	X		0.5	X	X
629*		X	X	X	X	X	X		0.5	X	X
630*		X	X	X	X	X	X			X	X
648*		X			X	X	X		0.5	X	X
649*		X							0.5	X	
686⋆											X
706⋆											X
935‡		0.789	X	X	0.692	0.692	0.692	0.763	0.711	0.842	
936‡		0.789	X	X	0.692	0.692	0.692	0.763	0.711	0.842	
937‡		0.789	X	X	0.692	0.692	0.692	0.763	0.711	0.842	
938‡		0.789	X	X	0.692	0.692	0.692	0.763	0.711	0.842	
939‡		0.789	X	X	0.692	0.692	0.692	0.763	0.711	0.842	
956‡		0.132						0.461	0.243		
957‡		0.132						0.461	0.243		
958‡		0.132						0.461	0.243		
959‡		0.132						0.461	0.243		
976‡		0.132						0.461	0.243		
977‡		0.132						0.461	0.243		
978‡		0.132						0.461	0.243		
979‡		0.132						0.461	0.243		
Obj Value		2705.8	2361.3	2163.6	1824.8	1537.1	1249.3	993.9	656.3	394.5	128.0

¶No final solution after 5000 iterations. †= Constitution Gardens Lake. * = Reflection Pool.
§= Lincoln Reflection Pool. ⋆ = Noise Type Pixels. ‡= Tidal Basin.

In these runs, the solution were primarily noninteger in nature. Only the third (2/8) and last (1.0/0) runs resulted in integer solutions. In these runs, the pixels selected varied greatly, but some of the solutions were repeated (e.g. runs 4/6–6/4). These runs show that shortening the border length alone will not directly result in integer solutions. In the final set of runs, the border has been further shortened to 52.

4.3.3.5 Border Length Of 52 In this final set of runs, the border length has been set to 52. The results of my model with a maximize objective function, area of 24, border length of 52, and varying lambda weights are shown in Table 23.

In these runs, the pixels selected remained fairly constant as the lambda weights were changed. However, the decimal allocation of the Tidal Basin pixels varied in runs 1/9 to 9/1. Only the final run (1.0/0) resulted in an integer solution. Given these noninteger solutions, an explanation is needed. In the next paragraph, the concept of an efficient frontier is applied to these sets of runs. In the following section a hypothesis is stated for the cause of this problem and computer programs are run to test the hypothesis.

4.3.3.6 Efficient Frontier From all the computer sets run, the best data for plotting an efficient frontier is from the runs with a border length of 192. In these runs all lambda weight combinations produce integer solutions, except the 0/1.0 combination (See Table 19. This particular program aborted after 5,000 iterations. In Table 24, the points for the outcome space is shown. The efficient frontier generated from these points is shown in Figure 34.

4.3.3.7 Inequality Border Length Programs In the previous set of computer runs, some of the outputs contained noninteger solutions. In these solutions, some of the pixels were not fully allocated (indicated by a decimal number rather than an X in the tables). One reason why these noninteger solutions may result is

Table 23. Results of Channel 1 – Channel 3 Runs with 52 Border Length

Selected Pixels (x_{ik})	GAMS Computer Runs: Channel 1 – 3 Combination Lambda Weights (Measured in Tenths)										
	0/1.0	1/9	2/8	3/7	4/6	5/5	6/4	7/3	8/2	9/1	1.0/0
528*	¶	X	X	X	X	X	X	X	X	X	
529*		X	X	X	X	X	X	X	X		
547*		0.091	0.091	X	X	X	X	X	X	X	X
548*		X	X	X	X	X	X	X	X	X	X
549*		X	X	X	X	X	X	X	X	X	X
550*		X	X	X	X	X	X	X	X	X	X
567*		0.091	0.091	X	X	X	X	X	X	X	X
568*		X	X	X	X	X	X	X	X	X	X
569*		X	X	X	X	X	X	X	X	X	X
570*		X	X	X	X	X	X	X	X	X	X
587*		0.091	0.091	X	X	X	X	X	X	X	X
588*		X	X	X	X	X	X	X	X	X	X
589*		X	X	X	X	X	X	X	X	X	X
590*		X	X	X	X	X	X	X	X	X	X
607*											X
608*		X	X	X	X	X	X	X	X	X	X
609*		X	X	X	X	X	X	X	X	X	X
610*		X	X	X	X	X	X	X	X	X	X
627*										X	X
628*		X	X	X	X	X	X	X	X	X	X
629*		X	X	X	X	X	X	X	X	X	X
630*		X	X	X	X	X	X	X	X	X	X
648*		X	X	X	X	X	X	X	X		X
649*		X	X	X	X	X	X	X	X		X
686⋆											X
706⋆											X
935‡		0.364	0.364	0.316	0.316	0.316	0.316	0.316	0.316	0.316	
936‡		0.364	0.364	0.316	0.316	0.316	0.316	0.316	0.316	0.316	
937‡		0.364	0.364	0.316	0.316	0.316	0.316	0.316	0.316	0.316	
938‡		0.364	0.364	0.316	0.316	0.316	0.316	0.316	0.316	0.316	
939‡		0.364	0.364	0.316	0.316	0.316	0.316	0.316	0.316	0.316	
955‡										0.	
956‡		0.364	0.364	0.053	0.053	0.053	0.053	0.053	0.053	0.053	
957‡		0.364	0.364	0.053	0.053	0.053	0.053	0.053	0.053	0.053	
958‡		0.364	0.364	0.053	0.053	0.053	0.053	0.053	0.053	0.053	
959‡		0.364	0.364	0.053	0.053	0.053	0.053	0.053	0.053	0.053	
975‡										0.	
976‡		0.364	0.364	0.053	0.053	0.053	0.053	0.053	0.053	0.053	
977‡		0.364	0.364	0.053	0.053	0.053	0.053	0.053	0.053	0.053	
978‡		0.364	0.364	0.053	0.053	0.053	0.053	0.053	0.053	0.053	
979‡		0.364	0.364	0.053	0.053	0.053	0.053	0.053	0.053	0.053	
Obj Value		2506.2	2233.8	1961.3	1693.2	1425.1	1157.1	889.0	620.9	338.5	106.0

¶No final solution after 5000 iterations. †= Constitution Gardens Lake. * = Reflection Pool.
§= Lincoln Reflection Pool. ⋆ = Noise Type Pixels. ‡= Tidal Basin.

Table 24. Efficient Frontier Points For Border Length = 192

Optimal Points	*Lambda Weights*		*Outcome Space*	
	λ_1	λ_2	f_1	f_2
A'	0	1.0	*	
B'	0.1	0.9	15.0	3159.0
C'	0.2	0.8	25.6	2812.8
D'	0.3	0.7	45.0	2457.0
E'	0.4	0.6	60.0	2106.0
F'	0.5	0.5	86.0	1745.0
G'	0.6	0.4	103.2	1396.0
H'	0.7	0.3	120.4	1047.0
I'	0.8	0.2	163.2	681.2
J'	0.9	0.1	271.8	282.6
K'	1.0	0	339.0	0
** No final solution after 5,000 iterations.*				

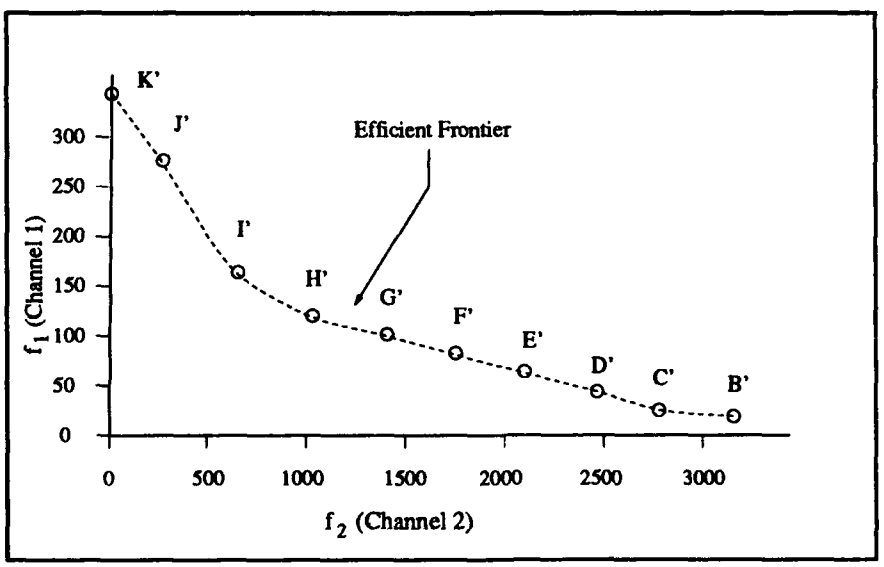

Figure 34. Efficient Frontier For Border Length = 192

that the feasible regions may not contain a water subregion of 24 pixels with the designated border length. In other words, each border length requires a certain geometric shape for given size area. If this geometric shape is not present in the image, the objective function may be over riding the x_{ik} binary variable constraint. If this is true, more than 24 pixels could be selected to fulfill the border length constraint. To test this hypothesis, I ran another set of programs where an upper limit is set on the border length. With this formulation, the model is allowed pick the border length that maximizes the objective function. Since the border constraint is an inequality rather than an equality, the model is not forced into only selecting a particular geometric shape for a given size area. In these computer runs, the Channel 1–2 combination is used.

Border Length $\leq$ 60 In this first set of runs, the border length has been set to less than or equal to 60. Therefore, the model is allowed to select border lengths that range from 36 (minimum border length) to 60. With this border length, the pixels are not forced to all be contiguous. The results of my model with a maximize objective function, area of 24, border length of less than for equal to 60, and varying lambda weights are shown in Table 25.

The results for these runs were surprising. I expected integer answers since the border length was allowed to vary. Rather, the second (1/9) through eighth (7/3) resulted in noninteger solutions. Only the last three runs resulted in integer answers. In order to give the model more flexibility in selecting the water pixels, the border length was increased to 100. These runs are discussed in the next paragraph.

Border Length $\leq$ 100 In this second set of runs, the border length has been set to less than 100. With this increased border length, the pixels selected can be more spread out. The results of my model with a maximize objective function, area of 24, border length of less than for equal to 100, and varying lambda weights are shown in Table 26.

Table 25. Results of Channel 1 – Channel 2 Runs; Border Length ≤ 60

Selected Pixels (x_{ik})	GAMS Computer Runs: Channel 1 – 2 Combination Lambda Weights (Measured in Tenths)										
	0/1.0	1/9	2/8	3/7	4/6	5/5	6/4	7/3	8/2	9/1	1.0/0
528	¶	X	X	X	X	X	X	X			
529		X	X	X	X	X	X	X			
542											X
547		X	X	X	X	X	X	X	X	X	X
548		X	X	X	X	X	X	X	X	X	X
549		X	X	X	X	X	X	X	X	X	X
550		X	X	X	X	X	X	X	X	X	X
567		X	X	X	X	X	X	X	X	X	X
568		X	X	X	X	X	X	X	X	X	X
569		X	X	X	X	X	X	X	X	X	X
570		X	X	X	X	X	X	X	X	X	X
587		X	X	X	X	X	X	X	X	X	X
588		X	X	X	X	X	X	X	X	X	X
589		X	X	X	X	X	X	X	X	X	X
590		X	X	X	X	X	X	X	X	X	X
607						X	X	X	X	X	X
608		X	X	X	X	X	X	X	X	X	X
609		X	X	X	X	X	X	X	X	X	X
610		X	X	X	X	X	X	X	X	X	X
627											X
628						0.462	0.462	0.462	X	X	X
629		X	X	X	X	0.462	0.462	0.462	X	X	X
630		X	X	X	X	0.462	0.462	0.462	X	X	
648											X
649											X
686											X
706											X
935		0.789	0.789	0.789	0.789	0.923	0.923	0.923	X	X	
936		0.789	0.789	0.789	0.789	0.923	0.923	0.923	X	X	
937		0.789	0.789	0.789	0.789	0.923	0.923	0.923	X	X	
938		0.789	0.789	0.789	0.789	0.923	0.923	0.923	X	X	
939		0.789	0.789	0.789	0.789	0.923	0.923	0.923	X	X	
956		0.132	0.132	0.132	0.132						
957		0.132	0.132	0.132	0.132						
958		0.132	0.132	0.132	0.132						
959		0.132	0.132	0.132	0.132						
976		0.132	0.132	0.132	0.132						
977		0.132	0.132	0.132	0.132						
978		0.132	0.132	0.132	0.132						
979		0.132	0.132	0.132	0.132						
Obj Value		2099.8	1877.7	1655.6	1433.5	1218.2	996.4	774.7	547.4	331.2	128.0

¶No final solution after 5000 iterations. †= Constitution Gardens Lake. * = Reflection Pool.
§= Lincoln Reflection Pool. ⋆ = Noise Type Pixels. ‡= Tidal Basin.

Table 26. Results of Channel 1 – Channel 2 Runs; Border Length ≤ 100

Selected Pixels (x_{1k})	GAMS Computer Runs: Channel 1 – 2 Combination Lambda Weights (Measured in Tenths)										
	0/1.0	1/9	2/8	3/7	4/6	5/5	6/4	7/3	8/2	9/1	1.0/0
163†	¶	X	X	X	X	X					
184†		X	X	X	X	X	X	X	X	X	
244†		X	X	X	X	X	X	X	X	X	
264†		X	X	X	X	X	X	X	X	X	X
528*		X	X	X	X	X	X	0.333			
529*		X	X	X	X	X	X	0.333			
542⋆		X	X	0.333	X	X	X	X	X	X	X
547*								0.333		X	X
548*		X	X	0.556	X	X	X	0.333		X	X
549*		X	X	0.556	X	X	X	0.333		X	
550*		X	X	X	X	X	X	X	X	X	
567*								X	X	X	X
568*		X	X	X	X	X	X	X	X	X	X
569*		X	X	0.556	X	X	X	0.333		X	
570*		X	X	X	X	X	X	X	X	X	
587*								X	X	X	
588*		X	X	X	X	X	X	X	X	X	
589*		X	X	X	X	X	X	X	X	X	
590*		X	X	X	X	X	X	X	X	X	
607*											X
608*		X	X	X	X	X	X	X	X	X	X
609*				X				X	X	X	
610*		X	X	X	X	X	X	X	X	X	
628*									X		
629*									X		
630*		X	X	X	X	X	X	X	X		
665⋆											X
686⋆											X
879⋆										X	X
935‡		X	X	X	X	X	X	X	X	X	X
936‡		X	X	X	X	X	X	X	X		X
937‡		X	X	X	X	X	X	X	X		X
938‡		X	X	X	X	X	X	X	X		X
939‡		X	X	X	X	X	X	X	X	X	X
951‡							X			X	
955‡											X
956‡											X
957‡				X							
958‡											X
959‡											X
968‡								X	X	X	
Obj Value		2494.1	2232.2	1966.6	1708.4	1446.5	1185.6	940.7	684.2	432.8	238.0

¶No final solution after 5000 iterations. †= Constitution Gardens Lake. * = Reflection Pool.
§= Lincoln Reflection Pool. ⋆ = Noise Type Pixels. ‡= Tidal Basin

With the increased flexibility in selecting water pixels, this set of runs resulted in primarily integer solutions. Since the border length has become less of a constraint, the model can more easily select integer solutions. Overall, the pixels selected did not change much from run to run. The biggest shift in selected pixels occurred between run 10 (9/1) and the last (1.0/0) run. In next paragraph, a final set of runs are made with a border of less than or equal 196.

Border Length $\leq$ 196 In this final set of runs, the border length has been set to less than or equal 196. With this border length, the selected pixels could all individual pixels in the image since this is above the maximum border length for an area of 24 pixels.[25] I would suspect that all of the solutions will be integer. The results of my model with a maximize objective function, area of 24, border length of less than for equal to 196, and varying lambda weights are shown in Table 27.

In this final set of runs, all the solutions were integer except the tenth run (9/1). I can not explain why the model did not pick the top 24 pixels to maximize the combined objective function. With a border length of 196, the model is not constrained by the border length. This is true since 196 is greater than the maximum border length (192). The biggest surprise in these runs was the successful completion of the first run (0/1.0). Most of the pixels selected by this run are non contiguous and are not located in the water subregion areas. In fact only two pixels selected were in areas previously associated with water. Pixel 264 was located in the Constitution Lake area and pixel 666 was a water noise-type pixel near the Reflection Pool. For this run the iteration count was 4799. This is high when compared to the rest of this set of runs. All the other runs were bounded by a low of 2,388 iterations for the fourth run (3/7) to a high of 3,075 for the last run (1.0/0).

[25] Recall that the maximum border length for an area of 24 pixels is 192.

Table 27. Results of Channel 1 – Channel 2 Runs; Border Length $\leq$ 196

Selected Pixels (x_{ik})	GAMS Computer Runs: Channel 1 – 2 Combination Lambda Weights (Measured in Tenths)										
	0/1.0	1/9	2/8	3/7	4/6	5/5	6/4	7/3	8/2	9/1	1.0/0
32•	X										
37•	X										
102•	X										
122•	X										
142†		X	X	X	X	X	X				
163†		X	X	X	X	X	X	X	X		
184†		X	X	X	X	X	X	X	X	X	X
186•	X										
187•	X										
244†		X	X	X	X	X	X	X	X	X	X
264†	X	X	X	X	X	X	X	X	X	X	X
342•	X										
522•	X										
528*		X	X	X	X	X	X	X	X		
529*		X	X	X	X	X	X	X	X		
542§		X	X	X	X	X	X	X	X	X	X
547*										X	X
550*		X	X	X	X	X	X	X	X	X	
552•	X										
555•	X										
557•	X										
563•	X										
567*									X	X	X
568*		X	X	X	X	X	X	X	X		
570*		X	X	X	X	X	X	X	X		
587*								X	X	X	X
588*		X	X	X	X	X	X	X	X	X	X
589*		X	X	X	X	X	X	X			
590*		X	X	X	X	X	X	X	X	0.5	
594⋆											X
606•	X										
608*		X	X	X	X	X	X	X	X	X	X
610*		X	X	X	X	X	X	X	X	0.5	
622§										X	X
628*											X
630*		X	X	X	X	X	X				
648*											X
665⋆										X	X
666⋆	X										
670•	X										
686⋆										X	X
696•	X										
698•	X										
706⋆										X	X
710⋆										X	X
879⋆									X	X	X
912•	X										
918•	X										
928•	X										
931•	X										
934•	X										
935‡		X	X	X	X	X	X	X	X	X	X
936‡		X	X	X	X	X	X	X	X	X	X
937‡		X	X	X	X	X	X	X	X	X	
938‡		X	X	X	X	X	X	X	X	X	X
939‡		X	X	X	X	X	X	X	X	X	X
947•	X										
951‡		X	X	X	X	X	X	X	X	X	X
968‡		X	X	X	X	X	X	X	X	X	X
974‡								X			
Obj Value	3720.0	2618.2	22346.4	2074.6	1802.8	1531.0	1255.6	984.8	725.4	509.4	339.0

• = Pixels located in nonwater areas. † = Constitution Gardens Lake. * = Reflection Pool.
§ = Lincoln Reflection Pool. ⋆ = Noise Type Pixels. ‡ = Tidal Basin.

4.3.3.8 Varying Area and Border Length Programs In the previous computer programs, the border length was allowed to vary in order to allow the model more flexibility in selecting pixels. It was hoped that integer solutions would result, but this was not always true. In this section, the area and border lengths are allowed to vary. Once again, I am attempting to get all integer answers. In the first set of runs, the area and border length are set less than a given value. In the next set of runs, I use two area sizes and two border lengths. In this set of four runs, I attempt to enforce contiguity by maximizing area and minimizing the border length. These last sets of runs are shown in Table 28 below. In these runs, I have used the Channel 1–3 combination.

Table 28. Varying Area and Border Length Runs

Border Length	*Area Size*	
	≥ 24	≥ 26
≤ 48	λ_1, λ_2	λ_1, λ_2
≤ 64	λ_1, λ_2	λ_1, λ_2

Area $\leq$ 28 & Border Length $\leq$ 64 In this set of runs, I have allowed the model to pick an area less than or equal to 28 pixels in size. The border length is allowed to vary up to 64 in length. The results of this these runs will be split over more than one area since the two largest bodies of water in the subimage are 24 and 26 pixels in size. The model has the flexibility to split over to areas since the border length is sufficiently long enough. The results from these runs are shown in Table 29.

In these runs, several interesting results occurred. The first noticeable result is that all except the sixth run were integer solutions[26]. I attribute this to the

[26]Note: Runs 0/1.0 and 1.0/0 were not run since one of the programs (5/5) was not integer. If the solutions are not all integer, they can not be directly compared.

Table 29. Results of Channel 1 - Channel 3 Runs; Area $\leq$ 28, Border Length $\leq$ 64

Selected Pixels (x_{ik})	GAMS Computer Runs: Channel 1 - 3 Combination Lambda Weights (Measured in Tenths)										
	0/1.0	1/9	2/8	3/7	4/6	5/5	6/4	7/3	8/2	9/1	1.0/0
528*	¶					X	X				¶
529*						X	X				
547*						X				X	
548*			X	X	X	X	X			X	
549*			X	X	X	X	X			X	
550*			X	X	X	X	X			X	
567*						X		X		X	
568*		X	X	X	X	X	X	X	X	X	
569*		X	X	X	X	X	X	X	X	X	
570*		X	X	X	X	X	X	X	X	X	
587*		X				X		X	X	X	
588*		X	X	X	X	X	X	X	X	X	
589*		X	X	X	X	X	X	X	X	X	
590*		X	X	X	X	X	X	X	X	X	
607*								X			
608*		X	X	X	X	X	X	X	X	X	
609*		X	X	X	X	X	X	X	X	X	
610*		X	X	X	X	X	X	X	X	X	
628*		X	X	X	X	X	X	X	X		
629*		X	X	X	X	X	X	X	X		
630*		X	X	X	X	X	X	X	X		
935‡		X	X	X	X	0.684	X	X	X	X	
936‡		X	X	X	X	0.684	X	X	X	X	
937‡		X	X	X	X	0.684	X	X	X	X	
938‡		X	X	X	X	0.684	X	X	X	X	
939‡		X	X	X	X	0.684	X	X	X	X	
955‡		X							X		
956‡		X	X	X	X	0.447		X	X	X	
957‡		X	X	X	X	0.447		X	X	X	
958‡		X	X	X	X	0.447	X	X	X	X	
959‡		X	X	X	X	0.447	X	X	X	X	
975‡		X							X		
976‡		X	X	X	X	0.447		X	X	X	
977‡		X	X	X	X	0.447		X	X	X	
978‡		X	X	X	X	0.447		X	X	X	
979‡		X	X	X	X	0.447		X	X	X	
Obj Value		2759.7	2660.4	2337.1	2013.8	1648.1184	1287.6	1009.9	685.6	422.9	.0

¶These runs were not run. †= Constitution Gardens Lake. * = Reflection Pool.
§= Lincoln Reflection Pool. ⋆ = Noise Type Pixels. ‡= Tidal Basin.

model having enough flexibility to select those pixels which maximize the objective function. The model is now required to pick the area and border length. The second interesting result is related to this flexibility. All the runs had a 28-pixel size solution, except the seventh run (6/4). In this run, the model chose an area of 24 pixels. In all the solutions, the pixels selected were divided between the Reflection Pool and the Tidal Basin.

Area $\geq$ 24 & Border Length $\leq$ 48 This set of runs begins the set of four programs where the area and border length are both bounded by two sizes. In the first set of runs, the area has be set greater than or equal to 24 and the border length has been set less than or equal to 48. Given then parameters, I would expect the model to select the Reflection Pool in total. Recall that the Reflection Pool had an area of 24 and a border length of 44. The results for these runs are shown in Table 30.

These set of runs had several significant results. The first, is that instead of selecting the Reflection Pool in total, the model selected pixels from here and from the Tidal Basin. However, the model was only able to select fractional pixels in the Tidal Basin area. The second significant result is that the pixels selected stayed nearly constant from run to run. A final observation about these runs is that the total area for a run was also fractional. All of runs 1/9 to 8/1 had a total area of 24.75, but run 9/1 had an area of 24.75.

Area $\geq$ 24 & Border Length $\leq$ 64 In this set of runs, the area remains set at greater than or equal to 24 and the border length has been changed from less than or equal to 48 to less than or equal to 64. Given these parameters, I would expect the solution to be more spread out due to the increased border length. The results for these runs are shown in Table 31.

Contrary to what I predicted, the pixels selected were not spread. This problem arose since the area size constraint was not maintained in runs 1/9 - 7/3. In these

Table 30. Results of Channel 1 – Channel 3 Runs; Area ≥ 24, Border Length ≤ 48

Selected Pixels (x_{ik})	GAMS Computer Runs: Channel 1 – 3 Combination Lambda Weights (Measured in Tenths)										
	0/1.0	1/9	2/8	3/7	4/6	5/5	6/4	7/3	8/2	9/1	1.0/0
528*	¶	X	X	X	X	X	X	X	X		
529*		X	X	X	X	X	X	X	X		
547*		X	X	X	X	X	X	X	X	X	X
548*		X	X	X	X	X	X	X	X	X	X
549*		X	X	X	X	X	X	X	X	X	X
550*		X	X	X	X	X	X	X	X	X	X
567*		X	X	X	X	X	X	X	X	X	X
568*		X	X	X	X	X	X	X	X	X	X
569*		X	X	X	X	X	X	X	X	X	X
570*		X	X	X	X	X	X	X	X	X	X
587*		X	X	X	X	X	X	X	X	X	X
588*		X	X	X	X	X	X	X	X	X	X
589*		X	X	X	X	X	X	X	X	X	X
590*		X	X	X	X	X	X	X	X	X	X
607*		X	X	X	X	X	X	X	X	X	X
608*		X	X	X	X	X	X	X	X	X	X
609*		X	X	X	X	X	X	X	X	X	X
610*		X	X	X	X	X	X	X	X	X	X
627*											X
628*		X	X	X	X	X	X	X	X	X	X
629*		X	X	X	X	X	X	X	X	X	X
630*		X	X	X	X	X	X	X	X	X	X
648*											X
649*											X
686*											0.483
706*											0.483
935‡		0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.375	0.069
936‡		0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.375	0.069
937‡		0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.375	0.069
938‡		0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.375	0.069
939‡		0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.375	0.069
955‡		0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.375	0.069
956‡		0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.375	0.069
957‡		0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.375	0.069
958‡		0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.375	0.069
959‡		0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.375	0.069
975‡		0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.375	0.069
976‡		0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.375	0.069
977‡		0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.375	0.069
978‡		0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.375	0.069
979‡		0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.375	0.069
Obj Value		2930	2185.3	1921.7	1657.8	11394.0	1130.3	866.5	602.8	339.7	90.0
Iterations		2930	3066	3088	2988	2883	2937	2936	2980	3199	

¶No final solution after 5000 iterations. † = Constitution Gardens Lake. * = Reflection Pool.
§ = Lincoln Reflection Pool. ⋆ = Noise Type Pixels. ‡ = Tidal Basin.

Table 31. Results of Channel 1 – Channel 3 Runs; Area $\geq$ 24, Border Length $\leq$ 64

Selected Pixels (x_{ik})	GAMS Computer Runs: Channel 1 – 3 Combination Lambda Weights (Measured in Tenths)										
	0/1.0	1/9	2/8	3/7	4/6	5/5	6/4	7/3	8/2	9/1	1.0/0
264†	¶										X
547*					X					X	X
548*				X						X	X
549*										X	X
550*										X	X
567*									X	X	X
568*									X	X	X
569*									X	X	X
570*									X	X	X
587*			X						X	X	X
588*									X	X	X
589*									X	X	X
590*									X	X	X
607*									X	X	X
608*									X	X	X
609*									X	X	X
610*									X	X	X
627*									X		X
628*									X		X
629*									X		X
630*									X		X
686⋆											X
706⋆											X
789⋆											X
935‡									X	X	
936‡		0.75	0.75	0.75	0.75	0.80	0.75	0.80	X	X	
937‡		0.75	0.75	0.75	0.75	0.80	0.75	0.80	X	X	
938‡		0.75	0.75	0.75	0.75	0.80	0.75	0.80	X	X	
939‡		0.75	0.75	0.75	0.75	0.80	0.75	0.80	X	X	
955‡									X	X	
956‡		0.75	0.75	0.75	0.75	0.80	0.75	0.80	X	X	
957‡		0.75	0.75	0.75	0.75	0.80	0.75	0.80	X	X	
958‡		0.75	0.75	0.75	0.75	0.80	0.75	0.80	X	X	
959‡		0.75	0.75	0.75	0.75	0.80	0.75	0.80	X	X	
975‡									X	X	
976‡		0.75	0.75	0.75	0.75	0.80	0.75	0.80	X	X	
977‡		0.75	0.75	0.75	0.75	0.80	0.75	0.80	X	X	
978‡		X	X	X	X	0.80	X	0.80	X	X	
979‡		X	X	X	X	X	X	X	X	X	
Obj Value		3005.1	2681.2	2357.2	2033.3	1730.2	1385.5	1074.8	720.0	428.6	144.0
Iterations		3108	3529	3141	3130	3042	2985	3375	2234	1942	

¶No final solution after 5000 iterations. † = Constitution Gardens Lake. * = Reflection Pool.
§ = Lincoln Reflection Pool. ⋆ = Noise Type Pixels. ‡ = Tidal Basin.

runs, only a few pixels were fully allocated while the rest were fractional. These answers are contrary to what one would expect given the extended border length. For these seven runs, neither the resource limit nor the iteration limit was reached. In fact, the iteration limit ranged from 2,985 (6/4) to 3,529 (2/8). More insight is required into the inner workings of GAMS to determine what caused these solutions below 24 pixels in size.

Area $\geq$ 26 & Border Length $\leq$ 48 In this set of runs, the area changes to greater than or equal to 26 and the border length has been changed to less than or equal to 48. Given these parameters, I would expect the solution to be infeasible. I expect this type of answer due to the relative short border length when compared to the size of the area. When the runs were made, this is just what happened. All the runs resulted in infeasible solutions.

Area $\geq$ 26 & Border Length $\leq$ 64 In this set of runs, the area remains set at greater than or equal to 26 and the border length has been changed to less than or equal to 64. Given these parameters, I would expect the solution to be the region of 26 pixels in the Tidal Basin area. This group of 26 pixels have a border length of 60. The results for these runs are shown in Table 32.

Contrary to what I hypothesized, the Tidal Basin pixels were not chosen exclusively. Rather, the pixels selected were chosen from the Reflection Pool and the Tidal Basin Area. Another significant result was that the pixels selected in the Tidal Basin were noninteger. Once again the inner workings of GAMS are coming into play. The GAMS software is attempting to maximize the objective fuction by taking partial pixels rather than whole pixels.

Conclusion In these runs of varying area and border length, the model appears to not be taking advantage of the relaxed constraints (See Tables 30-32). Instead of integer answers, the model is only giving partial integer answers.

Table 32. Results of Channel 1 – Channel 3 Runs; Area ≥ 26, Border Length ≤ 64

Selected Pixels (x_{1k})	GAMS Computer Runs: Channel 1 – 3 Combination Lambda Weights (Measured in Tenths)										
	0/1.0	1/9	2/8	3/7	4/6	5/5	6/4	7/3	8/2	9/1	1.0/0
528*	¶	X	X	X	X	X	X	X			X
529*		X	X	X	X	X	X	X			X
547*		X	X	X	X	X	X	X		X	X
548*		X	X	X	X	X	X	X		X	X
549*		X	X	X	X	X	X	X		X	X
550*		X	X	X	X	X	X	X		X	X
567*		X	X	X	X	X	X	X		X	X
568*		X	X	X	X	X			X	X	X
569*		X	X	X	X	X			X	X	X
570*		X	X	X	X	X			X	X	X
587*		X	X	X	X	X			X	X	X
588*			X	X		X			X	X	X
589*			X						X	X	X
590*			X						X	X	X
607*									X	X	X
608*									X	X	X
609*									X	X	X
610*									X	X	X
627*									X		X
628*									X		X
629*									X		X
630*									X		
648*											X
649*											X
686⋆											X
706⋆											X
879⋆											X
935‡		0.750	0.750	0.750	0.750	0.750	0.750	0.750	X	X	
936‡		0.750	0.750	0.750	0.750	0.750	0.750	0.750	X	X	
937‡		0.750	0.750	0.750	0.750	0.750	0.750	0.750	X	X	
938‡		0.750	0.750	0.750	0.750	0.750	0.750	0.750	X	X	
939‡		0.750	0.750	0.750	0.750	0.750	0.750	0.750	X	X	
955‡		0.750	0.750	0.750	0.750	0.750	0.750	0.750	X	X	
956‡		0.750	0.750	0.750	0.750	0.750	0.750	0.750	X	X	
957‡		0.750	0.750	0.750	0.750	0.750	0.750	0.750	X	X	
958‡		0.750	0.750	0.750	0.750	0.750	0.750	0.750	X	X	
959‡		0.750	0.750	0.750	0.750	0.750	0.750	0.750	X	X	
975‡		0.750	0.750	0.750	0.750	0.750	0.750	0.750	X	X	
976‡		0.750	0.750	0.750	0.750	0.750	0.750	0.750	X	X	
977‡		0.750	0.750	0.750	0.750	0.750	0.750	0.750	X	X	
978‡		0.750	0.750	0.750	0.750	0.750	0.750	0.750	X	X	
979‡		0.750	0.750	0.750	0.750	0.750	0.750	0.750	X	X	
Obj Value		3122.6	2787.0	2451.3	2115.7	1780.0	1444.4	1108.7	693.2	428.6	128.0
Iterations		2890	2851	2704	3177	3016	2786	2943	4410	1924	

¶No final solution after 5000 iterations. † = Constitution Gardens Lake. * = Reflection Pool.
§ = Lincoln Reflection Pool. ⋆ = Noise Type Pixels. ‡ = Tidal Basin.

It is significant to note that all the computer runs that resulted in answers did not exceed the iteration limit, resource limit, branch and bound node limit, or the work space limit. With no error messages in these runs, I was at a loss for how to explain these results. I next called the GAMS Development Corporation for a technical explanation. According to Erwin Kalvelagen, there is no problem with the model as it is currently written. In fact, he ran the 1/9 program of Table 30 and the output was integer. The integer solution was composed of the following pixels: 528-529, 547-550, 567-570, 587-590, 607-610, 627-630, and 648-649. The iteration count was 2,813 and the corresponding objective value was 2,253.2 (17). The 24 pixels selected are the same pixels selected when I attempted to find the Reflection Pool with area equal to 24 and border length equal to 44. The model has correctly maximized the area while minimizing the border length.

I had not arrived at this integer solution only because he used a different solver, GAMS/OSL, rather than the one I was using, GAMS/ZOOM. The OSL solver is only available on the 2.25 version of GAMS. The GAMS package installed at the Air Force Institute of Technology where I conducted my research only has Version 2.2 installed. I also asked if I could solve these problems with the GAMS/ZOOM solver and why I had noninteger solutions. He said that the GAMS/ZOOM solver was capable of solving these problems, but I would need to raise the iteration count. The answer to the second part of this question was more interesting. When GAMS/ZOOM is used to solve a mixed integer program, it solves the problem in two steps. In the first step, the problem is solved as an LP with the integer constraint relaxed. In the second step, the solver uses this information to solve for the integer solution. After the model has solved the first step, it looks to see if there is sufficient resources (iteration limit, resource limit, branch and bound node limit, and the work space limit) to complete the second step. The model uses the first step as an estimate for what resources are required in the second step. If the resources are not sufficient, the program will stop at this noninteger solution (17). The output does not tell the programmer that this

condition has occurred, but rather says that the program has successfully completed with a noninteger solution. This is a nuance of GAMS/ZOOM that I would not have known unless I contacted the GAMS Development Corporation. Erwin stated that this summary of successful program completion is confusing to the user and thus they have changed it in Version 2.25 of GAMS. Given this new information, I again ran the four sets of programs shown in Table 28. However, this time I changed the iteration limit to 25,000.

4.3.3.9 Increased Iteration Limit Programs The four sets of programs shown in Table 28 were rerun using an iteration limit of 25,000. The programs should not need this much, but I have increased it to this level for insurance. Erwin said the necessary iteration limit needed by GAMS/ZOOM should be about double the iteration count in the noninteger solutions. If this is true, most of the programs should successfully complete with integer solutions between 6,000 – 7,000 iterations.

Area $\geq$ 24 & Border Length $\leq$ 48 This set of runs begins the set of four programs where the area and border length are both bounded by two sizes. In the first set of runs, the area has be set greater than or equal to 24 and the border length has been set less than or equal to 48. Given then parameters, I would expect the model to select the Reflection Pool in total. Recall that the Reflection Pool had an area of 24 and a border length of 44. The results for these runs are shown in Table 33.

Table 33. Results of Channel 1 – Channel 3 Runs; Area $\geq$ 24, Border Length $\leq$ 48

Selected Pixels (x_{ik})	GAMS Computer Runs: Channel 1 - 3 Combination Lambda Weights (Measured in Tenths)										
	0/1.0	1/9	2/8	3/7	4/6	5/5	6/4	7/3	8/2	9/1	1.0/0
	¶	¶	¶	¶	¶	¶	*	*	¶	¶	*
Iterations	11852	22843	23259	23008	22771	23028			22912	23042	
¶No final solution. * = No solution after running 4+ hours on the VAX.											

The results from these runs were not what I expected. The iteration counts are nearly ten times those of the noninteger runs (See Table 30.) and the solutions were noninteger, rather than integer. At first, one would suspect the program that I wrote, but this is not the case. I sent my program to Erwin Kalvelagen via electronic mail and he ran the 1/9 program with the GAMS/OSL solver. The results from this run were what I expected — the Reflection Pool in total was selected by the model. This solution was the one that I expected the model to find. The combined objective value was 2,253.2 and this solution occurred after only 2,813 iterations (17). Since the program ran correctly on the GAMS/OSL solver, I do not suspect the program itself. Rather, I suspect this is another nuance of the GAMS/ZOOM solver that is not published. When I called Erwin Kalvelagen for assistance, he offered another possible explanation. He said it is possible that the relaxed LP (noninteger) solution is way off from the integer solution. If this is true, the model will have to run more than double the number of iterations required for a noninteger solution (17). A portion of the output from the GAMS/OSL solver and the program itself is shown in Appendix M.

Area $\geq$ 24 & Border Length $\leq$ 64 In this set of runs, the area remains set at greater than or equal to 24 and the border length has been changed from less than or equal to 48 to less than or equal to 64. Given these parameters, I would expect the solution to be more spread out due to the increased border length. The results for these runs are shown in Table 34.

In these set of runs, the pixels selected were from the Reflection Pool and the Tidal Basin. The only exception to this statement is run number eleven (1.0/0). In this run the selected pixels also came from the Constitution Gardens Lake and the Noise-Type pixels. These answers are in line with what I hypothesized before the runs. The reason why the pixels selected are not contiguous is that the border length has been increased. With a longer border length, the model is given more flexibility in selecting those pixels which will maximize the objective function. Notice in all

Table 34. Results of Channel 1 – Channel 3 Runs; Area $\geq$ 24, Border Length $\leq$ 64

Selected Pixels (x_{ik})	GAMS Computer Runs: Channel 1 – 3 Combination Lambda Weights (Measured in Tenths)										
	0/1.0	1/9	2/8	3/7	4/6	5/5	6/4	7/3	8/2	9/1	1.0/0
264†	¶										X
528*		X	X	X	X	X	X	X			
529*		X	X	X	X	X	X	X			
547*		X	X	X	X	X	X	X		X	X
548*		X	X	X	X	X	X	X		X	X
549*		X	X	X	X	X	X	X		X	X
550*		X	X	X	X	X	X	X		X	X
567*		X	X	X	X	X	X	X	X	X	X
568*		X	X	X	X	X	X	X	X	X	X
569*		X	X	X	X	X	X	X	X	X	X
570*		X	X	X	X	X	X	X	X	X	X
587*		X	X	X	X	X	X	X	X	X	X
588*		X	X	X	X	X	X	X	X	X	X
589*		X	X	X	X	X	X	X	X	X	X
590*		X	X	X	X	X	X	X	X	X	X
607*		X	X	X	X	X	X	X	X	X	X
608*		X	X	X	X	X	X	X	X	X	X
609*		X	X	X	X	X	X	X	X	X	X
610*		X	X	X	X	X	X	X	X	X	X
627*		X	X	X	X	X	X	X	X		X
628*		X	X	X	X	X	X	X	X		X
629*		X	X	X	X	X	X	X	X		X
630*		X	X	X	X	X	X	X	X		X
686⋆											X
706⋆											X
879⋆											X
935‡		X	X	X	X		X		X	X	
936‡		X	X	X	X	X	X		X	X	
937‡		X	X	X	X	X	X	X	X	X	
938‡						X		X	X	X	
939‡								X	X	X	
955‡		X	X	X	X		X		X	X	
956‡		X	X	X	X	X	X		X	X	
957‡		X	X	X	X	X	X	X	X	X	
958‡						X		X	X	X	
959‡								X	X	X	
975‡		X	X	X	X		X		X	X	
976‡		X	X	X	X	X	X		X	X	
977‡		X	X	X	X	X	X	X	X	X	
978‡						X		X	X	X	
979‡									X	X	
Obj Value		2878.8	2568.6	2258.4	1948.2	1686.0	1327.8	1046.1	720.0	428.6	144.0
Iterations	12172	4373	4776	4421	4371	4304	4386	4748	2236	1994	1090
Area		31	31	31	31	31	31	30	31	31	24
Border		64	64	64	64	64	64	64	64	64	64

¶No final solution after 12,172 iterations. †= Constitution Gardens Lake. * = Reflection Pool.
§= Lincoln Reflection Pool. ⋆ = Noise Type Pixels. ‡= Tidal Basin.

of the solutions, the model took advantage of this increased border length and used the full limit of 64. However, the number of pixels selected varied from 24 to 31.

Area $\geq$ 26 & Border Length $\leq$ 48 In this set of runs, the area changes to greater than or equal to 26 and the border length has been changed to less than or equal to 48. Given these parameters, I would expect the solution to be infeasible. I would expect this due to the relative short border length when compared to the size of the area.

Rather than burning computer time to prove this hypothesis, it can be shown that this solution is not possible with the given data set. Recall in the previous section that this set of problems resulted in infeasible solutions. Adding more iterations to these problems will not result in feasible integer solutions. If the equations for calculating the border (Reference Chapter 3) are used, a minimum border length can be calculated. Since a smaller area will usually result in a smaller border length, I will assume an area of 26 pixels. With this area, the minimum border is 22. Recalling that the border length is defined as twice the border, a border length of 44 results. Several geometric representations of an area of 26 pixels with a border length of 44 are shown in Figure 35.

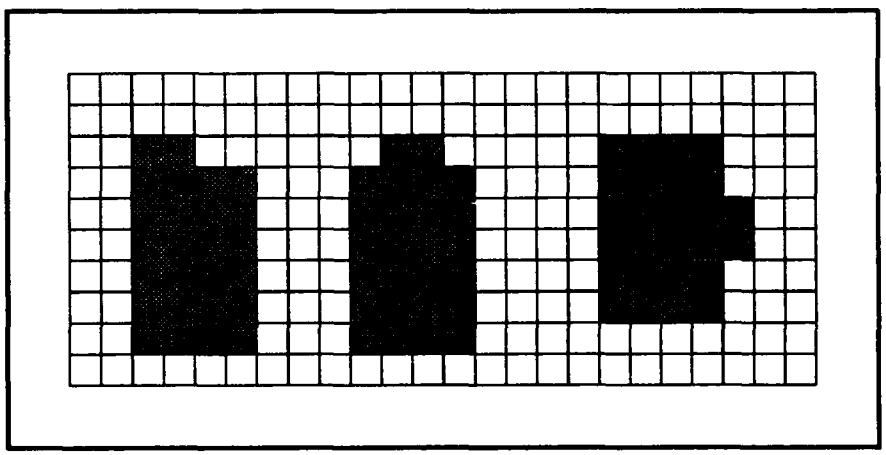

Figure 35. Geometric Shapes for Area = 26 and Border Length = 44

In Figure 35 all the subregions are composed of a main group of 24 pixels (6 $\times$ 4 matrix) with two adjacent pixels on the border of this main group. As long as these two pixels stay together, they can be moved anywhere around the main group border and the border length of 44 will be maintained. If these two pixels are split, the border length increases to 48. They can be moved anywhere on the border of the main group and the length will remain at 48 as long as the two pixels remain separate. The model will attempt to find a geometric pattern of this sort which maximizes the objective function. If the water subregions in Appendix G are reviewed, no geometric patterns of this sort are present in the Channel 1 image. Recall that Channel 1 acts as a filter for the Channel 3 data. The Reflection Pool is the one water subregion that comes closest to this geometric pattern, but the area and border length are not equal to 26 and 48, respectively[27]. Given that the necessary geometric pattern is not present in the image, the model will not be able to find a feasible solution.

Area $\geq$ 26 & Border Length $\leq$ 64 In this set of runs, the area remains set at greater than or equal to 26 and the border length has been changed to less than or equal to 64. Given these parameters, I would expect the solution to be the region of 26 pixels in the Tidal Basin area. This group of 26 pixels have a border length of 60. The results for these runs are shown in Table 35.

Contrary to what I hypothesized about the solution, the pixels selected were not exclusively from the Tidal Basin area. Rather, the model maximized the objective function by selecting pixels from the Reflection Pool, Noise-Type Pixels, and the Tidal Basin. It is interesting to note that the size of the area selected varied from 26 to 32 pixels, but the border length of 64 was maintained. Most of the solutions were the same (same pixels selected) however, for runs 3/7, 5/5, and 1.0/0 the selected pixels were different. In the first run (0/1.0), the program quit after

[27]The area and border length for the Reflection Pool is 24 and 44, respectively.

Table 35. Results of Channel 1 – Channel 3 Runs; Area $\geq$ 26, Border Length $\leq$ 64

Selected Pixels (x_{ik})	GAMS Computer Runs: Channel 1 - 3 Combination Lambda Weights (Measured in Tenths)										
	0/1.0	1/9	2/8	3/7	4/6	5/5	6/4	7/3	8/2	9/1	1.0/0
528*	¶			X		X					X
529*				X		X					X
547*		X	X	X	X	X	X	X	X	X	X
548*		X	X	X	X	X	X	X	X	X	X
549*		X	X	X	X	X	X	X	X	X	X
550*		X	X	X	X	X	X	X	X	X	X
567*		X	X	X	X	X	X	X	X	X	X
568*		X	X	X	X	X	X	X	X	X	X
569*		X	X	X	X	X	X	X	X	X	X
570*		X	X	X	X	X	X	X	X	X	X
587*		X	X	X	X	X	X	X	X	X	X
588*		X	X	X	X	X	X	X	X	X	X
589*		X	X	X	X	X	X	X	X	X	X
590*		X	X	X	X	X	X	X	X	X	X
607*		X	X	X	X	X	X	X	X	X	X
608*		X	X	X	X	X	X	X	X	X	X
609*		X	X	X	X	X	X	X	X	X	X
610*		X	X	X	X	X	X	X	X	X	X
627*		X	X	X	X	X	X	X	X	X	X
628*		X	X	X	X	X	X	X	X	X	X
629*		X	X	X	X	X	X	X	X	X	X
630*		X	X	X	X	X	X	X	X	X	
648*											X
649*											X
686⋆											X
706⋆											X
879⋆											X
936‡		X	X		X		X	X	X	X	
937‡		X	X	X	X	X	X	X	X	X	
938‡		X	X	X	X	X	X	X	X	X	
939‡		X	X	X	X	X	X	X	X	X	
956‡		X	X		X		X	X	X	X	
957‡		X	X	X	X	X	X	X	X	X	
958‡		X	X	X	X	X	X	X	X	X	
959‡		X	X	X	X	X	X	X	X	X	
976‡		X	X		X		X	X	X	X	
977‡		X	X	X	X	X	X	X	X	X	
978‡		X	X	X	X	X	X	X	X	X	
979‡		X	X	X	X	X	X	X	X	X	
Obj Value		3060.9	2731.8	2361.8	2073.6	1713.5	1415.4	1086.3	757.2	428.6	128.0
Iterations		7053	7686	5157	7769	5097	7139	6467	5964	1924	2997
Area		32	32	31	32	31	32	32	32	32	26
Border		64	64	64	64	64	64	64	64	64	64

¶No final solution after 12,172 iterations. †= Constitution Gardens Lake. * = Reflection Pool.
§= Lincoln Reflection Pool. ⋆ = Noise Type Pixels. ‡= Tidal Basin.

12,172 iterations. This can be explained by the information supplied by the GAMS Corporation. This program quit after solving the relaxed (noninteger) LP. The program then looks ahead to see what resources are available for solving the restricted (integer) LP. In this case, the program decided that it would need more than 12,828 iterations so it stopped at the relaxed solution.

Conclusion All of the results from the four sets of runs were fairly predictable once the iteration limit had been raised sufficiently. The result of raising the iteration limit is that all of the solutions were integer. The iteration count for these integer solutions were approximately double that of the noninteger solutions. The only exception to these statements is the computer runs with an area greater than or equal to 24 and a border length less than or equal to 48. In these runs, the iteration count was usually over 20,000. These unusually high iteration counts also caused the computer time to increase significantly. These 20,000+ iteration count runs took over four hours on the VAX while the others were usually under 45 minutes. Some of these lower iteration count runs ran in a few minutes. When one of these high iteration count programs were run with a GAMS/OSL solver, an integer solution resulted in under 2,500 iterations. Given that the program is correct, there are two possible explanations for the noninteger solutions. The first is that this is another nuance of GAMS/ZOOM that is unpublished. A second more likely explanation is that the initial relaxed (noninteger) solution is way off from the integer solution. If this is true, this could result in the high iteration count seen.

4.3.4 Vegetation Index Programs Another method of processing multispectral imagery is to first combine the channels into a vegetation index before searching for a particular spectral and spatial pattern. Recall from Chapter 2 that two vegetation indices were defined as follows:

$$\text{VI} = \text{Near-IR} - \text{Red} \tag{68}$$

$$\text{NVI} = \frac{\text{Near-IR} - \text{Red}}{\text{Near-IR} + \text{Red}} \tag{69}$$

In these equations, the VI stands for Vegetation Index and NVI stands for Normalized Vegetation Index. The variables in the equations represent the gray value of a pixel in the red and near-IR imaging bands. With SPOT imagery, the red band corresponds to Channel 2 and the near-IR band corresponds to Channel 3. By implementing these equations, two bands of imagery are combined before processing occurs. In general both indices result in high values for vegetation areas due to their relatively high near-IR reflectance and low red reflectance. In contrast, clouds, water, and snow have negative values due to their larger visible reflectance than near-IR reflectance. Rock and bare soil have similar reflectance values in both bands so their index is close to zero (21:598).

With this knowledge about the indices, the image analyst knows to look for negative pixel values if he is looking for water regions. In Appendix L, I have combined SPOT Channels 2 and 3 using the VI and NVI equations.[28] This merging of the data was accomplished by running a FORTRAN program I created for this purpose. The output of the program will be used in an unweighted objective function since there is *only one channel* of data. Using the water regions identified by the three-channel filter (Appendix I), the combined data can be examined for negative values in these areas. A review of this data failed to show any negative pixel values in the four major water subregions. Two individual *noise-type* water pixels did have a negative values. This result was true for both sets of data and is shown graphically in Appendix L.

Another method to combine multispectral imagery is a ratio of the near-IR and red bands. This ratio is applicable to SPOT, Landsat TM and MSS, and NOAA AVHRR imagery (12:87-88). This ratio equation is given below.

[28]The FORTRAN program created to merge the two channels is also included in Appendix L.

$$GV_{Ratio} = \frac{GV_{Near-IR}}{GV_{Red}} \tag{70}$$

In this equation, GV_{Ratio} is the ratioed gray value, $GV_{Near-IR}$ is the pixel gray value in the near-IR band, and GV_{Red} is the pixel gray value in the red band. Another FORTRAN program was used to perform the ratio of the two channels. The results of this program and the program code are included in Appendix L. In general, water is identified by ratios less than one (23:189-191). The same examination of the data, as performed on the vegetation indices, was performed on this ratioed data. Again only two individual *noise-type* water pixels were found in the water subregions identified in Appendix I.

Since none of the three methods for combining two channels of imagery worked, it is necessary to analyze why this occurred. There are several factors that contribute to this problem. The first factor to consider is the VI and NVI indices. Both of these indices were created for analysis of NOAA AVHRR imagery. While the NOAA AVHRR and the SPOT satellite both have a red and near-IR imaging band, they differ in size. For the AVHRR, the red band is measured from 0.58–0.68 μm, while the near-IR is measured from 0.725 - 1.10 μm (34:36). The SPOT satellite's red band is measured from 0.61–0.68 μm, while the near-IR is measured from 0.79–0.89 μm (21:582). Given these differences, several statements can be made. Since the red bands are approximately the same, the gray values for a red image will be similar, but slightly different. Secondly, since the near-IR bands are different, the gray values should be noticeably different. If the gray values are different, the ratios produced by the VI and NVI will be different for the NOAA AVHRR and SPOT systems. Without actual imagery from both systems, it is difficult to tell how great this difference is. However, it seems likely that this difference would be a significant factor in the vegetation indices not working for the SPOT imagery.

A second factor why neither the vegetation indices nor the band ratio worked on the SPOT data is the preprocessing that has occurred on the imagery. In Sec-

tion 4.2.1 a summary of this preprocessing was given. Final processing of the SPOT imagery by the EROS Data Center included a multipoint linear stretch. To understand how this stretch can affect the gray values in the image, it is necessary to look at the spectral reflectance curves for the four *themes* (grassy vegetation, water, roads, and pathways) in the image and also explain the concept of a multipoint linear stretch. In Figure 36 the spectral reflectance curves for the four themes are shown.

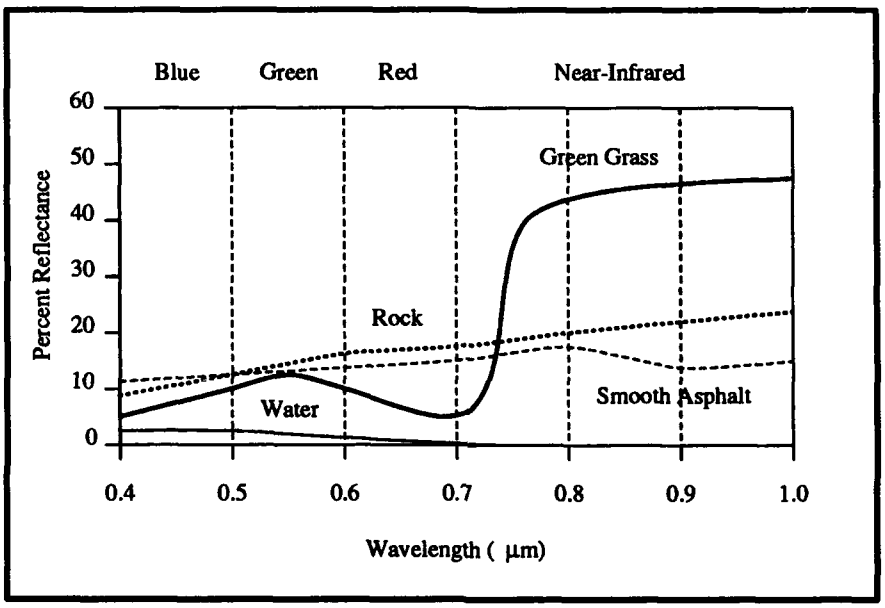

Figure 36. Land Cover Reflectance Curves for SPOT Subimage (9:3-101), (15:159), (30:85)

A review of Figure 36 shows that rock and asphalt have similar spectral signatures both in the SPOT red and near-IR bands. In the red band, there is little distinction between all the four themes, but in the near-IR band the four themes are separated to a greater extent. If the gray values in the image followed these curves, the near-IR band would distinguish the four themes when using the vegetation indices and band ratioing. In fact, Channel 3 should have done the best job (not Channel 1) in the single band analysis. Since the image data did not allow the water subregions to be resolved, the gray values must have been altered from these

curves. The multipoint linear stretch may be the key to how the reflectance curves were altered.

In general, sensor data rarely extends over the entire range of gray levels of recording devices. For this reason, contrast stretching is used to "expand the narrow range of brightness values typically present in an input image over a wider range of gray values" (21:627). The result of this stretching is an output image in which the contrast between features of interest have been accentuated. To explain the concept of a multipoint linear stretch, it is necessary to consider a hypothetical example, beginning with a simple linear stretch. Consider an imaging system that is capable of recording at 256 gray levels. Figure 37 shows the histogram of a scene recorded by this system in one channel.

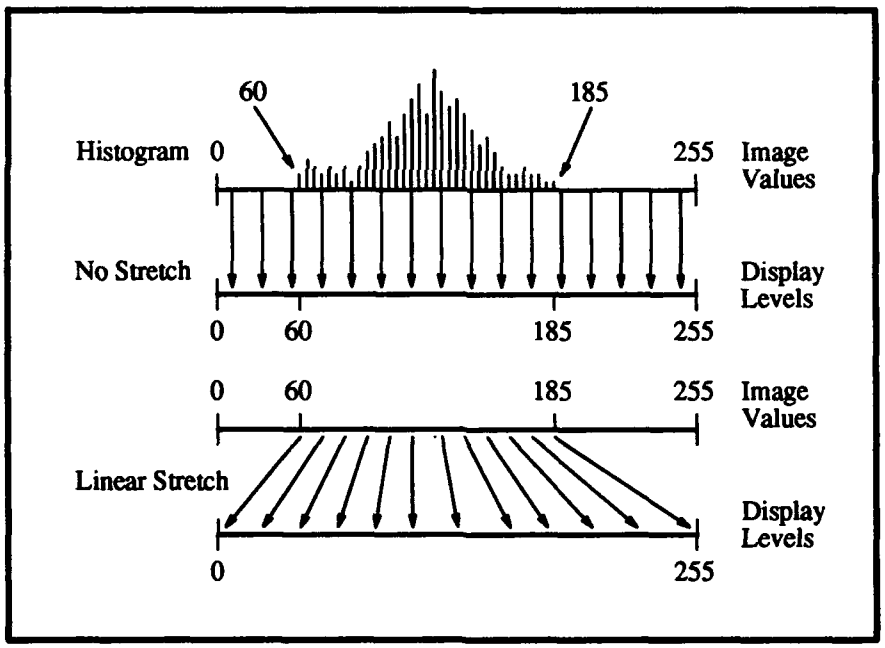

Figure 37. Principle of Linear Stretch Enhancement (21:631)

For this particular scene, the range of gray values is from 60–185. If no stretch is applied to this data, this will also be the range of gray values seen on the output

device[29]. If a linear stretch is applied to the image data, the gray value range can be uniformly increased to cover all 256 gray levels. From this figure, one can see that each of the original gray values have been changed through this stretch process. If this were the process applied to the SPOT imagery, it would have been fairly easily to compute the original gray values. The equation for the mapping of the original gray values into the linearly stretched gray values is given below (21:630-631):

$$DN' = \left(\frac{DN - MIN}{MAX - MIN}\right) 255 \tag{71}$$

where DN' = Digital number assigned to output pixel,

DN = Digital number assigned to original input pixel,

MIN = Minimum value of input image to be assigned value of 0,

MAX = Maximum value of input image to be assigned value of 256.

In the example given, the 60 level pixels would be mapped to 0 and the 185 level pixels would be mapped to 256. While this method is useful for visually displaying the image, application of vegetation indices may not be possible. This premise is based on the fact that the gray values have been changed.

The problem of a linear stretch is further compounded by the multipoint linear stretch. The principle of this enhancement process is shown in Figure 38. In this figure, the original histogram has been divided into three intervals: 60–100, 100-150, and 150–185. Each of these intervals are then stretched varying amounts. With this method, the image analyst is able to apply the linear stretch in varying amounts to enhance different portions of the histogram.

Once again this processing is very useful to the analyst who is visually examining the image. The analyst can apply a greater stretch where needed and less where

[29]This assumes the output device is capable of displaying 256 levels of gray.

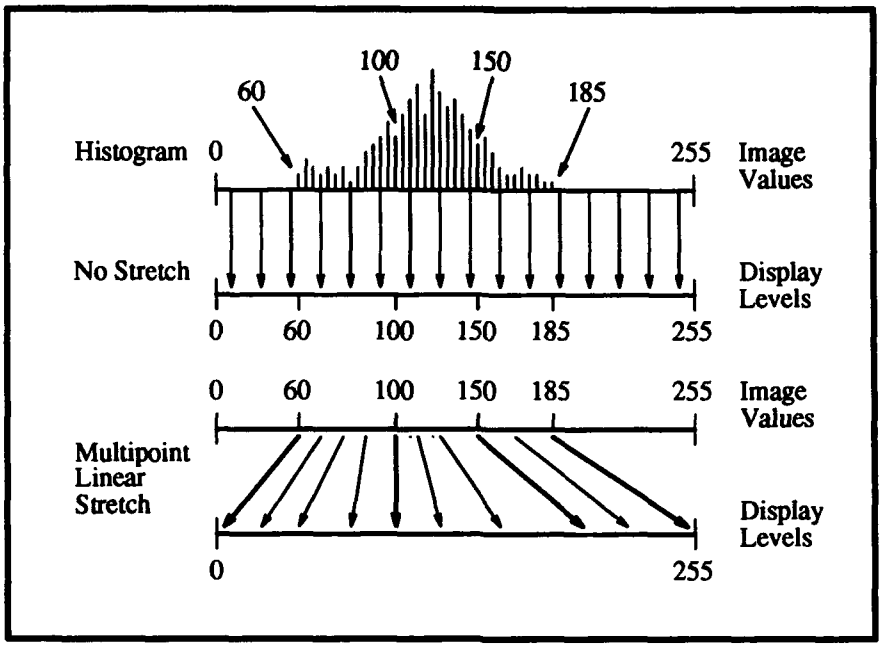

Figure 38. Principle of Multipoint Linear Stretch Enhancement

the contrast is already sufficient. This variance in the amount of stretching is shown in Figure 38. The third interval (150–185) has been stretched the least while the second interval (100–150) has been stretch the most. If the location of the intervals and the stretch algorithms were known, the original gray values of the SPOT subimage could again be calculated and if the original gray values were known, then the vegetation indices and band ratio should have been applicable. According to Glen Kelly of the EROS Data Center, the original paperwork on the processing of this particular SPOT image is no longer available. Without this information, one cannot state conclusively how significant this is as a reason why the vegetation indices and band ratio did not work.

When applying both vegetation indices, water is characteristically identified by negative values. With the band ratio, water is identified by values less than one. With the multipoint linear stretch, a hypothesis can be suggested as to why these values did not result. For the correct values to appear, water is required to have gray

values in the near-IR band that are less than the gray values in the red band. This is the natural character of water as shown previously in Figure 36. If a multipoint stretch was applied to the near-IR band, in such a manner that the gray values for water were stretched above the red band gray values, then the existing problem is possible. In contrast to this processed SPOT image, Landsat imagery is not allowed to be processed (by the provider) with this type of enhancement. This is probably due to the commercial interests, but it also avoids this sort of confusion (3).

One final factor that could contribute to this problem of distorted gray values is the imaging conditions. These conditions include: the illumination angle/time of day, moisture content of gravel and soil, and weather. The illumination angle/time of day has a great effect on how water is recorded. If the sun's reflection is recorded in the water areas, a specular reflection will occur. If the sun is not visible or the imaging angle is such that the sun's reflection is not recorded then a more diffuse reflection will occur. How the water subregions are recorded will greatly depend on whether a specular or diffuse reflection is present. Moisture content can also have an effect on gravel and soil. If the sun is at a low illumination angle and the moisture content is high (i.e. dew on the ground), then higher than normal gray values can result. Finally the weather also has an effect on the image. In addition to changing the moisture content, clouds can filter the spectral response from the scene being imaged. While the clouds may not cover the entire image, they can affect those areas that are covered.

Since none of the three algebraic combinations resulted in the correct values for water, I did not attempt to process the imagery further. If the data had been raw SPOT data I would expect the correct values to appear in the water subregions identified in Appendix I. Another possibility of finding the water subregions would have been to create new algebraic combinations. Lack of time did not permit me to attempt this possibility (and the uncertain nature of the pre-processing would leave the results with limited utility).

4.4 *Conclusion*

In reviewing the results of all the computer runs made in this chapter, several observations are evident. I will first comment on the multiobjective programs and then comment on the vegetation indices. In the multiobjective computer runs, Channel 1 was the critical channel in finding an optimal solution in a reasonable amount of time.[30] This statement is validated by the successful runs of Channel 1 alone, Channels 1 and 2 together, and Channels 1 and 3 together. If Channel 1 is removed from the image processing, spectral and spatial pattern recognition will not occur in a timely manner. The results of these runs lead to an important conclusion for the multiobjective function runs. The more spectrally distinct the target of interest is, the quicker an optimal solution can be found. A thorough spectral analysis of the image should tell the analyst if it is even worth running the program. Spectral analysis on Channels 2 and 3 revealed that water was not spectrally distinct in these channels. Since water was not distinct in these channels, the following runs were not successful after 5,000 iterations: Channel 2 alone, Channel 3 alone, and Channels 2 and 3 together. The key to processing a channel in which the target is not spectrally distinct is to combine this channel with one that is.

When the area and border constraints were set to 24 and 44 respectively, the reflection pool was found no matter what the lambda weights were.[31] The constraints are sufficiently tight so that this is the only possible solution. Since the same 24 pixels were chosen every time, the concept of an efficient frontier did not come into play. In an effort to make the model choose different pixels, the border length was increased. Since the B&W model is a multicriteria optimization problem in itself, changing the border length is similar to changing the lambda weights. As the border length constraint was increased, the number of non-contiguous pixels also increased. Recall

[30] I recheck ed the channel assignments, since the natural spectra suggest Channel 3 would best distinguish water.

[31] This statement does not apply to the following runs: Channel 2 alone, Channel 3 alone, and Channels 2 and 3 together.

that contiguity for a given pixel-size area is only enforced with border lengths near the minimum. With the change in border length, different pixels were selected by the model. Using the data collected from the border length = 192 runs, an efficient frontier was plotted for all lambda weights, except 1.0/0. Another problem that occurred was noninteger solutions. Up to this point, this had not occurred. It was suspected that the model was being forced to locate geometric patterns that were not present in the imagery. To see if this suspicion was true, another set of runs were made where the border length was set to an inequality rather than an equality. In these runs, the majority of the solutions were integer, but not all of them. In fact, in the border length ≤ 60 set of runs, the answers were primarily noninteger. By allowing a range of acceptable border lengths, the model has more flexibility in what pixels will be selected. This flexibility may have caused the increase in integer answers. A direct consequence of this flexibility is that the number of iterations required for a solution tended to increase to around 3,000.

The problem of noninteger solutions was mostly corrected by raising the iteration count. The reason why this is necessary is that the GAMS/ZOOM solver solves the problems in two steps. In the first step, the problem is solved as a relaxed (noninteger) LP. At this point the model stops and looks to see if there are sufficient resources for the problem to be solved again. In this second step, the problem is solved as an integer LP. The second step requires approximately the same amount of resources as the noninteger LP. However, this did not always hold true. In the runs with a area greater than or equal to 24 and a border length less than or equal to 48 the model did not give an integer solution after 20,000 plus iterations. It is suspected that the initial noninteger solution is far off from the correct integer solution. For this reason, the model will require significantly more resources than the noninteger solution. With the completion of these runs, the next method of processing imagery is considered.

The vegetation index and band ratio methods were simple in nature to understand and calculate, but the lack of proper data prevented me from attempting spectral and spatial recognition with these image combinations. If raw data were available, the probability of one of these combinations working should increase. There were three possible answers for why the data were not amenable to processing with these methods. The first possible reason is that the vegetation indices were designed for the NOAA AVHRR system and not for the SPOT system. In my research I did not find any vegetation indices developed uniquely for SPOT imagery. Since this was the case, I had hoped to adapt these two indices to SPOT imagery. Another contributing factor why the data was not useful is the preprocessing that occurred on the imagery. One of the preprocessing techniques used was the multipoint linear stretch. This technique has the potential to change the original gray values and thus, invalidated the vegetation indices and band ratio. A final factor that could cause the gray values to be out of the normal range is the imaging conditions. This is a catch all factor that considers the following: illumination angle/time of day, moisture content of gravel and soil, and weather. All of these can affect the gray values that are recorded for a scene. Once again, if the gray values change out of their characteristic spectral ranges then the combination methods explained may not be valid.

Without any GAMS runs for these combinations, I was not able to compare the two approaches for recognizing spectral and spatial patterns. This comparison could be accomplished if raw data were available or if the algorithm for the multipoint linear stretch were know. The latter would allow the given gray values to be remapped back into the original gray values.

V. Conclusions and Recommendations

The research accomplished in this thesis met the overall goal of developing a general purpose model to recognize spectral and spatial patterns in digital imagery. However, this model remains a research tool and is not an operational program that can be used for imagery analysis on full-size images. In the following sections, the successes as well as the work that still remains is called out.

5.1 Pattern Recognition Model

The B&W model was successfully modified into a tool that can be used to solve relatively small spectral and spatial pattern recognition problems. The model was successfully run with a combined three channel data set of 1,000 pixels.[1] If one is going to use this model, it is assumed that the analyst has a priori information about the target size and border length. In my analysis, it was shown that this knowledge alone may not be enough to find the target on the first computer run. If the target is not aligned with the pixel grid, the apparent target size and border length may increase. For this reason, a range of areas and border lengths is the most efficient manner to find a given target.

All the combined objective function solutions found by this model were composed of water pixels from the four regions and the noise-type pixels[2]. These noise-type pixels were candidates for selection since they had the correct spectral pattern identified for water. They were chosen at times when there was sufficient border length available and when they maximized the objective function. To verify that the solutions were indeed water, the solutions were checked against a map of the same

[1]It should be noted that each of the pixels have three gray values to correspond to the three channels.

[2]Note: When only Channel 2 was processed, some pixels were selected from nonwater areas (See Table 27). There are two reasons why this occurred. The first is that Channel 1 was not present to serve as a filter for the Channel 2 data. The second reason is that these nonwater pixels had a gray value in the correct spectral range for water.

area. The only water pixels that could not be found on the map were the noise-type pixels. This high degree of accuracy was possible because on the spectral bounds placed on the pixel selection.

The most important improvement to the B&W model was the addition of pixel spectral bound constraints. Rather than just looking for the highest or lowest cost pixels, the analyst can now set the pixel bounds for a specific target type. The more channels available, the more spectral bound constraints that can be specified. Each additional constraint serves as an additional filter for the imagery data. When these spectral constraints are combined, a type of parallelepiped classifier results. The model does not set the the spectral bounds for the targets to be located in the image. Rather, the analyst must perform a spectral analysis of the given imagery to determine the spectral range for the targets. If the spectral bounds are known a priori, then this analysis is not needed. In general, additional channels are only profitable if they help to reduce the feasible data set of pixels. If a given channel is being considered for incorporation into the model, the above test can be applied to see if it is beneficial to add the channel. When I combined Channels 1 and 3 together, the gain was marginal since only a few pixels were eliminated (Reference Section 4.2.4.3). Since the number of feasible pixels were only slightly reduced, the classification accuracy of using two channels was not significantly improved. For the data set used in this thesis, classification accuracy is nearly the same with only Channel 1 as compared to the Channel 1 – 3 combination. When the Channel 1 – 3 combination was used, only three water-noise-type pixels were eliminated from selection (Reference Tables 34 and 35). While Channel 1 alone could have been used to find water, this will not always hold true for other target types. In fact, this is one of the purposes of multispectral imagery. For a given imaging band, two target types may appear the same spectrally. In another band, one of these target types may appear spectrally similar to a third target type. Thus, if only one imaging band is used for analysis, the two spectrally similar targets will not be able to be

distinguished. By using both channels with the two spectral bounds the target can be distinguished from the others. In general, classification accuracy will improve with additional channels.

Additional channels alone will not improve the pixel selection process. The model's efficiency increases in direct proportion to the spectral distinctiveness of the target. This conclusion was seen when Channels 2 and 3 were processed individually and together. Since water was not distinct in these channels, the model aborted after 5,000 iterations. The spectral analysis of these channels showed that the feasible water-spectral-range pixels nearly covered the entire image. Given that there was hardly any reduction in the amount of data to be processed, it is not surprising that the program aborted. It should be noted though that these channels were successfully processed when combined with Channel 1. Channel 1 serves as a filter to eliminate almost all of the pixels that were nonwater.

To further test the capabilities of this model, the border length was allowed to vary while the area remained constant at 24. These runs were significant since these were the first ones where noninteger solutions were encountered. Several sets of runs were made, first with a set border length and then with a border length range. In these runs, the lambda weights for each channel were also allowed to vary. At first I suspected that the noninteger solutions were the result of the model being forced to find geometric patterns that did not exist. However, the computer runs with a border length range did not validate this premise. For some reason, the border length is overriding the assignment of the x_{ik}'s (individual pixels) as a binary variable. When the area and border constraints of 24 and 44 respectively were used to find the Reflection Pool, this problem was never encountered.

The final test for the model was to run four sets of programs where the area and border length were allowed to vary. In these runs, the selected pixels were not always contiguous. This phenomena occurred at times because the border length was long enough to allow more than one regions to be selected and because the

pixels which maximized the objective function were not always contiguous. One of these sets of runs did not converge to a integer solution as I had predicted. This is probably the result of there not being sufficient resources for the program to run to completion. Additional information on this phenomena is given in Section 5.2.

While the model currently works, its efficiency could be improved by reformulating the model into a network with side constraints problem. Optimal integer answers will be the rule (this is desired) and the computer processing time necessary should decrease. I attempted to reformulate the model as such, but was unable to proceed past Captain Reed's work. Integer solutions resulted for small problems, but as the problem size increased, this did not hold true. Reed has completed an entire thesis on this subject which offers a good starting point for this improvement. His thesis is entitled, *Binary Programming Models of Spatial Pattern Recognition: Applications in Remote Sensing Image Analysis* (26). In Appendix N, I have included some of my work in formulating the model as a network with side constraints problem.

Another improvement that remains to be implemented is the objective function which is biased to pick the most representative pixels (based on gray value) of a given target. Currently the model is biased to select pixels at the extremes of a given spectral range. This improved objective function was developed in Chapter 3, but was not implemented due to the incompatibility of the GAMS software. The improved objective function needs to be reformulated or a new software package selected that is compatible with this objective function.

5.2 Combining Single Channel Images

A successful method to combine single channel images in digital image processing is the multicriteria optimization, weighted sums approach. This method was used to successfully process Channel 1 individually and with Channel 2 or Channel 3. This success was due to the spectral distinctiveness of water in Channel 1. This

method is another improvement that was added to the B&W model. As originally designed, the B&W mode only had the capability to process one band of imagery at a time. Although the maximum number of bands processed simultaneously was only three, this method can be expanded to easily accommodate more channels. This maximum of three bands was set by the amount of data available (three SPOT multispectral channels) rather than a limitation of the improvement. This method also allows the generation of an efficient frontier to find the nondominated solutions.

In running the model with this feature, the user needs to ensure that the iteration limit is sufficient for an integer solution to result. If the model stops with a noninteger solution before the iteration limit or any other resource limit is exceeded, then the iteration limit needs to be increased. This nuance of GAMS/ZOOM is the direct result of how it solves an integer LP. In the two-step process, the problem is first solved as a relaxed (noninteger) LP. The model then stops at this point to see if there are sufficient resources to resolve the model as an integer LP. If there is, it continues on and solves for an integer solution. If not, it stops with the noninteger solution. If the iteration count needs to be increased, it should be at least double that of the count for the noninteger solution.

In digital imagery analysis, the weighting of the two channels can be used to identify subtleties within a given land cover type. The weights themselves do not help in contrasting the land cover type from the background. Rather, the spectral bounds perform this function. Once all the feasible pixels have been identified, the lambda weights can be used to select a particular type of water pixel. In Figure 39 the dots represent the pixels which fall within the Channel 1 and Channel 3 spectral bounds for water (0 - 22 and 5 - 200, respectively). This parallelepiped classifier is further divided into four quadrants. The use of the lambda weights will cause the model to select pixels from certain quadrants if possible. Reviewing the possible weights and their corresponding quadrants will better explain this concept.

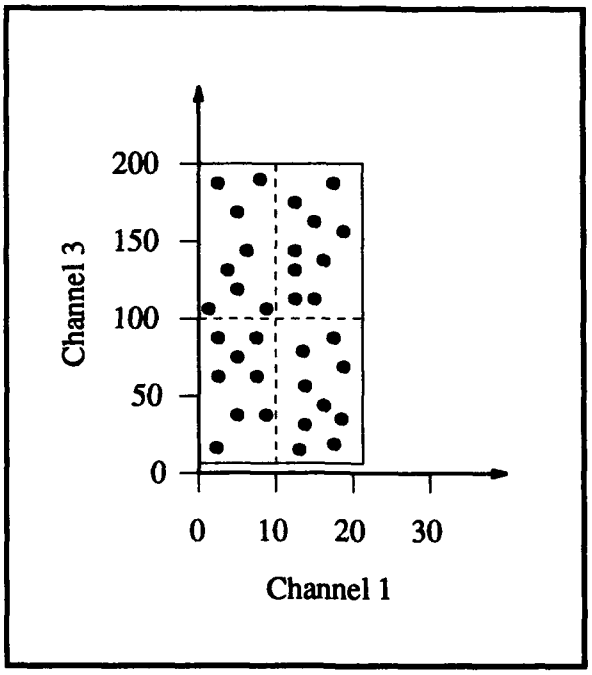

Figure 39. Parallelepiped Classifier For Water

If Channel 3 is given a weight of 1.0, the model will not care what the corresponding gray value of Channel 1 is when selecting pixels to maximize the objective function. As such, the model will attempt to select those pixels which are located above the horizontal dashed line. If Channel 1 is given a weight of 1.0, a similar selection occurs. In this case, the model does not care what the corresponding gray values of the Channel 3 pixels are. In this case, the model will attempt to select pixels to the right of the vertical dashed line. In both examples, I have said the model *will attempt to select* because there may not be enough pixels in one of the quadrants to allow the model to exclusively selected from that quadrant.

As the lambda weights change from 0/1.0 for Channel 1/Channel 3, the quadrants of emphasis change. For lambda weights 0/1.0 to .4/.6, the pixels in the upper half of the parallelepiped classifier will have priority in being selected. With lambda weights of .5/.5, both channels have equal priority in being selected. As such, pixels in the upper right quadrant should have priority in being selected. The reason for

the model selecting this quadrant is that both channels have maximum gray values in this quadrant. When reviewing Tables 34 and 35, there is not much of a noticeable change when the lambda weights change from .4/.6 to .5/.5. The reason why there is not a noticeable change in the pixels selected is that the pixel gray values for Channel 3 are so high when compared to Channel 1. Because of this difference in gray values, the Channel 3 pixels will tend to dominate the selection process. As the lambda weights change from .5/.5 to .9/.1, the pixels on the right of the vertical dashed line should have priority in the selection process. When the the two referenced tables are again reviewed, this change in pixel selection priority is not seen. Once again the reason is because the Channel 3 pixels are dominating the selection process. When the lambda weight changes from .9/.1 to 1.0/0, there is a noticeable change in the pixels selected. This result occurred since the selection process is not influenced by the high Channel 3 gray values.

In general, the image analyst can change the priority of the pixels selected by changing the lambda weights. However, the effectiveness of this method greatly depends on the target and its spectral characteristics. In the data set used for this thesis, this method's utility was not fully realized since Channel 3 dominated the selection process. If the spectral range for water in Channel 3 had been more restrictive, this method of changing the selection priority would have been better demonstrated. If an image analyst desires to change the selection priority to pixels on the other side of the dashed lines, the objective function only needs to be changed to a minimization objective function. If a minimization function is used, Channel 1 will dominate the selection process. However, this dominance will not be as great when compared to Channel 3's dominance with a maximize objective function. This concept is more clearly seen when the water spectral maximums and minimums are recalled. For Channel 1, the water spectral range is 0 - 22, while for Channel 3, the range is 5 - 200. Given these numbers, it is easy to see how a gray value of 200 dominates over 22, while the dominance of 0 over 5 is only slight. If the model was

run with a minimize objective function, I would expect to see more changes in the pixels selected as the lambda weights changed.

The actual physical interpretation of what each of these quadrants represent would require a larger data set and further analysis. Due to the relatively tight spectral bounds for Channel 1, a partial interpretation can be made for this channel. The pixels closest to 0 in gray value were most representative of water while pixels close to 22 represented a transition from water to another land cover type. Due to the wide spectral range for water in Channel 3, no similar statements could be made. If the spectral range in Channel 3 had been tighter, I would have been able to make a similar interpretation. Combining the interpretation of the two channels would then allow one to place a physical interpretation on each of the four quadrants in the parallelepiped classifier. In the next paragraph, an improvement for changing the lambda weights is discussed.

In the GAMS implementation of the model, I had to manually change the lambda weights and rerun each program. With an "ADBASE" type implementation of this model, all the lambda weights will be automatically tested and the efficient frontier identified.[3] With the ADBASE implementation of the model, the analyst's confidence level should rise in knowing that he or she has a nondominated solution.

5.3 Vegetation Indices and Band Ratio

For this part of the research, I was not able to proceed past the calculation of the vegetation indices and the band ratio. There are three possible answers why this occurred with the SPOT imagery. The first factor is that the vegetation indices were specifically developed for the NOAA AVHRR system. Since the red and near-IR imaging bands are not the same in both systems, different solutions would result from the two types of imagery. A second and more significant factor that may have

[3]If the reader is unfamiliar with the ADBASE software, he or she should contact Dr. Yupo Chan of the Operational Sciences Department at the Air Force Institute of Technology in Dayton, OH.

caused this problem is the preprocessing that occurred on the SPOT imagery. One of the preprocessing techniques used was a multipoint linear stretch. This enhancement technique alone may have caused this problem. A third factor that could cause the gray values to be out of the normal range is the imaging conditions. This is a catch all factor that considers the following: illumination angle/time of day, moisture content of gravel and soil, and weather. All of these factors can affect the gray values that are recorded for a scene. Once again, if the gray values change out of their characteristic spectral ranges then the combination methods explained may not be valid.

I suspect that if raw (unprocessed) SPOT imagery was available, these methods for combining two bands of imagery would work. The spectral response curves, for land covers present in the SPOT image, show the basis for these combination methods and interpretation. If raw imagery was not available, the original gray values could still be computed if the algorithm for the multipoint linear stretch was known.

5.4 Comparison of Processing Methods

Since I was not able to process the combined images created by the vegetation indices, I was not able to compare which data set resulted in the most accurate answer nor which method was the most computer execution time efficient. I would suspect that the computer execution time would be the same since the number of equations for each method should also be the same. As far as the accuracy of the solution rendered from two different data sets, it is hard to say. The vegetation index method combines the channels geometrically while the weighted sums approach is additive. Once again the data set with the most spectrally distinct target will be the superior one.

5.5 *Creation of GAMS Programs*

As the size of the image processing problem grows, so does the amount of data needed in a GAMS program. Anyone using this model should take advantage of the FORTRAN programs I have created for this express purpose. The adjacency and cost matrix each consisted of 1,000 lines of data.

5.6 *Final Conclusion*

This thesis effort successfully incorporated the spectral and spatial characteristics of targets into an pattern recognition model. In the past, most research has focused on only using the spectral characteristics of a target to locate it in an image. Research continues in this area as well as in the spatial pattern recognition area. Few models today attempt to incorporate both the spectral and spatial patterns in finding objects of interest. This thesis is a step forward in marrying these two methods of finding targets in digital imagery.

Appendix A. *LINDO Computer Runs*

In this appendix are contained four completed LINDO runs explained in Chapter 3. The title of each run is designated with a six digit alpha numeric designation to indicate the type of run being made. The first two digits indicate the number of subregions to be found in the image. The next two digits indicates the size of the first subregion and the last two digits indicates the size of the second subregion. For example, the first LINDO run is designated S2-1-1. This title indicates that there are two subregions to be found both with a area of one pixel. The complete LINDO outputs were not included due to the the volume of output. However, I have included the complete LINDO program for each programs so that the results can be replicated.

A.1 LINDO S2-1-1, Run #1

```
 !  Search for two subregions of sizes: one and one.  Border lengths:
 !  four and four.
 !

MIN     11 X11 + 5 X21 + 6 X31 + 8 X41 + 13 X51 + 2 X61 + 5 X71 + X81
     + X91 + 11 X12 + 5 X22 + 6 X32 + 8 X42 + 13 X52 + 2 X62 + 5 X72 + X82
     + X92
SUBJECT TO
       2)   X11 + X21 + X31 + X41 + X51 + X61 + X71 + X81 + X91 =    1
       3)   X12 + X22 + X32 + X42 + X52 + X62 + X72 + X82 + X92 =    1
       4)   X11 - X21 - P121 + N121 =    0
       5)   X11 - X41 - P141 + N141 =    0
       6) - X11 + X21 - P211 + N211 =    0
       7)   X21 - X51 - P251 + N251 =    0
       8)   X21 - X31 - P231 + N231 =    0
       9) - X[illegible] + X31 - P321 + N321 =    0
      10)   X31   X61 - P361 + N361 =    0
      11) - X11 + X41 - P411 + N411 =    0
      12)   X41 - X51 - P451 + N451 =    0
      13)   X41 - X71 - P471 + N471 =    0
```

14) - X21 + X51 - P521 + N521 = 0
15) - X41 + X51 - P541 + N541 = 0
16) X51 - X61 - P561 + N561 = 0
17) X51 - X81 - P581 + N581 = 0
18) - X31 + X61 - P631 + N631 = 0
19) - X51 + X61 - P651 + N651 = 0
20) X61 - X91 - P691 + N691 = 0
21) - X41 + X71 - P741 + N741 = 0
22) X71 - X81 - P781 + N781 = 0
23) - X71 + X81 - P871 + N871 = 0
24) - X51 + X81 - P851 + N851 = 0
25) X81 - X91 - P891 + N891 = 0
26) - X61 + X91 - P961 + N961 = 0
27) - X81 + X91 - P981 + N981 = 0
28) X12 - X22 - P122 + N122 = 0
29) X12 - X42 - P142 + N142 = 0
30) - X12 + X22 - P212 + N212 = 0
31) X22 - X52 - P252 + N252 = 0
32) X22 - X32 - P232 + N232 = 0
33) - X22 + X32 - P322 + N322 = 0
34) X32 - X62 - P362 + N362 = 0
35) - X12 + X42 - P412 + N412 = 0
36) X42 - X52 - P452 + N452 = 0
37) X42 - X72 - P472 + N472 = 0
38) - X22 + X52 - P522 + N522 = 0
39) - X42 + X52 - P542 + N542 = 0
40) X52 - X62 - P562 + N562 = 0
41) X52 - X82 - P582 + N582 = 0
42) - X32 + X62 - P632 + N632 = 0
43) - X52 + X62 - P652 + N652 = 0
44) X62 - X92 - P692 + N692 = 0
45) - X42 + X72 - P742 + N742 = 0
46) X72 - X82 - P782 + N782 = 0
47) - X72 + X82 - P872 + N872 = 0
48) - X52 + X82 - P852 + N852 = 0
49) X82 - X92 - P892 + N892 = 0
50) - X62 + X92 - P962 + N962 = 0
51) - X82 + X92 - P982 + N982 = 0
52) P121 + N121 + P141 + N141 + P211 + N211 + P251 + N251 + P231
+ N231 + P321 + N321 + P361 + N361 + P411 + N411 + P451 + N451 + P471
+ N471 + P521 + N521 + P541 + N541 + P561 + N561 + P581 + N581 + P631
+ N631 + P651 + N651 + P691 + N691 + P741 + N741 + P781 + N781 + P871

```
     + N871 + P851 + N851 + P891 + N891 + P961 + N961 + P981 + N981
     =    4
      53)   P122 + N122 + P142 + N142 + P212 + N212 + P252 + N252 + P232
     + N232 + P322 + N322 + P362 + N362 + P412 + N412 + P452 + N452 + P472
     + N472 + P522 + N522 + P542 + N542 + P562 + N562 + P582 + N582 + P632
     + N632 + P652 + N652 + P692 + N692 + P742 + N742 + P782 + N782 + P872
     + N872 + P852 + N852 + P892 + N892 + P962 + N962 + P982 + N982
     =    4
      54)   X11 + X12 <=   1
      55)   X21 + X22 <=   1
      56)   X31 + X32 <=   1
      57)   X41 + X42 <=   1
      58)   X51 + X52 <=   1
      59)   X61 + X62 <=   1
      60)   X71 + X72 <=   1
      61)   X81 + X82 <=   1
      62)   X91 + X92 <=   1
END
INTE   114


      OBJECTIVE FUNCTION VALUE

      1)     6.0000000

VARIABLE        VALUE          REDUCED COST
     X11          .000000         11.000000
     X21          .000000          5.000000
     X31          .000000          6.000000
     X41          .000000          8.000000
     X51          .000000         13.000000
     X61          .000000          2.000000
     X71         1.000000          5.000000
     X81          .000000          1.000000
     X91          .000000          1.000000
     X12          .000000         11.000000
     X22          .000000          5.000000
     X32          .000000          6.000000
     X42          .000000          8.000000
     X52          .000000         13.000000
     X62          .000000          2.000000
     X72          .000000          5.000000
```

X82	.000000	1.000000
X92	1.000000	1.000000
P121	.000000	.000000
N121	.000000	.000000
P141	.000000	.000000
N141	.000000	.000000
P211	.000000	.000000
N211	.000000	.000000
P251	.000000	.000000
N251	.000000	.000000
P231	.000000	.000000
N231	.000000	.000000
P321	.000000	.000000
N321	.000000	.000000
P361	.000000	.000000
N361	.000000	.000000
P411	.000000	.000000
N411	.000000	.000000
P451	.000000	.000000
N451	.000000	.000000
P471	.000000	.000000
N471	1.000000	.000000
P521	.000000	.000000
N521	.000000	.000000
P541	.000000	.000000
N541	.000000	.000000
P561	.000000	.000000
N561	.000000	.000000
P581	.000000	.000000
N581	.000000	.000000
P631	.000000	.000000
N631	.000000	.000000
P651	.000000	.000000
N651	.000000	.000000
P691	.000000	.000000
N691	.000000	.000000
P741	1.000000	.000000
N741	.000000	.000000
P781	1.000000	.000000
N781	.000000	.000000
P871	.000000	.000000
N871	1.000000	.000000

P851	.000000	.000000
N851	.000000	.000000
P891	.000000	.000000
N891	.000000	.000000
P961	.000000	.000000
N961	.000000	.000000
P981	.000000	.000000
N981	.000000	.000000
P122	.000000	.000000
N122	.000000	.000000
P142	.000000	.000000
N142	.000000	.000000
P212	.000000	.000000
N212	.000000	.000000
P252	.000000	.000000
N252	.000000	.000000
P232	.000000	.000000
N232	.000000	.000000
P322	.000000	.000000
N322	.000000	.000000
P362	.000000	.000000
N362	.000000	.000000
P412	.000000	.000000
N412	.000000	.000000
P452	.000000	.000000
N452	.000000	.000000
P472	.000000	.000000
N472	.000000	.000000
P522	.000000	.000000
N522	.000000	.000000
P542	.000000	.000000
N542	.000000	.000000
P562	.000000	.000000
N562	.000000	.000000
P582	.000000	.000000
N582	.000000	.000000
P632	.000000	.000000
N632	.000000	.000000
P652	.000000	.000000
N652	.000000	.000000
P692	.000000	.000000
N692	1.000000	.000000

P742	.000000	.000000
N742	.000000	.000000
P782	.000000	.000000
N782	.000000	.000000
P872	.000000	.000000
N872	.000000	.000000
P852	.000000	.000000
N852	.000000	.000000
P892	.000000	.000000
N892	1.000000	.000000
P962	1.000000	.000000
N962	.000000	.000000
P982	1.000000	.000000
N982	.000000	.000000

NO. ITERATIONS= 289
BRANCHES= 6 DETERM.= 1.000E 0

A.2 LINDO S2-1-1, Run #2

```
 !  Search for two subregions of sizes: one and one.  Border lengths:
 !  three and two.
 !

MIN     11 X11 + 5 X21 + 6 X31 + 8 X41 + 13 X51 + 2 X61 + 5 X71 + X81
     + X91 + 11 X12 + 5 X22 + 6 X32 + 8 X42 + 13 X52 + 2 X62 + 5 X72 + X82
     + X92
SUBJECT TO
       ! Size constraint for the two subregions.
       2)   X11 + X21 + X31 + X41 + X51 + X61 + X71 + X81 + X91 =    1
       3)   X12 + X22 + X32 + X42 + X52 + X62 + X72 + X82 + X92 =    1
       ! Contiguity constraint.
       4)   X11 - X21 - P121 + N121 =    0
       5)   X11 - X41 - P141 + N141 =    0
       6) - X11 + X21 - P211 + N211 =    0
       7)   X21 - X51 - P251 + N251 =    0
       8)   X21 - X31 - P231 + N231 =    0
       9) - X21 + X31 - P321 + N321 =    0
      10)   X31 - X61 - P361 + N361 =    0
      11) - X11 + X41 - P411 + N411 =    0
      12)   X41 - X51 - P451 + N451 =    0
      13)   X41 - X71 - P471 + N471 =    0
      14) - X21 + X51 - P521 + N521 =    0
      15) - X41 + X51 - P541 + N541 =    0
      16)   X51 - X61 - P561 + N561 =    0
      17)   X51 - X81 - P581 + N581 =    0
      18) - X31 + X61 - P631 + N631 =    0
      19) - X51 + X61 - P651 + N651 =    0
      20)   X61 - X91 - P691 + N691 =    0
      21) - X41 + X71 - P741 + N741 =    0
      22)   X71 - X81 - P781 + N781 =    0
      23) - X71 + X81 - P871 + N871 =    0
      24) - X51 + X81 - P851 + N851 =    0
      25)   X81 - X91 - P891 + N891 =    0
      26) - X61 + X91 - P961 + N961 =    0
      27) - X81 + X91 - P981 + N981 =    0
      28)   X12 - X22 - P122 + N122 =    0
      29)   X12 - X42 - P142 + N142 =    0
      30) - X12 + X22 - P212 + N212 =    0
      31)   X22 - X52 - P252 + N252 =    0
```

```
32)   X22 - X32 - P232 + N232 =    0
33) - X22 + X32 - P322 + N322 =    0
34)   X32 - X62 - P362 + N362 =    0
35) - X12 + X42 - P412 + N412 =    0
36)   X42 - X52 - P452 + N452 =    0
37)   X42 - X72 - P472 + N472 =    0
38) - X22 + X52 - P522 + N522 =    0
39) - X42 + X52 - P542 + N542 =    0
40)   X52 - X62 - P562 + N562 =    0
41)   X52 - X82 - P582 + N582 =    0
42) - X32 + X62 - P632 + N632 =    0
43) - X52 + X62 - P652 + N652 =    0
44)   X62 - X92 - P692 + N692 =    0
45) - X42 + X72 - P742 + N742 =    0
46)   X72 - X82 - P782 + N782 =    0
47) - X72 + X82 - P872 + N872 =    0
48) - X52 + X82 - P852 + N852 =    0
49)   X82 - X92 - P892 + N892 =    0
50) - X62 + X92 - P962 + N962 =    0
51) - X82 + X92 - P982 + N982 =    0
! Border length constraints.
52)   P121 + N121 + P141 + N141 + P211 + N211 + P251 + N251 + P231
+ N231 + P321 + N321 + P361 + N361 + P411 + N411 + P451 + N451 + P471
+ N471 + P521 + N521 + P541 + N541 + P561 + N561 + P581 + N581 + P631
+ N631 + P651 + N651 + P691 + N691 + P741 + N741 + P781 + N781 + P871
+ N871 + P851 + N851 + P891 + N891 + P961 + N961 + P981 + N981
=    3
53)   P122 + N122 + P142 + N142 + P212 + N212 + P252 + N252 + P232
+ N232 + P322 + N322 + P362 + N362 + P412 + N412 + P452 + N452 + P472
+ N472 + P522 + N522 + P542 + N542 + P562 + N562 + P582 + N582 + P632
+ N632 + P652 + N652 + P692 + N692 + P742 + N742 + P782 + N782 + P872
+ N872 + P852 + N852 + P892 + N892 + P962 + N962 + P982 + N982
=    2
! Each pixel only assigned to one subregion.
54)  x11 + x12 <=    1
55)  x21 + x22 <=    1
56)  x31 + x32 <=    1
57)  x41 + x42 <=    1
58)  x51 + x52 <=    1
59)  x61 + x62 <=    1
60)  x71 + x72 <=    1
61)  x81 + x82 <=    1
```

```
      62)  x91 + x92 <=    1

      ! All variables defined to be 0 or 1.
 INTE    114
```

No output for this run.

A.3 LINDO S2-1-1, Run #4

```
  !  Search for two subregions of sizes: one and one.  Border lengths:
  !  six and eight.
  !
MIN     11 X11 + 5 X21 + 6 X31 + 8 X41 + 13 X51 + 2 X61 + 5 X71 + X81
     + X91 + 11 X12 + 5 X22 + 6 X32 + 8 X42 + 13 X52 + 2 X62 + 5 X72 + X82
     + X92
SUBJECT TO
       2)   X11 + X21 + X31 + X41 + X51 + X61 + X71 + X81 + X91 =    1
       3)   X12 + X22 + X32 + X42 + X52 + X62 + X72 + X82 + X92 =    1
       4)   X11 - X21 - P121 + N121 =    0
       5)   X11 - X41 - P141 + N141 =    0
       6) - X11 + X21 - P211 + N211 =    0
       7)   X21 - X51 - P251 + N251 =    0
       8)   X21 - X31 - P231 + N231 =    0
       9) - X21 + X31 - P321 + N321 =    0
      10)   X31 - X61 - P361 + N361 =    0
      11) - X11 + X41 - P411 + N411 =    0
      12)   X41 - X51 - P451 + N451 =    0
      13)   X41 - X71 - P471 + N471 =    0
      14) - X21 + X51 - P521 + N521 =    0
      15) - X41 + X51 - P541 + N541 =    0
      16)   X51 - X61 - P561 + N561 =    0
      17)   X51 - X81 - P581 + N581 =    0
      18) - X31 + X61 - P631 + N631 =    0
      19) - X51 + X61 - P651 + N651 =    0
      20)   X61 - X91 - P691 + N691 =    0
      21) - X41 + X71 - P741 + N741 =    0
      22)   X71 - X81 - P781 + N781 =    0
      23) - X71 + X81 - P871 + N871 =    0
      24) - X51 + X81 - P851 + N851 =    0
      25)   X81 - X91 - P891 + N891 =    0
      26) - X61 + X91 - P961 + N961 =    0
      27) - X81 + X91 - P981 + N981 =    0
      28)   X12 - X22 - P122 + N122 =    0
      29)   X12 - X42 - P142 + N142 =    0
      30) - X12 + X22 - P212 + N212 =    0
      31)   X22 - X52 - P252 + N252 =    0
      32)   X22 - X32 - P232 + N232 =    0
      33) - X22 + X32 - P322 + N322 =    0
      34)   X32 - X62 - P362 + N362 =    0
```

```
 35) - X12 + X42 - P412 + N412 =    0
 36)   X42 - X52 - P452 + N452 =    0
 37)   X42 - X72 - P472 + N472 =    0
 38) - X22 + X52 - P522 + N522 =    0
 39) - X42 + X52 - P542 + N542 =    0
 40)   X52 - X62 - P562 + N562 =    0
 41)   X52 - X82 - P582 + N582 =    0
 42) - X32 + X62 - P632 + N632 =    0
 43) - X52 + X62 - P652 + N652 =    0
 44)   X62 - X92 - P692 + N692 =    0
 45) - X42 + X72 - P742 + N742 =    0
 46)   X72 - X82 - P782 + N782 =    0
 47) - X72 + X82 - P872 + N872 =    0
 48) - X52 + X82 - P852 + N852 =    0
 49)   X82 - X92 - P892 + N892 =    0
 50) - X62 + X92 - P962 + N962 =    0
 51) - X82 + X92 - P982 + N982 =    0
 52)   P121 + N121 + P141 + N141 + P211 + N211 + P251 + N251 + P231
+ N231 + P321 + N321 + P361 + N361 + P411 + N411 + P451 + N451 + P471
+ N471 + P521 + N521 + P541 + N541 + P561 + N561 + P581 + N581 + P631
+ N631 + P651 + N651 + P691 + N691 + P741 + N741 + P781 + N781 + P871
+ N871 + P851 + N851 + P891 + N891 + P961 + N961 + P981 + N981
=    6
 53)   P122 + N122 + P142 + N142 + P212 + N212 + P252 + N252 + P232
+ N232 + P322 + N322 + P362 + N362 + P412 + N412 + P452 + N452 + P472
+ N472 + P522 + N522 + P542 + N542 + P562 + N562 + P582 + N582 + P632
+ N632 + P652 + N652 + P692 + N692 + P742 + N742 + P782 + N782 + P872
+ N872 + P852 + N852 + P892 + N892 + P962 + N962 + P982 + N982
=    8
 54)   X11 + X12 <=   1
 55)   X21 + X22 <=   1
 56)   X31 + X32 <=   1
 57)   X41 + X42 <=   1
 58)   X51 + X52 <=   1
 59)   X61 + X62 <=   1
 60)   X71 + X72 <=   1
 61)   X81 + X82 <=   1
 62)   X91 + X92 <=   1
END
INTE   114
```

OBJECTIVE FUNCTION VALUE

1) 2.0000000

VARIABLE	VALUE	REDUCED COST
X11	.000000	11.000000
X21	.000000	5.000000
X31	.000000	6.000000
X41	.000000	8.000000
X51	.000000	13.000000
X61	.000000	2.000000
X71	.000000	5.000000
X81	1.000000	1.000000
X91	.000000	1.000000
X12	.000000	11.000000
X22	.000000	5.000000
X32	.000000	6.000000
X42	.000000	8.000000
X52	.000000	13.000000
X62	.000000	2.000000
X72	.000000	5.000000
X82	.000000	1.000000
X92	1.000000	1.000000
P121	.000000	.000000
N121	.000000	.000000
P141	.000000	.000000
N141	.000000	.000000
P211	.000000	.000000
N211	.000000	.000000
P251	.000000	.000000
N251	.000000	.000000
P231	.000000	.000000
N231	.000000	.000000
P321	.000000	.000000
N321	.000000	.000000
P361	.000000	.000000
N361	.000000	.000000
P411	.000000	.000000
N411	.000000	.000000
P451	.000000	.000000
N451	.000000	.000000
P471	.000000	.000000

N471	.000000	.000000
P521	.000000	.000000
N521	.000000	.000000
P541	.000000	.000000
N541	.000000	.000000
P561	.000000	.000000
N561	.000000	.000000
P581	.000000	.000000
N581	1.000000	.000000
P631	.000000	.000000
N631	.000000	.000000
P651	.000000	.000000
N651	.000000	.000000
P691	.000000	.000000
N691	.000000	.000000
P741	.000000	.000000
N741	.000000	.000000
P781	.000000	.000000
N781	1.000000	.000000
P871	1.000000	.000000
N871	.000000	.000000
P851	1.000000	.000000
N851	.000000	.000000
P891	1.000000	.000000
N891	.000000	.000000
P961	.000000	.000000
N961	.000000	.000000
P981	.000000	.000000
N981	1.000000	.000000
P122	.000000	.000000
N122	.000000	.000000
P142	.000000	.000000
N142	.000000	.000000
P212	.000000	.000000
N212	.000000	.000000
P252	.000000	.000000
N252	.000000	.000000
P232	.000000	.000000
N232	.000000	.000000
P322	.000000	.000000
N322	.000000	.000000
P362	1.000000	.000000

N362	1.000000	.000000
P412	.000000	.000000
N412	.000000	.000000
P452	.000000	.000000
N452	.000000	.000000
P472	.000000	.000000
N472	.000000	.000000
P522	.000000	.000000
N522	.000000	.000000
P542	.000000	.000000
N542	.000000	.000000
P562	.000000	.000000
N562	.000000	.000000
P582	1.000000	.000000
N582	1.000000	.000000
P632	.000000	.000000
N632	.000000	.000000
P652	.000000	.000000
N652	.000000	.000000
P692	.000000	.000000
N692	1.000000	.000000
P742	.000000	.000000
N742	.000000	.000000
P782	.000000	.000000
N782	.000000	.000000
P872	.000000	.000000
N872	.000000	.000000
P852	.000000	.000000
N852	.000000	.000000
P892	.000000	.000000
N892	1.000000	.000000
P962	1.000000	.000000
N962	.000000	.000000
P982	1.000000	.000000
N982	.000000	.000000

NO. ITERATIONS= 105
BRANCHES= 1 DETERM.= 1.000E 0

A.4 LINDO S2-1-1, Run #5

```
  !  Search for two subregions of sizes: one and one.  Border lengths:
  !  four and six.
  !
MIN     11 X11 + 5 X21 + 6 X31 + 8 X41 + 13 X51 + 2 X61 + 5 X71 + X81
     + X91 + 11 X12 + 5 X22 + 6 X32 + 8 X42 + 13 X52 + 2 X62 + 5 X72 + X82
     + X92
SUBJECT TO
       2)   X11 + X21 + X31 + X41 + X51 + X61 + X71 + X81 + X91 =    1
       3)   X12 + X22 + X32 + X42 + X52 + X62 + X72 + X82 + X92 =    1
       4)   X11 - X21 - P121 + N121 =    0
       5)   X11 - X41 - P141 + N141 =    0
       6) - X11 + X21 - P211 + N211 =    0
       7)   X21 - X51 - P251 + N251 =    0
       8)   X21 - X31 - P231 + N231 =    0
       9) - X21 + X31 - P321 + N321 =    0
      10)   X31 - X61 - P361 + N361 =    0
      11) - X11 + X41 - P411 + N411 =    0
      12)   X41 - X51 - P451 + N451 =    0
      13)   X41 - X71 - P471 + N471 =    0
      14) - X21 + X51 - P521 + N521 =    0
      15) - X41 + X51 - P541 + N541 =    0
      16)   X51 - X61 - P561 + N561 =    0
      17)   X51 - X81 - P581 + N581 =    0
      18) - X31 + X61 - P631 + N631 =    0
      19) - X51 + X61 - P651 + N651 =    0
      20)   X61 - X91 - P691 + N691 =    0
      21) - X41 + X71 - P741 + N741 =    0
      22)   X71 - X81 - P781 + N781 =    0
      23) - X71 + X81 - P871 + N871 =    0
      24) - X51 + X81 - P851 + N851 =    0
      25)   X81 - X91 - P891 + N891 =    0
      26) - X61 + X91 - P961 + N961 =    0
      27) - X81 + X91 - P981 + N981 =    0
      28)   X12 - X22 - P122 + N122 =    0
      29)   X12 - X42 - P142 + N142 =    0
      30) - X12 + X22 - P212 + N212 =    0
      31)   X22 - X52 - P252 + N252 =    0
      32)   X22 - X32 - P232 + N232 =    0
      33) - X22 + X32 - P322 + N322 =    0
      34)   X32 - X62 - P362 + N362 =    0
```

```
 35) - X12 + X42 - P412 + N412 =    0
 36)   X42 - X52 - P452 + N452 =    0
 37)   X42 - X72 - P472 + N472 =    0
 38) - X22 + X52 - P522 + N522 =    0
 39) - X42 + X52 - P542 + N542 =    0
 40)   X52 - X62 - P562 + N562 =    0
 41)   X52 - X82 - P582 + N582 =    0
 42) - X32 + X62 - P632 + N632 =    0
 43) - X52 + X62 - P652 + N652 =    0
 44)   X62 - X92 - P692 + N692 =    0
 45) - X42 + X72 - P742 + N742 =    0
 46)   X72 - X82 - P782 + N782 =    0
 47) - X72 + X82 - P872 + N872 =    0
 48) - X52 + X82 - P852 + N852 =    0
 49)   X82 - X92 - P892 + N892 =    0
 50) - X62 + X92 - P962 + N962 =    0
 51) - X82 + X92 - P982 + N982 =    0
 52)   P121 + N121 + P141 + N141 + P211 + N211 + P251 + N251 + P231
+ N231 + P321 + N321 + P361 + N361 + P411 + N411 + P451 + N451 + P471
+ N471 + P521 + N521 + P541 + N541 + P561 + N561 + P581 + N581 + P631
+ N631 + P651 + N651 + P691 + N691 + P741 + N741 + P781 + N781 + P871
+ N871 + P851 + N851 + P891 + N891 + P961 + N961 + P981 + N981
=    4
 53)   P122 + N122 + P142 + N142 + P212 + N212 + P252 + N252 + P232
+ N232 + P322 + N322 + P362 + N362 + P412 + N412 + P452 + N452 + P472
+ N472 + P522 + N522 + P542 + N542 + P562 + N562 + P582 + N582 + P632
+ N632 + P652 + N652 + P692 + N692 + P742 + N742 + P782 + N782 + P872
+ N872 + P852 + N852 + P892 + N892 + P962 + N962 + P982 + N982
=    6
 54)   X11 + X12 <=   1
 55)   X21 + X22 <=   1
 56)   X31 + X32 <=   1
 57)   X41 + X42 <=   1
 58)   X51 + X52 <=   1
 59)   X61 + X62 <=   1
 60)   X71 + X72 <=   1
 61)   X81 + X82 <=   1
 62)   X91 + X92 <=   1
END
INTE   114
```

OBJECTIVE FUNCTION VALUE

1) 2.0000000

VARIABLE	VALUE	REDUCED COST
X11	.000000	11.000000
X21	.000000	5.000000
X31	.000000	6.000000
X41	.000000	8.000000
X51	.000000	13.000000
X61	.000000	2.000000
X71	.000000	5.000000
X81	.000000	1.000000
X91	1.000000	1.000000
X12	.000000	11.000000
X22	.000000	5.000000
X32	.000000	6.000000
X42	.000000	8.000000
X52	.000000	13.000000
X62	.000000	2.000000
X72	.000000	5.000000
X82	1.000000	1.000000
X92	.000000	1.000000
P121	.000000	.000000
N121	.000000	.000000
P141	.000000	.000000
N141	.000000	.000000
P211	.000000	.000000
N211	.000000	.000000
P251	.000000	.000000
N251	.000000	.000000
P231	.000000	.000000
N231	.000000	.000000
P321	.000000	.000000
N321	.000000	.000000
P361	.000000	.000000
N361	.000000	.000000
P411	.000000	.000000
N411	.000000	.000000
P451	.000000	.000000
N451	.000000	.000000
P471	.000000	.000000

N471	.000000	.000000
P521	.000000	.000000
N521	.000000	.000000
P541	.000000	.000000
N541	.000000	.000000
P561	.000000	.000000
N561	.000000	.000000
P581	.000000	.000000
N581	.000000	.000000
P631	.000000	.000000
N631	.000000	.000000
P651	.000000	.000000
N651	.000000	.000000
P691	.000000	.000000
N691	1.000000	.000000
P741	.000000	.000000
N741	.000000	.000000
P781	.000000	.000000
N781	.000000	.000000
P871	.000000	.000000
N871	.000000	.000000
P851	.000000	.000000
N851	.000000	.000000
P891	.000000	.000000
N891	1.000000	.000000
P961	1.000000	.000000
N961	.000000	.000000
P981	1.000000	.000000
N981	.000000	.000000
P122	.000000	.000000
N122	.000000	.000000
P142	.000000	.000000
N142	.000000	.000000
P212	.000000	.000000
N212	.000000	.000000
P252	.000000	.000000
N252	.000000	.000000
P232	.000000	.000000
N232	.000000	.000000
P322	.000000	.000000
N322	.000000	.000000
P362	.000000	.000000

N362	.000000	.000000
P412	.000000	.000000
N412	.000000	.000000
P452	.000000	.000000
N452	.000000	.000000
P472	.000000	.000000
N472	.000000	.000000
P522	.000000	.000000
N522	.000000	.000000
P542	.000000	.000000
N542	.000000	.000000
P562	.000000	.000000
N562	.000000	.000000
P582	.000000	.000000
N582	1.000000	.000000
P632	.000000	.000000
N632	.000000	.000000
P652	.000000	.000000
N652	.000000	.000000
P692	.000000	.000000
N692	.000000	.000000
P742	.000000	.000000
N742	.000000	.000000
P782	.000000	.000000
N782	1.000000	.000000
P872	1.000000	.000000
N872	.000000	.000000
P852	1.000000	.000000
N852	.000000	.000000
P892	1.000000	.000000
N892	.000000	.000000
P962	.000000	.000000
N962	.000000	.000000
P982	.000000	.000000
N982	1.000000	.000000

NO. ITERATIONS= 88
BRANCHES= 1 DETERM.= 1.000E 0

A.5 LINDO S2-1-1, Run #6

```
 !  Search for two subregions of sizes: one and one.  Border lengths:
 !  eight and eight.
 !
MIN     11 X11 + 5 X21 + 6 X31 + 8 X41 + 13 X51 + 2 X61 + 5 X71 + X81
     + X91 + 11 X12 + 5 X22 + 6 X32 + 8 X42 + 13 X52 + 2 X62 + 5 X72 + X82
     + X92
SUBJECT TO
       ! Size constraint for the two subregions.
       2)   X11 + X21 + X31 + X41 + X51 + X61 + X71 + X81 + X91 =    1
       3)   X12 + X22 + X32 + X42 + X52 + X62 + X72 + X82 + X92 =    1
       ! Contiguity constraint.
       4)   X11 - X21 - P121 + N121 =    0
       5)   X11 - X41 - P141 + N141 =    0
       6) - X11 + X21 - P211 + N211 =    0
       7)   X21 - X51 - P251 + N251 =    0
       8)   X21 - X31 - P231 + N231 =    0
       9) - X21 + X31 - P321 + N321 =    0
      10)   X31 - X61 - P361 + N361 =    0
      11) - X11 + X41 - P411 + N411 =    0
      12)   X41 - X51 - P451 + N451 =    0
      13)   X41 - X71 - P471 + N471 =    0
      14) - X21 + X51 - P521 + N521 =    0
      15) - X41 + X51 - P541 + N541 =    0
      16)   X51 - X61 - P561 + N561 =    0
      17)   X51 - X81 - P581 + N581 =    0
      18) - X31 + X61 - P631 + N631 =    0
      19) - X51 + X61 - P651 + N651 =    0
      20)   X61 - X91 - P691 + N691 =    0
      21) - X41 + X71 - P741 + N741 =    0
      22)   X71 - X81 - P781 + N781 =    0
      23) - X71 + X81 - P871 + N871 =    0
      24) - X51 + X81 - P851 + N851 =    0
      25)   X81 - X91 - P891 + N891 =    0
      26) - X61 + X91 - P961 + N961 =    0
      27) - X81 + X91 - P981 + N981 =    0
      28)   X12 - X22 - P122 + N122 =    0
      29)   X12 - X42 - P142 + N142 =    0
      30) - X12 + X22 - P212 + N212 =    0
      31)   X22 - X52 - P252 + N252 =    0
      32)   X22 - X32 - P232 + N232 =    0
```

```
33) - X22 + X32 - P322 + N322 =    0
34)   X32 - X62 - P362 + N362 =    0
35) - X12 + X42 - P412 + N412 =    0
36)   X42 - X52 - P452 + N452 =    0
37)   X42 - X72 - P472 + N472 =    0
38) - X22 + X52 - P522 + N522 =    0
39) - X42 + X52 - P542 + N542 =    0
40)   X52 - X62 - P562 + N562 =    0
41)   X52 - X82 - P582 + N582 =    0
42) - X32 + X62 - P632 + N632 =    0
43) - X52 + X62 - P652 + N652 =    0
44)   X62 - X92 - P692 + N692 =    0
45) - X42 + X72 - P742 + N742 =    0
46)   X72 - X82 - P782 + N782 =    0
47) - X72 + X82 - P872 + N872 =    0
48) - X52 + X82 - P852 + N852 =    0
49)   X82 - X92 - P892 + N892 =    0
50) - X62 + X92 - P962 + N962 =    0
51) - X82 + X92 - P982 + N982 =    0
! Border length constraints.
52)   P121 + N121 + P141 + N141 + P211 + N211 + P251 + N251 + P231
+ N231 + P321 + N321 + P361 + N361 + P411 + N411 + P451 + N451 + P471
+ N471 + P521 + N521 + P541 + N541 + P561 + N561 + P581 + N581 + P631
+ N631 + P651 + N651 + P691 + N691 + P741 + N741 + P781 + N781 + P871
+ N871 + P851 + N851 + P891 + N891 + P961 + N961 + P981 + N981
=    8
53)   P122 + N122 + P142 + N142 + P212 + N212 + P252 + N252 + P232
+ N232 + P322 + N322 + P362 + N362 + P412 + N412 + P452 + N452 + P472
+ N472 + P522 + N522 + P542 + N542 + P562 + N562 + P582 + N582 + P632
+ N632 + P652 + N652 + P692 + N692 + P742 + N742 + P782 + N782 + P872
+ N872 + P852 + N852 + P892 + N892 + P962 + N962 + P982 + N982
=    8
! Each pixel only assigned to one subregion.
54)  x11 + x12 <=    1
55)  x21 + x22 <=    1
56)  x31 + x32 <=    1
57)  x41 + x42 <=    1
58)  x51 + x52 <=    1
59)  x61 + x62 <=    1
60)  x71 + x72 <=    1
61)  x81 + x82 <=    1
62)  x91 + x92 <=    1
```

```
      ! All variables defined to be 0 or 1.
 INTE    114



LP OPTIMUM FOUND AT STEP    105
OBJECTIVE VALUE =   2.00000000
FIX ALL VARS.(   14)  WITH RC >   1.00000
SET      X81 TO >=    1 AT    1, BND=  -2.000     TWIN= -2.000        118

NEW INTEGER SOLUTION OF    2.00000000     AT BRANCH      1 PIVOT    118

      OBJECTIVE FUNCTION VALUE

      1)     2.0000000

 VARIABLE        VALUE          REDUCED COST
      X11          .000000         10.000000
      X21          .000000          4.000000
      X31          .000000          5.000000
      X41          .000000          7.000000
      X51          .000000         12.000000
      X61          .000000          1.000000
      X71          .000000          4.000000
      X81         1.000000           .000000
      X91          .000000           .000000
      X12          .000000         10.000000
      X22          .000000          4.000000
      X32          .000000          5.000000
      X42          .000000          7.000000
      X52          .000000         12.000000
      X62          .000000          1.000000
      X72          .000000          4.000000
      X82          .000000           .000000
      X92         1.000000           .000000
     P121          .000000           .000000
     N121          .000000           .000000
     P141          .000000           .000000
     N141          .000000           .000000
     P211          .000000           .000000
```

N211	.000000	.000000
P251	.000000	.000000
N251	.000000	.000000
P231	1.000000	.000000
N231	1.000000	.000000
P321	.000000	.000000
N321	.000000	.000000
P361	.000000	.000000
N361	.000000	.000000
P411	.000000	.000000
N411	.000000	.000000
P451	.000000	.000000
N451	.000000	.000000
P471	.000000	.000000
N471	.000000	.000000
P521	.000000	.000000
N521	.000000	.000000
P541	.000000	.000000
N541	.000000	.000000
P561	.000000	.000000
N561	.000000	.000000
P581	.000000	.000000
N581	1.000000	.000000
P631	.000000	.000000
N631	.000000	.000000
P651	.000000	.000000
N651	.000000	.000000
P691	.000000	.000000
N691	.000000	.000000
P741	.000000	.000000
N741	.000000	.000000
P781	.000000	.000000
N781	1.000000	.000000
P871	1.000000	.000000
N871	.000000	.000000
P851	1.000000	.000000
N851	.000000	.000000
P891	1.000000	.000000
N891	.000000	.000000
P961	.000000	.000000
N961	.000000	.000000
P981	.000000	.000000

N981	1.000000	.000000
P122	.000000	.000000
N122	.000000	.000000
P142	.000000	.000000
N142	.000000	.000000
P212	1.000000	.000000
N212	1.000000	.000000
P252	.000000	.000000
N252	.000000	.000000
P232	.000000	.000000
N232	.000000	.000000
P322	.000000	.000000
N322	.000000	.000000
P362	1.000000	.000000
N362	1.000000	.000000
P412	.000000	.000000
N412	.000000	.000000
P452	.000000	.000000
N452	.000000	.000000
P472	.000000	.000000
N472	.000000	.000000
P522	.000000	.000000
N522	.000000	.000000
P542	.000000	.000000
N542	.000000	.000000
P562	.000000	.000000
N562	.000000	.000000
P582	.000000	.000000
N582	.000000	.000000
P632	.000000	.000000
N632	.000000	.000000
P652	.000000	.000000
N652	.000000	.000000
P692	.000000	.000000
N692	1.000000	.000000
P742	.000000	.000000
N742	.000000	.000000
P782	.000000	.000000
N782	.000000	.000000
P872	.000000	.000000
N872	.000000	.000000
P852	.000000	.000000

```
N852          .000000          .000000
P892          .000000          .000000
N892         1.000000          .000000
P962         1.000000          .000000
N962          .000000          .000000
P982         1.000000          .000000
N982          .000000          .000000


NO. ITERATIONS=     118
BRANCHES=    1 DETERM.=  1.000E    0
BOUND ON OPTIMUM:  2.000000
DELETE      X81 AT LEVEL     1
ENUMERATION COMPLETE. BRANCHES=     1 PIVOTS=     118

LAST INTEGER SOLUTION IS THE BEST FOUND
RE-INSTALLING BEST SOLUTION...
```

Appendix B. *GAMS Computer Runs With Image Frame*

In this appendix are contained seven GAMS programs and outputs. The progrmas are complete while the output is just a summary of the results. This reduction in the output was necesssitated due to the lenthy nature of the GAMS output. The same titling convention is used in this chapter as was used in Appendix A. Note a complete listing of the GAMS output is possible by running the programs below.

B.1 GAMS S2-1-1, Run #1

This program is set to find two subregions both of one-pixel size. The border length constraints have been set to eight and eight.

```
GAMS 2.20 VAX VMS                                  16-AUG-1992 16:55  PAGE     1
G E N E R A L   A L G E B R A I C   M O D E L I N G   S Y S T E M
C O M P I L A T I O N


  1  SETS
  2     I / 1*25/
  3     K / 1*2/
  4  ALIAS(I,J)
  5  SETS
  6     ADJ(I,J)  ADJACENCY MATRIX
  7         /1.(2,6)
  8          2.(1,3,7)
  9          3.(2,4,8)
 10          4.(3,5,9)
 11          5.(4,10)
 12          6.(1,7,11)
 13          7.(2,6,8,12)
 14          8.(3,7,9,13)
 15          9.(4,8,10,14)
 16          10.(5,9,15)
 17          11.(6,12,16)
 18          12.(7,11,13,17)
 19          13.(8,12,14,18)
 20          14.(9,13,15,19)
```

```
 21          15.(10,14,20)
 22          16.(11,17,21)
 23          17.(12,16,18,22)
 24          18.(13,17,19,23)
 25          19.(14,18,20,24)
 26          20.(15,19,25)
 27          21.(16,22)
 28          22.(17,21,23)
 29          23.(18,22,24)
 30          24.(19,23,25)
 31          25.(20,24)/
 32
 33     PARAMETERS
 34         M(K) PIXEL SIZE OF KTH SUBREGION
 35           /1 1, 2 1/
 36
 37         L(K) BORDER OF KTH SUBREGION
 38           /1 8, 2 8/
 39
 40         C(I) COST OF EACH PIXEL
 41           / 1 99,  2 99,  3 99,  4 99,  5 99
 42             6 99,  7 11,  8  5,  9  6, 10 99
 43            11 99, 12  8, 13 13, 14  2, 15 99
 44            16 99, 17  5, 18  1, 19  1, 20 99
 45            21 99, 22 99, 23 99, 24 99, 25 99/;
 46
 47     VARIABLES
 48        Z          TOTAL COST OF ACQUIRED PIXELS
 49        X(I,K)     BINARY VARIABLE
 50        P(I,J,K)   BINARY VARIABLE
 51        N(I,J,K)   BINARY VARIABLE
 52
 53     BINARY VARIABLES X,P,N;
 54
GAMS 2.20 VAX VMS                                16-AUG-1992 16:55  PAGE      2
G E N E R A L   A L G E B R A I C   M O D E L I N G   S Y S T E M
C O M P I L A T I O N


 55     EQUATIONS
 56        OBJ        OBJECTIVE FUNCTION
 57        C1(K)      CONSTRAINT FOR SIZE OF KTH SUBREGION
 58        C2(I,J,K)  CONSTRAINT FOR CONTIGUITY
 59        C3(K)      CONSTRAINT FOR BORDER LENGTH
 60        C4(I,J,K)  CONSTRAINT FOR P&N MUTUAL EXCLUSION
```

```
61        C5(I)     CONSTRAINT FOR SUBREGIONS;
62
63     OBJ..
64        Z =E= SUM((I,K), C(I)*X(I,K));
65
66     C1(K)..
67        SUM(I,X(I,K)) =E= M(K);
68
69     C2(I,J,K) $ADJ(I,J)..
70        X(I,K) - X(J,K) - P(I,J,K) + N(I,J,K) =E= 0;
71
72     C3(K)..
73        SUM((I,J) $ADJ(I,J), P(I,J,K) + N(I,J,K)) =E= L(K);
74
75     C4(I,J,K) $ADJ(I,J)..
76        P(I,J,K) + N(I,J,K) =L= 1;
77
78     C5(I)..
79        SUM(K, X(I,K)) =L= 1;
80
81     MODEL SUBREGION /ALL/;
82
83     OPTIONS ITERLIM = 1000000, RESLIM = 1000000, WORK = 20000;
84
85     SOLVE SUBREGION USING MIP MINIMIZING Z;
86
87     OPTION MIP=ZOOM;
88
89     DISPLAY X.L,P.L,N.L;
90
GAMS 2.20 VAX VMS                                16-AUG-1992 17:00  PAGE     30
G E N E R A L   A L G E B R A I C   M O D E L I N G   S Y S T E M
E X E C U T I N G


----     89 VARIABLE  X.L          BINARY VARIABLE

             1          2

18       1.000
19                  1.000


----     89 VARIABLE  P.L          BINARY VARIABLE
```

```
                1          2

18.13       1.000
18.17       1.000
18.19       1.000
18.23       1.000
19.14                  1.000
19.18                  1.000
19.20                  1.000
19.24                  1.000


----     89 VARIABLE  N.L          BINARY VARIABLE

                1          2

13.18       1.000
14.19                  1.000
17.18       1.000
18.19                  1.000
19.18       1.000
20.19                  1.000
23.18       1.000
24.19                  1.000


**** FILE SUMMARY

INPUT   GS092D:[JAMRINE]S25X5.GMS;4
OUTPUT  GS092D:[JAMRINE]S25X5.LIS;6

EXECUTION TIME       =        2.000 SECONDS
```

B.2 GAMS S2-1-3, Run #2

This program is set to find two subregions of size one and three. The border length constraints have been set to eight and sixteen.

```
GAMS 2.20 VAX VMS                                    16-AUG-1992 20:11  PAGE      1
G E N E R A L   A L G E B R A I C   M O D E L I N G   S Y S T E M
C O M P I L A T I O N


  1  SETS
  2     I / 1*25/
  3     K / 1*2/
  4  ALIAS(I,J)
  5  SETS
  6     ADJ(I,J)  ADJACENCY MATRIX
  7         /1.(2,6)
  8          2.(1,3,7)
  9          3.(2,4,8)
 10          4.(3,5,9)
 11          5.(4,10)
 12          6.(1,7,11)
 13          7.(2,6,8,12)
 14          8.(3,7,9,13)
 15          9.(4,8,10,14)
 16          10.(5,9,15)
 17          11.(6,12,16)
 18          12.(7,11,13,17)
 19          13.(8,12,14,18)
 20          14.(9,13,15,19)
 21          15.(10,14,20)
 22          16.(11,17,21)
 23          17.(12,16,18,22)
 24          18.(13,17,19,23)
 25          19.(14,18,20,24)
 26          20.(15,19,25)
 27          21.(16,22)
 28          22.(17,21,23)
 29          23.(18,22,24)
 30          24.(19,23,25)
 31          25.(20,24)/
 32
 33     PARAMETERS
```

```
 34          M(K) PIXEL SIZE OF KTH SUBREGION
 35            /1 1, 2 3/
 36
 37          L(K) BORDER OF KTH SUBREGION
 38            /1 8, 2 16/
 39
 40          C(I) COST OF EACH PIXEL
 41            / 1 99,  2 99,  3 99,  4 99,  5 99
 42              6 99,  7 11,  8  5,  9  6, 10 99
 43             11 99, 12  8, 13 13, 14  2, 15 99
 44             16 99, 17  5, 18  1, 19  1, 20 99
 45             21 99, 22 99, 23 99, 24 99, 25 99/;
 46
 47     VARIABLES
 48        Z          TOTAL COST OF ACQUIRED PIXELS
 49        X(I,K)     BINARY VARIABLE
 50        P(I,J,K)   BINARY VARIABLE
 51        N(I,J,K)   BINARY VARIABLE
 52
 53     BINARY VARIABLES X,P,N;
 54
GAMS 2.20 VAX VMS                                16-AUG-1992 20:11  PAGE      2
G E N E R A L   A L G E B R A I C   M O D E L I N G   S Y S T E M
C O M P I L A T I O N


 55     EQUATIONS
 56        OBJ        OBJECTIVE FUNCTION
 57        C1(K)      CONSTRAINT FOR SIZE OF KTH SUBREGION
 58        C2(I,J,K)  CONSTRAINT FOR CONTIGUITY
 59        C3(K)      CONSTRAINT FOR BORDER LENGTH
 60        C4(I,J,K)  CONSTRAINT FOR P&N MUTUAL EXCLUSION
 61        C5(I)      CONSTRAINT FOR SUBREGIONS;
 62
 63     OBJ..
 64        Z =E= SUM((I,K), C(I)*X(I,K));
 65
 66     C1(K)..
 67        SUM(I,X(I,K)) =E= M(K);
 68
 69     C2(I,J,K) $ADJ(I,J)..
 70        X(I,K) - X(J,K) - P(I,J,K) + N(I,J,K) =E= 0;
 71
 72     C3(K)..
 73        SUM((I,J) $ADJ(I,J), P(I,J,K) + N(I,J,K)) =E= L(K);
```

```
 74
 75     C4(I,J,K) $ADJ(I,J)..
 76        P(I,J,K) + N(I,J,K) =L= 1;
 77
 78     C5(I)..
 79        SUM(K, X(I,K)) =L= 1;
 80
 81     MODEL SUBREGION /ALL/;
 82
 83     OPTIONS ITERLIM = 1000000, RESLIM = 1000000, WORK = 20000;
 84
 85     SOLVE SUBREGION USING MIP MINIMIZING Z;
 86
 87     OPTION MIP=ZOOM;
 88
 89     DISPLAY X.L,P.L,N.L;
 90
GAMS 2.20 VAX VMS                                16-AUG-1992 20:16  PAGE     30
G E N E R A L   A L G E B R A I C   M O D E L I N G   S Y S T E M
E X E C U T I N G


----     89 VARIABLE  X.L          BINARY VARIABLE

             1          2

14       1.000
17                  1.000
18                  1.000
19                  1.000


----     89 VARIABLE  P.L          BINARY VARIABLE

                1          2

14.9        1.000
14.13       1.000
14.15       1.000
14.19       1.000
17.12                   1.000
17.16                   1.000
17.22                   1.000
18.13                   1.000
18.23                   1.000
```

```
19.14                    1.000
19.20                    1.000
19.24                    1.000


----     89 VARIABLE  N.L          BINARY VARIABLE

                1          2

9 .14       1.000
12.17                    1.000
13.14       1.000
13.18                    1.000
14.19                    1.000
15.14       1.000
16.17                    1.000
19.14       1.000
20.19                    1.000
22.17                    1.000
23.18                    1.000
24.19                    1.000


**** FILE SUMMARY

INPUT   GS092D:[JAMRINE]S25X51.GMS;1
OUTPUT  GS092D:[JAMRINE]S25X51.LIS;1

EXECUTION TIME       =        2.500 SECONDS
```

B.3 GAMS S2-2-5, Run #3

This program is set to find two subregions of size two and five. The border length constraints have been set to twelve and twenty four.

```
GAMS 2.20 VAX VMS                                  16-AUG-1992 20:16  PAGE     1
G E N E R A L   A L G E B R A I C   M O D E L I N G   S Y S T E M
C O M P I L A T I O N


  1  SETS
  2     I / 1*25/
  3     K / 1*2/
  4  ALIAS(I,J)
  5  SETS
  6     ADJ(I,J)  ADJACENCY MATRIX
  7         /1.(2,6)
  8          2.(1,3,7)
  9          3.(2,4,8)
 10          4.(3,5,9)
 11          5.(4,10)
 12          6.(1,7,11)
 13          7.(2,6,8,12)
 14          8.(3,7,9,13)
 15          9.(4,8,10,14)
 16          10.(5,9,15)
 17          11.(6,12,16)
 18          12.(7,11,13,17)
 19          13.(8,12,14,18)
 20          14.(9,13,15,19)
 21          15.(10,14,20)
 22          16.(11,17,21)
 23          17.(12,16,18,22)
 24          18.(13,17,19,23)
 25          19.(14,18,20,24)
 26          20.(15,19,25)
 27          21.(16,22)
 28          22.(17,21,23)
 29          23.(18,22,24)
 30          24.(19,23,25)
 31          25.(20,24)/
 32
 33     PARAMETERS
```

```
34          M(K) PIXEL SIZE OF KTH SUBREGION
35            /1 2, 2 5/
36
37          L(K) BORDER OF KTH SUBREGION
38            /1 12, 2 24/
39
40          C(I) COST OF EACH PIXEL
41            / 1 99,  2 99,  3 99,  4 99,  5 99
42              6 99,  7 11,  8  5,  9  6, 10 99
43             11 99, 12  8, 13 13, 14  2, 15 99
44             16 99, 17  5, 18  1, 19  1, 20 99
45             21 99, 22 99, 23 99, 24 99, 25 99/;
46
47     VARIABLES
48        Z          TOTAL COST OF ACQUIRED PIXELS
49        X(I,K)     BINARY VARIABLE
50        P(I,J,K)   BINARY VARIABLE
51        N(I,J,K)   BINARY VARIABLE
52
53     BINARY VARIABLES X,P,N;
54
```



```
55     EQUATIONS
56        OBJ        OBJECTIVE FUNCTION
57        C1(K)      CONSTRAINT FOR SIZE OF KTH SUBREGION
58        C2(I,J,K)  CONSTRAINT FOR CONTIGUITY
59        C3(K)      CONSTRAINT FOR BORDER LENGTH
60        C4(I,J,K)  CONSTRAINT FOR P&N MUTUAL EXCLUSION
61        C5(I)      CONSTRAINT FOR SUBREGIONS;
62
63     OBJ..
64        Z =E= SUM((I,K), C(I)*X(I,K));
65
66     C1(K)..
67        SUM(I,X(I,K)) =E= M(K);
68
69     C2(I,J,K) $ADJ(I,J)..
70        X(I,K) - X(J,K) - P(I,J,K) + N(I,J,K) =E= 0;
71
72     C3(K)..
73        SUM((I,J) $ADJ(I,J), P(I,J,K) + N(I,J,K)) =E= L(K);
```

```
 74
 75     C4(I,J,K) $ADJ(I,J)..
 76        P(I,J,K) + N(I,J,K) =L= 1;
 77
 78     C5(I)..
 79        SUM(K, X(I,K)) =L= 1;
 80
 81     MODEL SUBREGION /ALL/;
 82
 83     OPTIONS ITERLIM = 1000000, RESLIM = 1000000, WORK = 20000;
 84
 85     SOLVE SUBREGION USING MIP MINIMIZING Z;
 86
 87     OPTION MIP=ZOOM;
 88
 89     DISPLAY X.L,P.L,N.L;
 90
GAMS 2.20 VAX VMS                                16-AUG-1992 20:22  PAGE     30
G E N E R A L   A L G E B R A I C   M O D E L I N G   S Y S T E M
E X E C U T I N G


----     89 VARIABLE  X.L          BINARY VARIABLE

             1          2

8                    1.000
9                    1.000
12       1.000
14                   1.000
17       1.000
18                   1.000
19                   1.000


----     89 VARIABLE  P.L          BINARY VARIABLE

                1          2

8 .3                   1.000
8 .7                   1.000
8 .13                  1.000
9 .4                   1.000
9 .10                  1.000
12.7        1.000
```

```
12.11       1.000
12.13       1.000
14.13                   1.000
14.15                   1.000
17.16       1.000
17.18       1.000
17.22       1.000
18.13                   1.000
18.17                   1.000
18.23                   1.000
19.20                   1.000
19.24                   1.000


----     89 VARIABLE  N.L          BINARY VARIABLE

                1          2

3 .8                    1.000
4 .9                    1.000
7 .8                    1.000
7 .12       1.000
10.9                    1.000
11.12       1.000
13.8                    1.000
13.12       1.000
13.14                   1.000
13.18                   1.000
15.14                   1.000
16.17       1.000
17.18                   1.000
GAMS 2.20 VAX VMS                                16-AUG-1992 20:22  PAGE     31
G E N E R A L   A L G E B R A I C   M O D E L I N G   S Y S T E M
E X E C U T I N G


         89 VARIABLE  N.L          BINARY VARIABLE

                1          2

18.17       1.000
20.19                   1.000
22.17       1.000
23.18                   1.000
24.19                   1.000
```

```
**** FILE SUMMARY

INPUT   GS092D:[JAMRINE]S25X52.GMS;1
OUTPUT  GS092D:[JAMRINE]S25X52.LIS;1

EXECUTION TIME       =        2.180 SECONDS
```

B.4 GAMS S2-3-3, Run #4

This program is set to find two subregions of size three and three. The border length constraints have been set to sixteen and sixteen.

```
GAMS 2.20 VAX VMS                                  16-AUG-1992 20:22  PAGE      1
G E N E R A L   A L G E B R A I C   M O D E L I N G   S Y S T E M
C O M P I L A T I O N


  1  SETS
  2     I / 1*25/
  3     K / 1*2/
  4  ALIAS(I,J)
  5  SETS
  6     ADJ(I,J)  ADJACENCY MATRIX
  7         /1.(2,6)
  8          2.(1,3,7)
  9          3.(2,4,8)
 10          4.(3,5,9)
 11          5.(4,10)
 12          6.(1,7,11)
 13          7.(2,6,8,12)
 14          8.(3,7,9,13)
 15          9.(4,8,10,14)
 16          10.(5,9,15)
 17          11.(6,12,16)
 18          12.(7,11,13,17)
 19          13.(8,12,14,18)
 20          14.(9,13,15,19)
 21          15.(10,14,20)
 22          16.(11,17,21)
 23          17.(12,16,18,22)
 24          18.(13,17,19,23)
 25          19.(14,18,20,24)
 26          20.(15,19,25)
 27          21.(16,22)
 28          22.(17,21,23)
 29          23.(18,22,24)
 30          24.(19,23,25)
 31          25.(20,24)/
 32
 33     PARAMETERS
```

```
34          M(K) PIXEL SIZE OF KTH SUBREGION
35            /1 3, 2 3/
36
37          L(K) BORDER OF KTH SUBREGION
38            /1 16, 2 16/
39
40          C(I) COST OF EACH PIXEL
41            / 1 99,  2 99,  3 99,  4 99,  5 99
42              6 99,  7 11,  8  5,  9  6, 10 99
43             11 99, 12  8, 13 13, 14  2, 15 99
44             16 99, 17  5, 18  1, 19  1, 20 99
45             21 99, 22 99, 23 99, 24 99, 25 99/;
46
47     VARIABLES
48        Z          TOTAL COST OF ACQUIRED PIXELS
49        X(I,K)     BINARY VARIABLE
50        P(I,J,K)   BINARY VARIABLE
51        N(I,J,K)   BINARY VARIABLE
52
53     BINARY VARIABLES X,P,N;
54
```



```
55     EQUATIONS
56        OBJ        OBJECTIVE FUNCTION
57        C1(K)      CONSTRAINT FOR SIZE OF KTH SUBREGION
58        C2(I,J,K)  CONSTRAINT FOR CONTIGUITY
59        C3(K)      CONSTRAINT FOR BORDER LENGTH
60        C4(I,J,K)  CONSTRAINT FOR P&N MUTUAL EXCLUSION
61        C5(I)      CONSTRAINT FOR SUBREGIONS;
62
63     OBJ..
64        Z =E= SUM((I,K), C(I)*X(I,K));
65
66     C1(K)..
67        SUM(I,X(I,K)) =E= M(K);
68
69     C2(I,J,K) $ADJ(I,J)..
70        X(I,K) - X(J,K) - P(I,J,K) + N(I,J,K) =E= 0;
71
72     C3(K)..
73        SUM((I,J) $ADJ(I,J), P(I,J,K) + N(I,J,K)) =E= L(K);
```

```
 74
 75     C4(I,J,K) $ADJ(I,J)..
 76        P(I,J,K) + N(I,J,K) =L= 1;
 77
 78     C5(I)..
 79        SUM(K, X(I,K)) =L= 1;
 80
 81     MODEL SUBREGION /ALL/;
 82
 83     OPTIONS ITERLIM = 1000000, RESLIM = 1000000, WORK = 20000;
 84
 85     SOLVE SUBREGION USING MIP MINIMIZING Z;
 86
 87     OPTION MIP=ZOOM;
 88
 89     DISPLAY X.L,P.L,N.L;
 90
GAMS 2.20 VAX VMS                                16-AUG-1992 20:27  PAGE     30
G E N E R A L   A L G E B R A I C   M O D E L I N G   S Y S T E M
E X E C U T I N G


----     89 VARIABLE  X.L          BINARY VARIABLE

             1          2

8        1.000
9        1.000
14       1.000
17                  1.000
18                  1.000
19                  1.000


----     89 VARIABLE  P.L          BINARY VARIABLE

                1          2

8 .3        1.000
8 .7        1.000
8 .13       1.000
9 .4        1.000
9 .10       1.000
14.13       1.000
14.15       1.000
```

```
14.19        1.000
17.12                    1.000
17.16                    1.000
17.22                    1.000
18.13                    1.000
18.23                    1.000
19.14                    1.000
19.20                    1.000
19.24                    1.000


----      89 VARIABLE  N.L            BINARY VARIABLE

                1           2

3 .8         1.000
4 .9         1.000
7 .8         1.000
10.9         1.000
12.17                    1.000
13.8         1.000
13.14        1.000
13.18                    1.000
14.19                    1.000
15.14        1.000
16.17                    1.000
19.14        1.000
20.19                    1.000
22.17                    1.000
23.18                    1.000
24.19                    1.000
GAMS 2.20 VAX VMS                                16-AUG-1992 20:27  PAGE     31
G E N E R A L   A L G E B R A I C   M O D E L I N G   S Y S T E M
E X E C U T I N G



**** FILE SUMMARY

INPUT    GS092D:[JAMRINE]S25X53.GMS;1
OUTPUT   GS092D:[JAMRINE]S25X53.LIS;1

EXECUTION TIME       =        2.200 SECONDS
```

B.5 GAMS S2-1-4, Run #5

This program is set to find two subregions of size one and four. The border length constraints have been set to eight and twenty.

```
GAMS 2.20 VAX VMS                                  16-AUG-1992 20:28  PAGE      1
G E N E R A L   A L G E B R A I C   M O D E L I N G   S Y S T E M
C O M P I L A T I O N


  1  SETS
  2     I / 1*25/
  3     K / 1*2/
  4  ALIAS(I,J)
  5  SETS
  6     ADJ(I,J)  ADJACENCY MATRIX
  7         /1.(2,6)
  8          2.(1,3,7)
  9          3.(2,4,8)
 10          4.(3,5,9)
 11          5.(4,10)
 12          6.(1,7,11)
 13          7.(2,6,8,12)
 14          8.(3,7,9,13)
 15          9.(4,8,10,14)
 16          10.(5,9,15)
 17          11.(6,12,16)
 18          12.(7,11,13,17)
 19          13.(8,12,14,18)
 20          14.(9,13,15,19)
 21          15.(10,14,20)
 22          16.(11,17,21)
 23          17.(12,16,18,22)
 24          18.(13,17,19,23)
 25          19.(14,18,20,24)
 26          20.(15,19,25)
 27          21.(16,22)
 28          22.(17,21,23)
 29          23.(18,22,24)
 30          24.(19,23,25)
 31          25.(20,24)/
 32
 33     PARAMETERS
```

```
34          M(K) PIXEL SIZE OF KTH SUBREGION
35            /1 1, 2 4/
36
37          L(K) BORDER OF KTH SUBREGION
38            /1 8, 2 20/
39
40          C(I) COST OF EACH PIXEL
41            / 1 99,  2 99,  3 99,  4 99,  5 99
42              6 99,  7 11,  8  5,  9  6, 10 99
43             11 99, 12  8, 13 13, 14  2, 15 99
44             16 99, 17  5, 18  1, 19  1, 20 99
45             21 99, 22 99, 23 99, 24 99, 25 99/;
46
47     VARIABLES
48        Z          TOTAL COST OF ACQUIRED PIXELS
49        X(I,K)     BINARY VARIABLE
50        P(I,J,K)   BINARY VARIABLE
51        N(I,J,K)   BINARY VARIABLE
52
53     BINARY VARIABLES X,P,N;
54
```



```
55     EQUATIONS
56        OBJ        OBJECTIVE FUNCTION
57        C1(K)      CONSTRAINT FOR SIZE OF KTH SUBREGION
58        C2(I,J,K)  CONSTRAINT FOR CONTIGUITY
59        C3(K)      CONSTRAINT FOR BORDER LENGTH
60        C4(I,J,K)  CONSTRAINT FOR P&N MUTUAL EXCLUSION
61        C5(I)      CONSTRAINT FOR SUBREGIONS;
62
63     OBJ..
64        Z =E= SUM((I,K), C(I)*X(I,K));
65
66     C1(K)..
67        SUM(I,X(I,K)) =E= M(K);
68
69     C2(I,J,K) $ADJ(I,J)..
70        X(I,K) - X(J,K) - P(I,J,K) + N(I,J,K) =E= 0;
71
72     C3(K)..
73        SUM((I,J) $ADJ(I,J), P(I,J,K) + N(I,J,K)) =E= L(K);
```

```
 74
 75     C4(I,J,K) $ADJ(I,J)..
 76        P(I,J,K) + N(I,J,K) =L= 1;
 77
 78     C5(I)..
 79        SUM(K, X(I,K)) =L= 1;
 80
 81     MODEL SUBREGION /ALL/;
 82
 83     OPTIONS ITERLIM = 1000000, RESLIM = 1000000, WORK = 20000;
 84
 85     SOLVE SUBREGION USING MIP MINIMIZING Z;
 86
 87     OPTION MIP=ZOOM;
 88
 89     DISPLAY X.L,P.L,N.L;
 90
GAMS 2.20 VAX VMS                                16-AUG-1992 20:28  PAGE     30
G E N E R A L   A L G E B R A I C   M O D E L I N G   S Y S T E M
E X E C U T I N G


----     89 VARIABLE  X.L          BINARY VARIABLE

              1          2

8        1.000
14                  1.000
17                  1.000
18                  1.000
19                  1.000


----     89 VARIABLE  P.L          BINARY VARIABLE

                 1          2

8 .3        1.000
8 .7        1.000
8 .9        1.000
8 .13       1.000
14.9                    1.000
14.13                   1.000
14.15                   1.000
17.12                   1.000
```

```
17.16                    1.000
17.22                    1.000
18.13                    1.000
18.23                    1.000
19.20                    1.000
19.24                    1.000


----     89 VARIABLE  N.L          BINARY VARIABLE

                1          2

3 .8        1.000
7 .8        1.000
9 .8        1.000
9 .14                    1.000
12.17                    1.000
13.8        1.000
13.14                    1.000
13.18                    1.000
15.14                    1.000
16.17                    1.000
20.19                    1.000
22.17                    1.000
23.18                    1.000
24.19                    1.000
```



```
**** FILE SUMMARY

INPUT   GS092D:[JAMRINE]S25X54.GMS;1
OUTPUT  GS092D:[JAMRINE]S25X54.LIS;1

EXECUTION TIME       =        2.120 SECONDS
```

B.6 GAMS S2-1-5, Run #6

This program is set to find two subregions of size one and five. The border length constraints have been set to eight and twenty four.

```
GAMS 2.20 VAX VMS                                  16-AUG-1992 20:31  PAGE      1
G E N E R A L   A L G E B R A I C   M O D E L I N G   S Y S T E M
C O M P I L A T I O N


  1  SETS
  2     I / 1*25/
  3     K / 1*2/
  4  ALIAS(I,J)
  5  SETS
  6     ADJ(I,J)  ADJACENCY MATRIX
  7         /1.(2,6)
  8          2.(1,3,7)
  9          3.(2,4,8)
 10          4.(3,5,9)
 11          5.(4,10)
 12          6.(1,7,11)
 13          7.(2,6,8,12)
 14          8.(3,7,9,13)
 15          9.(4,8,10,14)
 16          10.(5,9,15)
 17          11.(6,12,16)
 18          12.(7,11,13,17)
 19          13.(8,12,14,18)
 20          14.(9,13,15,19)
 21          15.(10,14,20)
 22          16.(11,17,21)
 23          17.(12,16,18,22)
 24          18.(13,17,19,23)
 25          19.(14,18,20,24)
 26          20.(15,19,25)
 27          21.(16,22)
 28          22.(17,21,23)
 29          23.(18,22,24)
 30          24.(19,23,25)
 31          25.(20,24)/
 32
 33     PARAMETERS
```

```
34          M(K) PIXEL SIZE OF KTH SUBREGION
35            /1 1, 2 5/
36
37          L(K) BORDER OF KTH SUBREGION
38            /1 8, 2 24/
39
40          C(I) COST OF EACH PIXEL
41            / 1 99,  2 99,  3 99,  4 99,  5 99
42              6 99,  7 11,  8  5,  9  6, 10 99
43             11 99, 12  8, 13 13, 14  2, 15 99
44             16 99, 17  5, 18  1, 19  1, 20 99
45             21 99, 22 99, 23 99, 24 99, 25 99/;
46
47     VARIABLES
48        Z          TOTAL COST OF ACQUIRED PIXELS
49        X(I,K)     BINARY VARIABLE
50        P(I,J,K)   BINARY VARIABLE
51        N(I,J,K)   BINARY VARIABLE
52
53     BINARY VARIABLES X,P,N;
54
```



```
55     EQUATIONS
56        OBJ        OBJECTIVE FUNCTION
57        C1(K)      CONSTRAINT FOR SIZE OF KTH SUBREGION
58        C2(I,J,K)  CONSTRAINT FOR CONTIGUITY
59        C3(K)      CONSTRAINT FOR BORDER LENGTH
60        C4(I,J,K)  CONSTRAINT FOR P&N MUTUAL EXCLUSION
61        C5(I)      CONSTRAINT FOR SUBREGIONS;
62
63     OBJ..
64        Z =E= SUM((I,K), C(I)*X(I,K));
65
66     C1(K)..
67        SUM(I,X(I,K)) =E= M(K);
68
69     C2(I,J,K) $ADJ(I,J)..
70        X(I,K) - X(J,K) - P(I,J,K) + N(I,J,K) =E= 0;
71
72     C3(K)..
73        SUM((I,J) $ADJ(I,J), P(I,J,K) + N(I,J,K)) =E= L(K);
```

```
 74
 75     C4(I,J,K) $ADJ(I,J)..
 76        P(I,J,K) + N(I,J,K) =L= 1;
 77
 78     C5(I)..
 79        SUM(K, X(I,K)) =L= 1;
 80
 81     MODEL SUBREGION /ALL/;
 82
 83     OPTIONS ITERLIM = 1000000, RESLIM = 1000000, WORK = 20000;
 84
 85     SOLVE SUBREGION USING MIP MINIMIZING Z;
 86
 87     OPTION MIP=ZOOM;
 88
 89     DISPLAY X.L,P.L,N.L;
 90
GAMS 2.20 VAX VMS                                16-AUG-1992 20:36  PAGE     30
G E N E R A L   A L G E B R A I C   M O D E L I N G   S Y S T E M
E X E C U T I N G


----     89 VARIABLE  X.L          BINARY VARIABLE

             1          2

8        1.000
9                   1.000
14                  1.000
17                  1.000
18                  1.000
19                  1.000


----     89 VARIABLE  P.L          BINARY VARIABLE

                1          2

8 .3        1.000
8 .7        1.000
8 .9        1.000
8 .13       1.000
9 .4                    1.000
9 .8                    1.000
9 .10                   1.000
```

```
14.13                    1.000
14.15                    1.000
17.12                    1.000
17.16                    1.000
17.22                    1.000
18.13                    1.000
18.23                    1.000
19.20                    1.000
19.24                    1.000


----     89 VARIABLE  N.L          BINARY VARIABLE

                1          2

3 .8        1.000
4 .9                     1.000
7 .8        1.000
8 .9                     1.000
9 .8        1.000
10.9                     1.000
12.17                    1.000
13.8        1.000
13.14                    1.000
13.18                    1.000
15.14                    1.000
16.17                    1.000
20.19                    1.000
22.17                    1.000
23.18                    1.000
24.19                    1.000
GAMS 2.20 VAX VMS                                16-AUG-1992 20:36  PAGE     31
G E N E R A L   A L G E B R A I C   M O D E L I N G   S Y S T E M
E X E C U T I N G



**** FILE SUMMARY

INPUT   GS092D:[JAMRINE]S25X55.GMS;1
OUTPUT  GS092D:[JAMRINE]S25X55.LIS;1

EXECUTION TIME       =        2.330 SECONDS
```

B.7 GAMS S2-1-6, Run #7

This program is set to find two subregions of size one and six. The border length constraints have been set to eight and twenty.

```
GAMS 2.20 VAX VMS                                  17-AUG-1992 09:27  PAGE     1
G E N E R A L   A L G E B R A I C   M O D E L I N G   S Y S T E M
C O M P I L A T I O N


  1  SETS
  2     I / 1*25/
  3     K / 1*2/
  4  ALIAS(I,J)
  5  SETS
  6     ADJ(I,J)  ADJACENCY MATRIX
  7         /1.(2,6)
  8          2.(1,3,7)
  9          3.(2,4,8)
 10          4.(3,5,9)
 11          5.(4,10)
 12          6.(1,7,11)
 13          7.(2,6,8,12)
 14          8.(3,7,9,13)
 15          9.(4,8,10,14)
 16          10.(5,9,15)
 17          11.(6,12,16)
 18          12.(7,11,13,17)
 19          13.(8,12,14,18)
 20          14.(9,13,15,19)
 21          15.(10,14,20)
 22          16.(11,17,21)
 23          17.(12,16,18,22)
 24          18.(13,17,19,23)
 25          19.(14,18,20,24)
 26          20.(15,19,25)
 27          21.(16,22)
 28          22.(17,21,23)
 29          23.(18,22,24)
 30          24.(19,23,25)
 31          25.(20,24)/
 32
 33     PARAMETERS
```

```
 34         M(K) PIXEL SIZE OF KTH SUBREGION
 35           /1 1, 2 6/
 36
 37         L(K) BORDER OF KTH SUBREGION
 38           /1 8, 2 20/
 39
 40         C(I) COST OF EACH PIXEL
 41           / 1 99,  2 99,  3 99,  4 99,  5 99
 42             6 99,  7 11,  8  5,  9  6, 10 99
 43            11 99, 12  8, 13 13, 14  2, 15 99
 44            16 99, 17  5, 18  1, 19  1, 20 99
 45            21 99, 22 99, 23 99, 24 99, 25 99/;
 46
 47     VARIABLES
 48        Z          TOTAL COST OF ACQUIRED PIXELS
 49        X(I,K)     BINARY VARIABLE
 50        P(I,J,K)   BINARY VARIABLE
 51        N(I,J,K)   BINARY VARIABLE
 52
 53     BINARY VARIABLES X,P,N;
 54
```



```
 55     EQUATIONS
 56        OBJ        OBJECTIVE FUNCTION
 57        C1(K)      CONSTRAINT FOR SIZE OF KTH SUBREGION
 58        C2(I,J,K)  CONSTRAINT FOR CONTIGUITY
 59        C3(K)      CONSTRAINT FOR BORDER LENGTH
 60        C4(I,J,K)  CONSTRAINT FOR P&N MUTUAL EXCLUSION
 61        C5(I)      CONSTRAINT FOR SUBREGIONS;
 62
 63     OBJ..
 64        Z =E= SUM((I,K), C(I)*X(I,K));
 65
 66     C1(K)..
 67        SUM(I,X(I,K)) =E= M(K);
 68
 69     C2(I,J,K) $ADJ(I,J)..
 70        X(I,K) - X(J,K) - P(I,J,K) + N(I,J,K) =E= 0;
 71
 72     C3(K)..
 73        SUM((I,J) $ADJ(I,J), P(I,J,K) + N(I,J,K)) =E= L(K);
```

```
 74
 75     C4(I,J,K) $ADJ(I,J)..
 76        P(I,J,K) + N(I,J,K) =L= 1;
 77
 78     C5(I)..
 79        SUM(K, X(I,K)) =L= 1;
 80
 81     MODEL SUBREGION /ALL/;
 82
 83     OPTIONS ITERLIM = 1000000, RESLIM = 1000000, WORK = 20000;
 84
 85     SOLVE SUBREGION USING MIP MINIMIZING Z;
 86
 87     OPTION MIP=ZOOM;
 88
 89     DISPLAY X.L,P.L,N.L;
 90
GAMS 2.20 VAX VMS                                17-AUG-1992 09:34  PAGE     30
G E N E R A L   A L G E B R A I C   M O D E L I N G   S Y S T E M
E X E C U T I N G


----     89 VARIABLE  X.L           BINARY VARIABLE

             1           2

8                     1.000
9                     1.000
13                    1.000
14                    1.000
17       1.000
18                    1.000
19                    1.000


----     89 VARIABLE  P.L           BINARY VARIABLE

                 1           2

8 .3                    1.000
8 .7                    1.000
9 .4                    1.000
9 .10                   1.000
13.12                   1.000
14.15                   1.000
```

```
17.12        1.000
17.16        1.000
17.18        1.000
17.22        1.000
18.17                    1.000
18.23                    1.000
19.20                    1.000
19.24                    1.000


----      89 VARIABLE  N.L          BINARY VARIABLE

                1          2

3 .8                     1.000
4 .9                     1.000
7 .8                     1.000
10.9                     1.000
12.13                    1.000
12.17        1.000
15.14                    1.000
16.17        1.000
17.18                    1.000
18.17        1.000
20.19                    1.000
22.17        1.000
23.18                    1.000
24.19                    1.000
GAMS 2.20 VAX VMS                                17-AUG-1992 09:34  PAGE     31
G E N E R A L   A L G E B R A I C   M O D E L I N G   S Y S T E M
E X E C U T I N G



**** FILE SUMMARY

INPUT   GS092D:[JAMRINE]S25X56.GMS;2
OUTPUT  GS092D:[JAMRINE]S25X56.LIS;2

EXECUTION TIME       =        2.380 SECONDS
```

Appendix C. *GAMS Computer Runs; Min/Max Objective Function*

In this appendix, two GAMS computer runs are included. These runs show the effect of running the model with a minimize objective function and with a maximize objective function. In both programs, only one subregion of four-pixel size is being located.

C.1 GAMS S1-4, Run #1

```
GAMS 2.20 VAX VMS                                    18-AUG-1992 20:44  PAGE     1
G E N E R A L   A L G E B R A I C   M O D E L I N G   S Y S T E M
C O M P I L A T I O N


      SETS
         I / 1*36/
         K / 1*1/
      ALIAS(I,J)
      SETS
         ADJ(I,J)  ADJACENCY MATRIX
             /1.(2,7)
              2.(1,3,8)
              3.(2,4,9)
              4.(3,5,10)
              5.(4,6,11)
              6.(5,12)
              7.(1,8,13)
              8.(2,7,9,14)
              9.(3,8,10,15)
             10.(4,9,11,16)
             11.(5,10,12,17)
             12.(6,11,18)
             13.(7,14,19)
             14.(8,13,15,20)
             15.(9,14,16,21)
             16.(10,15,17,22)
             17.(11,16,18,23)
             18.(12,17,24)
```

```
 26              19.(13,20,25)
 27              20.(14,19,21,26)
 28              21.(15,20,22,27)
 29              22.(16,21,23,28)
 30              23.(17,22,24,29)
 31              24.(18,23,30)
 32              25.(19,26,31)
 33              26.(20,25,27,32)
 34              27.(21,26,28,33)
 35              28.(22,27,29,34)
 36              29.(23,28,30,35)
 37              30.(24,29,36)
 38              31.(25,32)
 39              32.(26,31,33)
 40              33.(27,32,34)
 41              34.(28,33,35)
 42              35.(29,34,36)
 43              36.(30,35)/
 44
 45          PARAMETERS
 46              M(K) PIXEL SIZE OF KTH SUBREGION
 47                 /1 4/
 48
 49              L(K) BORDER OF KTH SUBREGION
 50                 /1 20/
 51
 52              C(I) COST OF EACH PIXEL
 53                 / 1 99,  2 99,  3 99,  4 99,  5 99,  6 99
 54                   7 99,  8  1,  9  1, 10  1, 11  1, 12 99
```



```
 55                   13 99, 14  2, 15  2, 16  2, 17  2, 18 99
 56                   19 99, 20  3, 21  3, 22  3, 23  3, 24 99
 57                   25 99, 26  4, 27  4, 28  4, 29  4, 30 99
 58                   31 99, 32 99, 33 99, 34 99, 35 99, 36 99/;
 59
 60          VARIABLES
 61              Z          TOTAL COST OF ACQUIRED PIXELS
 62              X(I,K)     BINARY VARIABLE
 63              P(I,J,K)   BINARY VARIABLE
 64              N(I,J,K)   BINARY VARIABLE
 65
```

```
 66          BINARY VARIABLES X,P,N;
 67
 68          EQUATIONS
 69             OBJ       OBJECTIVE FUNCTION
 70             C1(K)     CONSTRAINT FOR SIZE OF KTH SUBREGION
 71             C2(I,J,K) CONSTRAINT FOR CONTIGUITY
 72             C3(K)     CONSTRAINT FOR BORDER LENGTH
 73             C4(I,J,K) CONSTRAINT FOR P&N MUTUAL EXCLUSION
 74             C5(I)     CONSTRAINT FOR SUBREGIONS;
 75
 76          OBJ..
 77             Z =E= SUM((I,K), C(I)*X(I,K));
 78
 79          C1(K)..
 80             SUM(I,X(I,K)) =E= M(K);
 81
 82          C2(I,J,K) $ADJ(I,J)..
 83             X(I,K) - X(J,K) - P(I,J,K) + N(I,J,K) =E= 0;
 84
 85          C3(K)..
 86             SUM((I,J) $ADJ(I,J), P(I,J,K) + N(I,J,K)) =E= L(K);
 87
 88          C4(I,J,K) $ADJ(I,J)..
 89             P(I,J,K) + N(I,J,K) =L= 1;
 90
 91          C5(I)..
 92             SUM(K, X(I,K)) =L= 1;
 93
 94          MODEL SUBREGION /ALL/;
 95
 96          OPTIONS ITERLIM = 1000000, RESLIM = 1000000, WORK = 20000;
 97
 98          SOLVE SUBREGION USING MIP MINIMIZING Z;
 99
100          OPTION MIP=ZOOM;
101
102         DISPLAY X.L,P.L,N.L;
103
104
```



```
----    102 VARIABLE  X.L          BINARY VARIABLE
```

```
              1

8         1.000
9         1.000
10        1.000
11        1.000


----     102 VARIABLE  P.L            BINARY VARIABLE

                 1

8 .2        1.000
8 .7        1.000
8 .14       1.000
9 .3        1.000
9 .15       1.000
10.4        1.000
10.16       1.000
11.5        1.000
11.12       1.000
11.17       1.000


----     102 VARIABLE  N.L            BINARY VARIABLE

                 1

2 .8        1.000
3 .9        1.000
4 .10       1.000
5 .11       1.000
7 .8        1.000
12.11       1.000
14.8        1.000
15.9        1.000
16.10       1.000
17.11       1.000


**** FILE SUMMARY

INPUT   GS092D:[JAMRINE]S26X6.GMS;1
OUTPUT  GS092D:[JAMRINE]S26X6.LIS;1
```

```
EXECUTION TIME        =        2.100 SECONDS
```

C.2 GAMS S1-4, Run #2

```
GAMS 2.20 VAX VMS                                18-AUG-1992 20:57  PAGE     1
G E N E R A L   A L G E B R A I C   M O D E L I N G   S Y S T E M
C O M P I L A T I O N


      SETS
         I / 1*36/
         K / 1*1/
      ALIAS(I,J)
      SETS
         ADJ(I,J)  ADJACENCY MATRIX
             /1.(2,7)
              2.(1,3,8)
              3.(2,4,9)
              4.(3,5,10)
              5.(4,6,11)
              6.(5,12)
              7.(1,8,13)
              8.(2,7,9,14)
              9.(3,8,10,15)
             10.(4,9,11,16)
             11.(5,10,12,17)
             12.(6,11,18)
             13.(7,14,19)
             14.(8,13,15,20)
             15.(9,14,16,21)
             16.(10,15,17,22)
             17.(11,16,18,23)
             18.(12,17,24)
             19.(13,20,25)
             20.(14,19,21,26)
             21.(15,20,22,27)
             22.(16,21,23,28)
             23.(17,22,24,29)
             24.(18,23,30)
             25.(19,26,31)
             26.(20,25,27,32)
             27.(21,26,28,33)
             28.(22,27,29,34)
             29.(23,28,30,35)
             30.(24,29,36)
```

```
 38            31.(25,32)
 39            32.(26,31,33)
 40            33.(27,32,34)
 41            34.(28,33,35)
 42            35.(29,34,36)
 43            36.(30,35)/
 44
 45         PARAMETERS
 46            M(K) PIXEL SIZE OF KTH SUBREGION
 47               /1 4/
 48
 49            L(K) BORDER OF KTH SUBREGION
 50               /1 20/
 51
 52            C(I) COST OF EACH PIXEL
 53               / 1 0,  2  0,  3  0,  4  0,  5  0,  6  0
 54                 7 0,  8  1,  9  1, 10  1, 11  1, 12  0
```



```
 55                13 0, 14  2, 15  2, 16  2, 17  2, 18  0
 56                19 0, 20  3, 21  3, 22  3, 23  3, 24  0
 57                25 0, 26  4, 27  4, 28  4, 29  4, 30  0
 58                31 0, 32  0, 33  0, 34  0, 35  0, 36  0/;
 59
 60          VARIABLES
 61             Z          TOTAL COST OF ACQUIRED PIXELS
 62             X(I,K)     BINARY VARIABLE
 63             P(I,J,K)   BINARY VARIABLE
 64             N(I,J,K)   BINARY VARIABLE
 65
 66          BINARY VARIABLES X,P,N;
 67
 68          EQUATIONS
 69             OBJ        OBJECTIVE FUNCTION
 70             C1(K)      CONSTRAINT FOR SIZE OF KTH SUBREGION
 71             C2(I,J,K)  CONSTRAINT FOR CONTIGUITY
 72             C3(K)      CONSTRAINT FOR BORDER LENGTH
 73             C4(I,J,K)  CONSTRAINT FOR P&N MUTUAL EXCLUSION
 74             C5(I)      CONSTRAINT FOR SUBREGIONS;
 75
 76          OBJ..
 77             Z =E= SUM((I,K), C(I)*X(I,K));
```

```
 78
 79          C1(K)..
 80             SUM(I,X(I,K)) =E= M(K);
 81
 82          C2(I,J,K) $ADJ(I,J)..
 83             X(I,K) - X(J,K) - P(I,J,K) + N(I,J,K) =E= 0;
 84
 85          C3(K)..
 86             SUM((I,J) $ADJ(I,J), P(I,J,K) + N(I,J,K)) =E= L(K);
 87
 88          C4(I,J,K) $ADJ(I,J)..
 89             P(I,J,K) + N(I,J,K) =L= 1;
 90
 91          C5(I)..
 92             SUM(K, X(I,K)) =L= 1;
 93
 94          MODEL SUBREGION /ALL/;
 95
 96          OPTIONS ITERLIM = 1000000, RESLIM = 1000000, WORK = 20000;
 97
 98          SOLVE SUBREGION USING MIP MAXIMIZING Z;
 99
100          OPTION MIP=ZOOM;
101
102         DISPLAY X.L,P.L,N.L;
103
104
```



```
----    102 VARIABLE  X.L          BINARY VARIABLE

            1

26       1.000
27       1.000
28       1.000
29       1.000


----    102 VARIABLE  P.L          BINARY VARIABLE

               1
```

```
26.20        1.000
26.25        1.000
26.32        1.000
27.21        1.000
27.33        1.000
28.22        1.000
28.34        1.000
29.23        1.000
29.30        1.000
29.35        1.000


----     102 VARIABLE  N.L           BINARY VARIABLE

                 1

20.26        1.000
21.27        1.000
22.28        1.000
23.29        1.000
25.26        1.000
30.29        1.000
32.26        1.000
33.27        1.000
34.28        1.000
35.29        1.000


**** FILE SUMMARY

INPUT   GS092D:[JAMRINE]S26X61.GMS;2
OUTPUT  GS092D:[JAMRINE]S26X61.LIS;2

EXECUTION TIME       =        1.940 SECONDS
```

Appendix D. *GAMS Computer Runs With Subregion Bounds*

In this appendix, three GAMS computer runs are included. These runs show the effect of adding subregion bounds contraints explained in Chapter 3. In the first run, the subregions totals (cost totals) are set to a certain value. In the second and third run, the subregions are given a range of possible cost totals.

D.1 GAMS S2-3-4, Run #1

```
GAMS 2.20 VAX VMS                                20-AUG-1992 21:09  PAGE      1
G E N E R A L   A L G E B R A I C   M O D E L I N G   S Y S T E M
C O M P I L A T I O N


  1
  2       SETS
  3          I / 1*36/
  4          K / 1*2/
  5       ALIAS(I,J)
  6       SETS
  7          ADJ(I,J)  ADJACENCY MATRIX
  8              /1.(2,7)
  9               2.(1,3,8)
 10               3.(2,4,9)
 11               4.(3,5,10)
 12               5.(4,6,11)
 13               6.(5,12)
 14               7.(1,8,13)
 15               8.(2,7,9,14)
 16               9.(3,8,10,15)
 17              10.(4,9,11,16)
 18              11.(5,10,12,17)
 19              12.(6,11,18)
 20              13.(7,14,19)
 21              14.(8,13,15,20)
 22              15.(9,14,16,21)
 23              16.(10,15,17,22)
 24              17.(11,16,18,23)
 25              18.(12,17,24)
```

```
26              19.(13,20,25)
27              20.(14,19,21,26)
28              21.(15,20,22,27)
29              22.(16,21,23,28)
30              23.(17,22,24,29)
31              24.(18,23,3C)
32              25.(19,26,31)
33              26.(20,25,27,32)
34              27.(21,26,28,33)
35              28.(22,27,29,34)
36              29.(23,28,30,35)
37              30.(24,29,36)
38              31.(25,32)
39              32.(26,31,33)
40              33.(27,32,34)
41              34.(28,33,35)
42              35.(29,34,36)
43              36.(30,35)/
44
45          PARAMETERS
46              M(K) PIXEL SIZE OF KTH SUBREGION
47                 /1 3, 2 4/
48
49              L(K) BORDER OF KTH SUBREGION
50                 /1 16, 2 20/
51
52              C(I) COST OF EACH PIXEL
53                 / 1 99,  2  99,  3  99,  4  99,  5  99,  6  99
54                   7 99,  8   1,  9   1, 10   1, 11   1, 12  99
```



```
55                  13 99, 14   2, 15   2, 16   2, 17   2, 18  99
56                  19 99, 20   3, 21   3, 22   3, 23   3, 24  99
57                  25 99, 26   4, 27   4, 28   4, 29   4, 30  99
58                  31 99, 32  99, 33  99, 34  99, 35  99, 36  99/;
59
60           VARIABLES
61              Z          TOTAL COST OF ACQUIRED PIXELS
62              X(I,K)     BINARY VARIABLE
63              P(I,J,K)   BINARY VARIABLE
64              N(I,J,K)   BINARY VARIABLE
```

```
65
66          BINARY VARIABLES X,P,N;
67
68          EQUATIONS
69             OBJ       OBJECTIVE FUNCTION
70             C1(K)     CONSTRAINT FOR SIZE OF KTH SUBREGION
71             C2(I,J,K) CONSTRAINT FOR CONTIGUITY
72             C3(K)     CONSTRAINT FOR BORDER LENGTH
73             C4(I,J,K) CONSTRAINT FOR P&N MUTUAL EXCLUSION
74             C5(I)     CONSTRAINT FOR SUBREGIONS
75             C6        CONSTRAINT FOR SUBREGION 1 COST
76             C7        CONSTRAINT FOR SUBREGION 2 COST;
77
78
79          OBJ..
80             Z =E= SUM((I,K), C(I)*X(I,K));
81
82          C1(K)..
83             SUM(I,X(I,K)) =E= M(K);
84
85          C2(I,J,K) $ADJ(I,J)..
86             X(I,K) - X(J,K) - P(I,J,K) + N(I,J,K) =E= 0;
87
88          C3(K)..
89             SUM((I,J) $ADJ(I,J), P(I,J,K) + N(I,J,K)) =E= L(K);
90
91          C4(I,J,K) $ADJ(I,J)..
92             P(I,J,K) + N(I,J,K) =L= 1;
93
94          C5(I)..
95             SUM(K, X(I,K)) =L= 1;
96
97          C6..
98             SUM(I, C(I)*X(I,"1")) =E= 3;
99
100         C7..
101            SUM(I, C(I)*X(I,"2")) =E= 16;
102
103         MODEL SUBREGION /ALL/;
104
105         OPTIONS ITERLIM = 1000000, RESLIM = 1000000, WORK = 20000;
106
107         SOLVE SUBREGION USING MIP MINIMIZING Z;
108
```

```
 109          OPTION MIP=ZOOM;
 110
 111          DISPLAY X.L,P.L,N.L;
 112
 113
```



```
----    111 VARIABLE  X.L          BINARY VARIABLE

            1          2

8        1.000
9        1.000
10       1.000
26                  1.000
27                  1.000
28                  1.000
29                  1.000


----    111 VARIABLE  P.L          BINARY VARIABLE

               1          2

8 .2        1.000
8 .7        1.000
8 .14       1.000
9 .3        1.000
9 .15       1.000
10.4        1.000
10.11       1.000
10.16       1.000
26.20                   1.000
26.25                   1.000
26.32                   1.000
27.21                   1.000
27.33                   1.000
28.22                   1.000
```

```
28.34                    1.000
29.23                    1.000
29.30                    1.000
29.35                    1.000


----     111 VARIABLE  N.L          BINARY VARIABLE

                1          2

2 .8        1.000
3 .9        1.000
4 .10       1.000
7 .8        1.000
11.10       1.000
14.8        1.000
15.9        1.000
16.10       1.000
20.26                    1.000
21.27                    1.000
22.28                    1.000
23.29                    1.000
25.26                    1.000
GAMS 2.20 VAX VMS                              20-AUG-1992 21:41  PAGE     41
G E N E R A L   A L G E B R A I C   M O D E L I N G   S Y S T E M
E X E C U T I N G

        111 VARIABLE  N.L          BINARY VARIABLE

                1          2

30.29                    1.000
32.26                    1.000
33.27                    1.000
34.28                    1.000
35.29                    1.000

**** FILE SUMMARY

INPUT   GS092D:[JAMRINE]S26X62.GMS;30
OUTPUT  GS092D:[JAMRINE]S26X62.LIS;32

EXECUTION TIME       =        2.850 SECONDS
```

D.2 GAMS S2-3-4, Run #2

```
GAMS 2.20 VAX VMS                                   23-AUG-1992 17:29  PAGE     1
G E N E R A L   A L G E B R A I C   M O D E L I N G   S Y S T E M
C O M P I L A T I O N


      SETS
         I / 1*36/
         K / 1*2/
      ALIAS(I,J)
      SETS
         ADJ(I,J)  ADJACENCY MATRIX
             /1.(2,7)
              2.(1,3,8)
              3.(2,4,9)
              4.(3,5,10)
              5.(4,6,11)
              6.(5,12)
              7.(1,8,13)
              8.(2,7,9,14)
              9.(3,8,10,15)
             10.(4,9,11,16)
             11.(5,10,12,17)
             12.(6,11,18)
             13.(7,14,19)
             14.(8,13,15,20)
             15.(9,14,16,21)
             16.(10,15,17,22)
             17.(11,16,18,23)
             18.(12,17,24)
             19.(13,20,25)
             20.(14,19,21,26)
             21.(15,20,22,27)
             22.(16,21,23,28)
             23.(17,22,24,29)
             24.(18,23,30)
             25.(19,26,31)
             26.(20,25,27,32)
             27.(21,26,28,33)
             28.(22,27,29,34)
             29.(23,28,30,35)
             30.(24,29,36)
```

```
 38            31.(25,32)
 39            32.(26,31,33)
 40            33.(27,32,34)
 41            34.(28,33,35)
 42            35.(29,34,36)
 43            36.(30,35)/
 44
 45         PARAMETERS
 46            M(K) PIXEL SIZE OF KTH SUBREGION
 47               /1 8, 2 2/
 48
 49            L(K) BORDER OF KTH SUBREGION
 50               /1 24, 2 12/
 51
 52            C(I) COST OF EACH PIXEL
 53               / 1 99,  2 99,  3 99,  4 99,  5 99,  6 99
 54                 7 99,  8  1,  9  1, 10  1, 11  1, 12 99
```



```
 55                 13 99, 14  2, 15  2, 16  2, 17  2, 18 99
 56                 19 99, 20  3, 21  3, 22  3, 23  3, 24 99
 57                 25 99, 26  4, 27  4, 28  4, 29  4, 30 99
 58                 31 99, 32 99, 33 99, 34 99, 35 99, 36 99/;
 59
 60         VARIABLES
 61            Z          TOTAL COST OF ACQUIRED PIXELS
 62            X(I,K)     BINARY VARIABLE
 63            P(I,J,K)   BINARY VARIABLE
 64            N(I,J,K)   BINARY VARIABLE
 65
 66         BINARY VARIABLES X,P,N;
 67
 68         EQUATIONS
 69            OBJ        OBJECTIVE FUNCTION
 70            C1(K)      CONSTRAINT FOR SIZE OF KTH SUBREGION
 71            C2(I,J,K)  CONSTRAINT FOR CONTIGUITY
 72            C3(K)      CONSTRAINT FOR BORDER LENGTH
 73            C4(I,J,K)  CONSTRAINT FOR P&N MUTUAL EXCLUSION
 74            C5(I)      CONSTRAINT FOR SUBREGIONS
 75            C6         CONSTRAINT FOR SUBREGION 1 COST (UPPER BOUND)
 76            C7         CONSTRAINT FOR SUBREGION 1 COST (LOWER BOUND)
 77            C8         CONSTRAINT FOR SUBREGION 2 COST (UPPER BOUND)
```

```
 78              C9        CONSTRAINT FOR SUBREGION 2 COST (LOWER BOUND);
 79
 80           OBJ..
 81              Z =E= SUM((I,K), C(I)*X(I,K));
 82
 83           C1(K)..
 84              SUM(I,X(I,K)) =E= M(K);
 85
 86           C2(I,J,K) $ADJ(I,J)..
 87              X(I,K) - X(J,K) - P(I,J,K) + N(I,J,K) =E= 0;
 88
 89           C3(K)..
 90              SUM((I,J) $ADJ(I,J), P(I,J,K) + N(I,J,K)) =E= L(K);
 91
 92           C4(I,J,K) $ADJ(I,J)..
 93              P(I,J,K) + N(I,J,K) =L= 1;
 94
 95           C5(I)..
 96              SUM(K, X(I,K)) =L= 1;
 97
 98           C6..
 99              SUM(I, C(I)*X(I,"1")) =L= 32;
100
101           C7..
102              SUM(I, C(I)*X(I,"1")) =G= 24;
103
104
105           C8..
106              SUM(I, C(I)*X(I,"2")) =L= 4;
107
108           C9..
```



```
109              SUM(I, C(I)*X(I,"2")) =G= 2;
110
111
112           MODEL SUBREGION /ALL/;
113
114           OPTIONS ITERLIM = 1000000, RESLIM = 1000000, WORK = 20000;
115
116           SOLVE SUBREGION USING MIP MINIMIZING Z;
117
```

```
 118          OPTION MIP=ZOOM;
 119
 120         DISPLAY X.L,P.L,N.L;
 121
 122
GAMS 2.20 VAX VMS                                23-AUG-1992 17:39  PAGE     40
G E N E R A L   A L G E B R A I C   M O D E L I N G   S Y S T E M
E X E C U T I N G


----     120 VARIABLE  X.L          BINARY VARIABLE

             1          2

8                    1.000
9                    1.000
14        1.000
15        1.000
20        1.000
21        1.000
22        1.000
26        1.000
27        1.000
28        1.000


----     120 VARIABLE  P.L          BINARY VARIABLE

                1          2

8 .2                     1.000
8 .7                     1.000
8 .14                    1.000
9 .3                     1.000
9 .10                    1.000
9 .15                    1.000
14.8         1.000
14.13        1.000
15.9         1.000
15.16        1.000
20.19        1.000
22.16        1.000
22.23        1.000
26.25        1.000
26.32        1.000
```

```
27.33       1.000
28.29       1.000
28.34       1.000


----    120 VARIABLE  N.L          BINARY VARIABLE

                1          2

2 .8                    1.000
3 .9                    1.000
7 .8                    1.000
8 .14       1.000
9 .15       1.000
10.9                    1.000
13.14       1.000
14.8                    1.000
15.9                    1.000
16.15       1.000
GAMS 2.20 VAX VMS                                23-AUG-1992 17:39  PAGE     41
G E N E R A L   A L G E B R A I C   M O D E L I N G   S Y S T E M
E X E C U T I N G

        120 VARIABLE  N.L          BINARY VARIABLE

                1          2

16.22       1.000
19.20       1.000
23.22       1.000
25.26       1.000
29.28       1.000
32.26       1.000
33.27       1.000
34.28       1.000

**** FILE SUMMARY

INPUT   GS092D:[JAMRINE]S26X63.GMS;4
OUTPUT  GS092D:[JAMRINE]S26X63.LIS;4

EXECUTION TIME       =        2.770 SECONDS
```

D.3 GAMS S2-3-4, Run #3

```
GAMS 2.20 VAX VMS                                  23-AUG-1992 18:10  PAGE     1
G E N E R A L   A L G E B R A I C   M O D E L I N G   S Y S T E M
C O M P I L A T I O N


      SETS
         I / 1*36/
         K / 1*2/
      ALIAS(I,J)
      SETS
         ADJ(I,J)  ADJACENCY MATRIX
             /1.(2,7)
              2.(1,3,8)
              3.(2,4,9)
              4.(3,5,10)
              5.(4,6,11)
              6.(5,12)
              7.(1,8,13)
              8.(2,7,9,14)
              9.(3,8,10,15)
             10.(4,9,11,16)
             11.(5,10,12,17)
             12.(6,11,18)
             13.(7,14,19)
             14.(8,13,15,20)
             15.(9,14,16,21)
             16.(10,15,17,22)
             17.(11,16,18,23)
             18.(12,17,24)
             19.(13,20,25)
             20.(14,19,21,26)
             21.(15,20,22,27)
             22.(16,21,23,28)
             23.(17,22,24,29)
             24.(18,23,30)
             25.(19,26,31)
             26.(20,25,27,32)
             27.(21,26,28,33)
             28.(22,27,29,34)
             29.(23,28,30,35)
             30.(24,29,36)
```

```
 38              31.(25,32)
 39              32.(26,31,33)
 40              33.(27,32,34)
 41              34.(28,33,35)
 42              35.(29,34,36)
 43              36.(30,35)/
 44
 45           PARAMETERS
 46              M(K) PIXEL SIZE OF KTH SUBREGION
 47                 /1 8, 2 2/
 48
 49              L(K) BORDER OF KTH SUBREGION
 50                 /1 24, 2 12/
 51
 52              C(I) COST OF EACH PIXEL
 53                 / 1 99,  2 99,  3 99,  4 99,  5 99,  6 99
 54                   7 99,  8  1,  9  1, 10  1, 11  1, 12 99
```



```
 55                   13 99, 14  2, 15  2, 16  2, 17  2, 18 99
 56                   19 99, 20  3, 21  3, 22  3, 23  3, 24 99
 57                   25 99, 26  4, 27  4, 28  4, 29  4, 30 99
 58                   31 99, 32 99, 33 99, 34 99, 35 99, 36 99/;
 59
 60           VARIABLES
 61              Z          TOTAL COST OF ACQUIRED PIXELS
 62              X(I,K)     BINARY VARIABLE
 63              P(I,J,K)   BINARY VARIABLE
 64              N(I,J,K)   BINARY VARIABLE
 65
 66           BINARY VARIABLES X,P,N;
 67
 68           EQUATIONS
 69              OBJ        OBJECTIVE FUNCTION
 70              C1(K)      CONSTRAINT FOR SIZE OF KTH SUBREGION
 71              C2(I,J,K)  CONSTRAINT FOR CONTIGUITY
 72              C3(K)      CONSTRAINT FOR BORDER LENGTH
 73              C4(I,J,K)  CONSTRAINT FOR P&N MUTUAL EXCLUSION
 74              C5(I)      CONSTRAINT FOR SUBREGIONS
 75              C6         CONSTRAINT FOR SUBREGION 1 COST (UPPER BOUND)
 76              C7         CONSTRAINT FOR SUBREGION 1 COST (LOWER BOUND)
 77              C8         CONSTRAINT FOR SUBREGION 2 COST (UPPER BOUND)
```

```
 78              C9        CONSTRAINT FOR SUBREGION 2 COST (LOWER BOUND);
 79
 80           OBJ..
 81              Z =E= SUM((I,K), C(I)*X(I,K));
 82
 83           C1(K)..
 84              SUM(I,X(I,K)) =E= M(K);
 85
 86           C2(I,J,K) $ADJ(I,J)..
 87              X(I,K) - X(J,K) - P(I,J,K) + N(I,J,K) =E= 0;
 88
 89           C3(K)..
 90              SUM((I,J) $ADJ(I,J), P(I,J,K) + N(I,J,K)) =E= L(K);
 91
 92           C4(I,J,K) $ADJ(I,J)..
 93              P(I,J,K) + N(I,J,K) =L= 1;
 94
 95           C5(I)..
 96              SUM(K, X(I,K)) =L= 1;
 97
 98           C6..
 99              SUM(I, C(I)*X(I,"1")) =L= 32;
100
101           C7..
102              SUM(I, C(I)*X(I,"1")) =G= 26;
103
104
105           C8..
106              SUM(I, C(I)*X(I,"2")) =L= 4;
107
108           C9..
```



```
109              SUM(I, C(I)*X(I,"2")) =G= 2;
110
111
112           MODEL SUBREGION /ALL/;
113
114           OPTIONS ITERLIM = 1000000, RESLIM = 1000000, WORK = 20000;
115
116           SOLVE SUBREGION USING MIP MINIMIZING Z;
117
```

```
 118          OPTION MIP=ZOOM;
 119
 120         DISPLAY X.L,P.L,N.L;
 121
 122
GAMS 2.20 VAX VMS                                  23-AUG-1992 18:21  PAGE     40
G E N E R A L   A L G E B R A I C   M O D E L I N G   S Y S T E M
E X E C U T I N G


----     120 VARIABLE  X.L          BINARY VARIABLE

             1           2

8                     1.000
9                     1.000
20        1.000
21        1.000
22        1.000
23        1.000
26        1.000
27        1.000
28        1.000
29        1.000


----     120 VARIABLE  P.L          BINARY VARIABLE

                1           2

8 .2                    1.000
8 .7                    1.000
8 .14                   1.000
9 .3                    1.000
9 .10                   1.000
9 .15                   1.000
20.14        1.000
20.19        1.000
21.15        1.000
22.16        1.000
23.17        1.000
23.24        1.000
26.25        1.000
26.32        1.000
27.33        1.000
```

```
28.34       1.000
29.30       1.000
29.35       1.000


----    120 VARIABLE  N.L          BINARY VARIABLE

                1          2

2 .8                    1.000
3 .9                    1.000
7 .8                    1.000
10.9                    1.000
14.8                    1.000
14.20       1.000
15.9                    1.000
15.21       1.000
16.22       1.000
17.23       1.000
GAMS 2.20 VAX VMS                                23-AUG-1992 18:21  PAGE     41
G E N E R A L   A L G E B R A I C   M O D E L I N G   S Y S T E M
E X E C U T I N G


        120 VARIABLE  N.L          BINARY VARIABLE

                1          2

19.20       1.000
24.23       1.000
25.26       1.000
30.29       1.000
32.26       1.000
33.27       1.000
34.28       1.000
35.29       1.000


**** FILE SUMMARY

INPUT   GS092D:[JAMRINE]S26X63B.GMS;2
OUTPUT  GS092D:[JAMRINE]S26X63B.LIS;2

EXECUTION TIME       =        2.860 SECONDS
```

Appendix E. *SPOT Subimage Gray Values*

In this appendix are displayed the gray values of the subimage by channel. Each 48 × 18 subimage is surrounded by a border of 999s. These numbers are not part of the actual subimage but rather assist the model in accounting for border lengths. Since the SPOT data is eight-bit, the allowable range for the gray value is 0 – 255.

E.1 Channel 1 Gray Values

999	999	999	999	999	999	999	999	999	999	999	999	999	999	999	999	999	999	999	999
999	224	222	214	213	215	219	221	217	198	211	204	157	174	145	139	168	90	58	999
999	168	192	204	210	213	210	198	214	211	204	211	174	186	192	163	163	82	107	999
999	121	139	139	139	174	157	133	204	168	121	174	163	107	127	168	145	82	157	999
999	192	168	163	192	198	174	174	211	174	121	186	127	90	145	192	121	74	180	999
999	157	145	139	151	163	139	174	210	168	139	157	139	186	204	163	107	82	139	999
999	25	99	133	99	133	157	180	204	180	168	157	211	214	151	99	99	74	90	999
999	0	33	145	151	151	157	151	213	151	151	192	204	192	74	145	157	58	90	999
999	0	0	74	133	115	133	174	210	115	145	192	174	139	121	192	157	99	145	999
999	0	0	16	133	139	151	180	192	192	192	168	145	163	186	180	145	82	145	999
999	0	0	0	139	168	157	186	192	215	168	115	151	192	180	139	115	107	157	999
999	0	0	0	121	145	163	180	192	210	115	151	174	210	180	133	115	82	99	999
999	0	0	6	121	157	168	192	210	218	174	145	192	204	157	168	107	107	121	999
999	0	0	22	157	157	174	213	215	216	211	163	127	127	127	168	133	127	163	999
999	0	0	133	168	115	99	174	174	133	192	192	139	82	127	174	127	139	180	999
999	0	107	180	139	139	139	198	204	127	163	210	198	151	115	163	107	115	151	999
999	180	198	107	107	145	174	186	214	180	151	186	213	215	180	157	90	74	121	999
999	214	107	58	151	151	145	168	211	163	157	127	145	211	198	168	99	74	139	999
999	157	115	127	151	139	145	186	192	163	145	145	127	157	204	180	107	107	168	999
999	139	151	133	157	180	180	163	192	210	180	163	168	163	168	192	127	145	163	999
999	204	186	174	192	139	107	115	133	151	151	163	204	180	121	180	145	157	198	999
999	204	212	180	127	99	66	82	25	58	82	82	133	151	115	163	139	127	198	999
999	99	174	127	99	145	133	127	58	33	50	90	115	151	127	174	107	90	151	999
999	127	99	58	133	180	139	151	157	121	115	151	168	198	212	214	174	168	204	999
999	163	186	145	127	151	192	213	215	210	157	168	198	210	217	221	210	198	214	999
999	180	218	145	66	204	215	139	127	180	180	133	115	74	151	212	133	151	180	999
999	192	215	174	192	216	145	0	0	58	204	174	127	58	145	220	127	115	157	999
999	22	168	214	224	186	9	0	0	0	213	225	227	223	221	225	157	107	174	999
999	0	90	168	215	145	13	0	0	0	192	215	215	192	215	223	151	127	192	999

999	0	90	127	186	133	19	9	0	0	163	145	33	6	145	215	133	139	174	999
999	0	33	145	180	163	0	9	0	0	192	139	127	107	151	215	139	139	192	999
999	16	151	215	216	180	0	3	0	0	212	219	225	223	221	218	90	99	174	999
999	186	215	168	192	213	82	6	0	58	215	215	212	198	215	219	74	25	151	999
999	192	133	0	22	198	180	25	25	151	139	41	25	74	168	216	115	74	145	999
999	204	107	74	74	16	127	99	82	50	25	25	82	168	168	211	133	163	204	999
999	198	121	204	174	22	66	74	25	16	82	157	168	151	180	168	74	163	213	999
999	115	139	186	204	133	133	145	121	121	163	174	139	121	151	139	74	107	163	999
999	127	145	115	157	139	192	204	192	168	168	145	145	121	157	168	127	127	151	999
999	145	157	151	168	168	186	186	186	174	127	99	157	192	157	133	121	115	121	999
999	163	157	145	133	133	168	198	198	186	145	82	163	174	139	99	121	115	121	999
999	163	157	145	163	168	168	186	163	163	121	50	151	180	99	74	127	99	139	999
999	168	139	121	192	210	139	121	121	157	174	66	66	66	41	127	157	99	121	999
999	157	145	174	192	127	58	41	133	212	214	133	90	90	58	145	186	90	66	999
999	151	186	145	82	41	107	151	163	157	157	151	163	139	90	127	133	41	22	999
999	145	107	41	74	139	163	151	163	163	121	127	163	163	127	115	90	127	127	999
999	99	82	127	174	157	121	107	145	163	133	121	82	74	115	115	99	115	127	999
999	115	168	157	127	145	151	174	127	121	107	41	25	25	16	6	3	9	19	999
999	168	168	127	139	180	174	157	74	50	13	0	0	0	0	0	0	0	0	999
999	145	145	180	186	139	99	22	0	0	0	0	0	0	0	0	0	0	0	999
999	999	999	999	999	999	999	999	999	999	999	999	999	999	999	999	999	999	999	999

E.2 Channel 2 Gray Values

999	999	999	999	999	999	999	999	999	999	999	999	999	999	999	999	999	999	999	999
999	208	208	183	183	177	194	206	206	188	194	155	111	127	116	122	155	116	111	999
999	105	116	127	127	116	116	138	161	149	138	111	39	67	96	116	144	111	138	999
999	20	20	39	67	58	29	10	96	96	29	39	20	9	29	67	96	77	149	999
999	111	77	77	86	29	39	67	105	58	9	20	10	9	39	86	67	77	177	999
999	155	122	86	29	10	9	67	86	58	20	29	9	58	96	77	67	86	161	999
999	155	161	144	20	9	48	96	77	39	39	67	105	116	77	58	67	77	133	999
999	96	161	177	116	67	86	105	138	39	67	116	122	105	9	77	111	58	144	999
999	48	122	199	183	105	105	122	138	67	116	138	86	39	10	86	127	111	149	999
999	39	7	138	206	155	155	194	206	194	138	77	58	39	48	77	96	86	138	999
999	20	7	77	210	210	205	211	214	213	133	20	67	67	48	48	67	116	133	999
999	7	48	48	209	210	207	212	214	211	86	67	58	39	39	58	48	96	105	999
999	39	29	111	210	212	208	216	216	213	133	67	58	29	10	58	67	122	96	999
999	48	10	155	212	205	166	208	208	194	133	96	48	8	20	77	96	116	111	399
999	6	77	206	194	86	86	133	127	67	105	116	48	6	9	58	86	116	111	999
999	48	172	205	122	58	67	116	111	29	48	96	67	39	20	58	77	122	105	999
999	172	183	77	48	58	58	67	96	67	29	39	96	111	77	58	77	105	77	999
999	155	86	20	48	39	20	58	96	58	29	9	20	96	105	96	96	105	86	999
999	67	29	39	39	10	20	48	86	67	58	20	9	58	86	122	116	105	96	999
999	48	48	10	29	29	39	58	86	96	58	20	29	48	67	116	133	116	96	999
999	105	48	29	29	10	10	29	77	86	77	67	86	67	10	96	122	133	122	999
999	105	133	105	39	29	10	58	48	48	29	48	58	58	8	86	111	105	127	999
999	96	138	105	67	96	67	58	48	29	48	48	58	67	48	111	111	105	122	999
999	77	67	39	96	122	58	48	48	29	39	48	67	58	116	172	188	183	183	999
999	86	105	48	10	58	111	133	161	127	58	10	29	29	111	194	205	199	208	999
999	116	166	77	7	96	183	166	188	194	111	9	7	6	20	133	122	138	138	999
999	155	206	166	138	177	161	105	116	194	199	96	20	6	67	188	122	105	58	999
999	105	199	210	212	177	67	67	48	138	206	155	133	116	155	208	155	105	105	999
999	5	155	183	206	133	58	116	29	127	207	133	67	7	67	183	133	122	105	999
999	9	188	149	172	122	58	105	105	144	194	77	2	0	7	122	86	122	105	999
999	5	133	194	183	155	7	96	77	116	194	67	10	8	39	149	116	133	105	999
999	39	172	209	209	188	5	67	39	111	207	183	161	144	183	206	122	122	86	999
999	111	172	144	166	199	116	58	29	133	206	166	116	77	138	199	105	67	67	999
999	138	105	3	8	155	161	58	77	155	133	20	5	6	58	149	96	58	96	999
999	133	86	58	58	10	105	96	86	48	20	9	48	96	105	155	138	155	161	999
999	96	58	116	105	10	39	48	29	10	39	86	111	105	138	138	105	172	206	999
999	9	9	20	86	39	48	67	67	86	96	77	29	10	96	111	105	122	166	999
999	7	7	5	9	10	77	86	96	77	58	9	9	9	67	111	122	122	122	999
999	9	7	8	9	9	39	67	58	39	9	6	39	86	77	86	96	105	86	999
999	6	8	7	8	8	10	58	48	39	10	9	67	111	96	105	96	116	96	999
999	7	7	7	8	8	10	39	29	29	10	8	86	122	96	96	111	116	133	999

999	10	7	6	10	39	20	10	20	39	58	10	20	48	67	127	138	105	127	999
999	58	20	10	29	29	8	8	58	116	111	39	10	48	39	122	161	96	77	999
999	77	105	39	9	10	29	77	58	58	58	86	96	77	39	77	105	29	29	999
999	122	86	9	8	67	86	67	48	67	67	116	144	144	116	96	105	116	122	999
999	96	58	67	86	67	67	58	77	122	127	155	166	172	194	172	161	155	161	999
999	58	67	58	67	77	111	155	138	149	155	138	149	155	138	138	133	127	166	999
999	105	86	58	116	138	155	161	144	144	105	29	20	39	9	10	58	48	48	999
999	122	111	144	183	149	138	77	48	67	9	29	58	96	10	48	58	48	29	999
999	999	999	999	999	999	999	999	999	999	999	999	999	999	999	999	999	999	999	999

E.3 Channel 3 Gray Values

999	999	999	999	999	999	999	999	999	999	999	999	999	999	999	999	999	999	999	999
999	207	206	201	200	203	202	205	204	194	203	183	122	133	128	139	178	133	122	999
999	106	117	139	139	139	117	128	161	167	150	128	55	82	100	117	161	122	139	999
999	10	10	19	55	46	28	19	100	91	19	28	28	9	19	82	111	91	167	999
999	100	82	82	64	64	46	73	117	55	9	37	10	8	64	111	91	82	194	999
999	167	128	82	28	19	28	73	106	64	19	28	19	64	111	82	100	106	178	999
999	167	189	139	9	10	55	100	82	46	55	64	111	128	82	55	100	91	139	999
999	117	201	201	128	64	91	122	133	46	82	133	128	106	8	82	117	82	133	999
999	100	167	205	201	91	91	133	144	73	122	150	111	46	9	100	117	117	161	999
999	64	8	161	206	167	172	201	204	200	150	106	55	55	55	82	111	106	133	999
999	73	8	91	210	207	203	209	212	208	122	28	64	73	55	73	73	117	139	999
999	37	91	91	210	208	205	209	209	206	82	91	91	55	46	55	55	106	91	999
999	82	55	122	209	207	204	211	211	208	133	91	73	37	28	73	73	122	100	999
999	122	46	172	207	183	156	202	203	200	144	106	55	8	19	100	106	133	122	999
999	10	117	202	183	82	55	122	122	64	117	122	64	8	10	82	100	139	122	999
999	73	201	203	111	37	37	100	111	19	46	91	100	46	37	73	73	122	117	999
999	201	201	73	46	46	46	73	117	64	28	55	122	144	91	73	106	111	91	999
999	183	91	8	46	37	19	55	111	46	37	10	19	117	111	100	106	100	100	999
999	55	19	9	19	9	19	46	91	73	46	10	10	46	106	111	117	117	111	999
999	46	37	10	28	28	55	64	91	106	82	64	64	55	55	122	139	139	111	999
999	100	55	46	64	37	19	37	64	91	82	64	106	82	19	111	128	139	144	999
999	111	139	106	46	28	19	73	46	64	55	64	73	73	10	100	106	122	150	999
999	73	133	106	91	100	73	73	28	37	46	73	64	82	64	122	111	111	150	999
999	91	64	55	111	117	55	55	55	28	37	46	73	91	128	183	194	201	202	999
999	106	106	64	19	46	117	161	183	133	46	28	37	64	117	200	201	203	205	999
999	117	172	82	6	111	202	201	202	201	111	9	8	6	28	144	122	133	144	999
999	172	205	167	156	201	172	106	133	204	203	106	37	8	82	189	117	100	73	999
999	111	203	208	212	201	73	106	73	183	209	178	156	133	172	207	156	106	106	999
999	6	183	201	205	150	82	156	73	161	209	156	91	8	73	189	133	122	128	999
999	10	201	150	161	128	73	128	133	161	201	82	2	0	8	117	100	139	117	999
999	5	122	178	183	183	19	133	111	122	203	82	28	8	37	167	122	150	128	999
999	46	172	204	205	201	19	106	91	117	206	201	183	167	183	205	139	133	117	999
999	128	183	139	178	205	161	82	55	161	206	189	133	100	150	201	111	82	111	999
999	156	91	2	7	156	194	100	111	194	150	19	5	7	73	167	91	82	106	999
999	150	82	46	46	10	128	117	100	55	37	8	46	111	100	161	139	172	178	999
999	111	55	111	106	8	37	55	28	10	64	117	111	111	144	156	117	189	205	999
999	8	9	37	82	46	46	64	64	100	106	100	37	19	82	117	111	133	183	999
999	8	9	6	8	37	100	111	106	82	55	28	28	10	55	122	117	139	139	999
999	28	9	9	19	28	64	55	55	37	10	8	37	106	91	106	100	111	100	999
999	10	9	8	8	7	28	55	73	64	19	8	91	117	106	100	111	128	111	999
999	8	8	8	8	10	37	55	55	46	10	8	91	139	111	111	128	133	144	999

999	28	9	8	37	46	28	28	37	64	82	19	28	55	82	144	161	122	133	999
999	73	46	37	55	19	9	9	82	144	139	64	28	55	46	133	172	111	100	999
999	111	122	55	9	19	55	100	82	73	82	106	111	100	37	91	100	28	10	999
999	133	91	9	28	100	100	73	82	82	73	122	156	156	122	111	82	117	128	999
999	100	82	91	100	82	82	64	82	122	133	178	183	194	205	201	172	161	178	999
999	91	117	91	55	91	111	172	156	172	189	167	189	178	183	167	150	178	200	999
999	122	117	100	133	156	167	167	161	156	122	64	64	64	10	28	73	64	73	999
999	128	128	156	194	161	156	91	73	73	19	46	82	111	28	100	91	82	64	999
999	999	999	999	999	999	999	999	999	999	999	999	999	999	999	999	999	999	999	999

Appendix F. *SPOT Subimage Gray Value Count and Program*

In this appendix are contained the gray value pixel counts by channel and the FORTRAN program created to generate this list. The FORTRAN program created this list by reading the pixel values in Appendix E.

F.1 Channel 1 Gray Value Count

```
CHANNEL 1 PIXEL COUNT BY GRAY VALUE
-----------------------------------

FOR GRAY VALUE =     0.       THE COUNT IS   58
FOR GRAY VALUE =     3.00000 THE COUNT IS   2
FOR GRAY VALUE =     6.00000 THE COUNT IS   4
FOR GRAY VALUE =     9.00000 THE COUNT IS   4
FOR GRAY VALUE =     13.0000 THE COUNT IS   2
FOR GRAY VALUE =     16.0000 THE COUNT IS   5
FOR GRAY VALUE =     19.0000 THE COUNT IS   2
FOR GRAY VALUE =     22.0000 THE COUNT IS   6
FOR GRAY VALUE =     25.0000 THE COUNT IS   11
FOR GRAY VALUE =     33.0000 THE COUNT IS   4
FOR GRAY VALUE =     41.0000 THE COUNT IS   7
FOR GRAY VALUE =     50.0000 THE COUNT IS   4
FOR GRAY VALUE =     58.0000 THE COUNT IS   11
FOR GRAY VALUE =     66.0000 THE COUNT IS   7
FOR GRAY VALUE =     74.0000 THE COUNT IS   19
FOR GRAY VALUE =     82.0000 THE COUNT IS   17
FOR GRAY VALUE =     90.0000 THE COUNT IS   15
FOR GRAY VALUE =     99.0000 THE COUNT IS   22
FOR GRAY VALUE =     107.000 THE COUNT IS   23
FOR GRAY VALUE =     115.000 THE COUNT IS   26
FOR GRAY VALUE =     121.000 THE COUNT IS   29
FOR GRAY VALUE =     127.000 THE COUNT IS   44
FOR GRAY VALUE =     133.000 THE COUNT IS   30
FOR GRAY VALUE =     139.000 THE COUNT IS   42
FOR GRAY VALUE =     145.000 THE COUNT IS   47
FOR GRAY VALUE =     151.000 THE COUNT IS   42
```

```
FOR GRAY VALUE =     157.000 THE COUNT IS   42
FOR GRAY VALUE =     163.000 THE COUNT IS   41
FOR GRAY VALUE =     168.000 THE COUNT IS   42
FOR GRAY VALUE =     174.000 THE COUNT IS   35
FOR GRAY VALUE =     180.000 THE COUNT IS   32
FOR GRAY VALUE =     186.000 THE COUNT IS   22
FOR GRAY VALUE =     192.000 THE COUNT IS   38
FOR GRAY VALUE =     198.000 THE COUNT IS   17
FOR GRAY VALUE =     204.000 THE COUNT IS   21
FOR GRAY VALUE =     210.000 THE COUNT IS   13
FOR GRAY VALUE =     211.000 THE COUNT IS   9
FOR GRAY VALUE =     212.000 THE COUNT IS   6
FOR GRAY VALUE =     213.000 THE COUNT IS   9
FOR GRAY VALUE =     214.000 THE COUNT IS   9
FOR GRAY VALUE =     215.000 THE COUNT IS   18
FOR GRAY VALUE =     216.000 THE COUNT IS   4
FOR GRAY VALUE =     217.000 THE COUNT IS   2
FOR GRAY VALUE =     218.000 THE COUNT IS   3
FOR GRAY VALUE =     219.000 THE COUNT IS   3
FOR GRAY VALUE =     220.000 THE COUNT IS   1
FOR GRAY VALUE =     221.000 THE COUNT IS   4
FOR GRAY VALUE =     222.000 THE COUNT IS   1
FOR GRAY VALUE =     223.000 THE COUNT IS   3
FOR GRAY VALUE =     224.000 THE COUNT IS   2
FOR GRAY VALUE =     225.000 THE COUNT IS   3
FOR GRAY VALUE =     227.000 THE COUNT IS   1
```

F.2 Channel 2 Gray Value Count

```
CHANNEL 2 PIXEL COUNT BY GRAY VALUE
-----------------------------------

FOR GRAY VALUE =     0.       THE COUNT IS   1
FOR GRAY VALUE =     2.00000 THE COUNT IS   1
FOR GRAY VALUE =     3.00000 THE COUNT IS   1
FOR GRAY VALUE =     5.00000 THE COUNT IS   5
FOR GRAY VALUE =     6.00000 THE COUNT IS   8
FOR GRAY VALUE =     7.00000 THE COUNT IS   16
FOR GRAY VALUE =     8.00000 THE COUNT IS   14
FOR GRAY VALUE =     9.00000 THE COUNT IS   28
FOR GRAY VALUE =    10.00000 THE COUNT IS   33
FOR GRAY VALUE =     20.0000 THE COUNT IS   26
FOR GRAY VALUE =     29.0000 THE COUNT IS   38
FOR GRAY VALUE =     39.0000 THE COUNT IS   41
FOR GRAY VALUE =     48.0000 THE COUNT IS   45
FOR GRAY VALUE =     58.0000 THE COUNT IS   59
FOR GRAY VALUE =     67.0000 THE COUNT IS   58
FOR GRAY VALUE =     77.0000 THE COUNT IS   39
FOR GRAY VALUE =     86.0000 THE COUNT IS   39
FOR GRAY VALUE =     96.0000 THE COUNT IS   48
FOR GRAY VALUE =    105.0000 THE COUNT IS   47
FOR GRAY VALUE =     111.000 THE COUNT IS   28
FOR GRAY VALUE =     116.000 THE COUNT IS   35
FOR GRAY VALUE =     122.000 THE COUNT IS   31
FOR GRAY VALUE =     127.000 THE COUNT IS   12
FOR GRAY VALUE =     133.000 THE COUNT IS   22
FOR GRAY VALUE =     138.000 THE COUNT IS   26
FOR GRAY VALUE =     144.000 THE COUNT IS   11
FOR GRAY VALUE =     149.000 THE COUNT IS   9
FOR GRAY VALUE =     155.000 THE COUNT IS   24
FOR GRAY VALUE =     161.000 THE COUNT IS   13
FOR GRAY VALUE =     166.000 THE COUNT IS   9
FOR GRAY VALUE =     172.000 THE COUNT IS   9
FOR GRAY VALUE =     177.000 THE COUNT IS   5
FOR GRAY VALUE =     183.000 THE COUNT IS   13
FOR GRAY VALUE =     188.000 THE COUNT IS   6
FOR GRAY VALUE =     194.000 THE COUNT IS   13
FOR GRAY VALUE =     199.000 THE COUNT IS   6
FOR GRAY VALUE =     205.000 THE COUNT IS   4
```

```
FOR GRAY VALUE =     206.000 THE COUNT IS   11
FOR GRAY VALUE =     207.000 THE COUNT IS   3
FOR GRAY VALUE =     208.000 THE COUNT IS   7
FOR GRAY VALUE =     209.000 THE COUNT IS   3
FOR GRAY VALUE =     210.000 THE COUNT IS   5
FOR GRAY VALUE =     211.000 THE COUNT IS   2
FOR GRAY VALUE =     212.000 THE COUNT IS   4
FOR GRAY VALUE =     213.000 THE COUNT IS   2
FOR GRAY VALUE =     214.000 THE COUNT IS   2
FOR GRAY VALUE =     216.000 THE COUNT IS   2
```

F.3 Channel 3 Gray Value Count

```
CHANNEL 3 PIXEL COUNT BY GRAY VALUE
-----------------------------------

FOR GRAY VALUE =     0.       THE COUNT IS   1
FOR GRAY VALUE =     2.00000 THE COUNT IS   2
FOR GRAY VALUE =     5.00000 THE COUNT IS   2
FOR GRAY VALUE =     6.00000 THE COUNT IS   4
FOR GRAY VALUE =     7.00000 THE COUNT IS   3
FOR GRAY VALUE =     8.00000 THE COUNT IS   27
FOR GRAY VALUE =     9.00000 THE COUNT IS   17
FOR GRAY VALUE =    10.00000 THE COUNT IS   21
FOR GRAY VALUE =     19.0000 THE COUNT IS   28
FOR GRAY VALUE =     28.0000 THE COUNT IS   32
FOR GRAY VALUE =     37.0000 THE COUNT IS   28
FOR GRAY VALUE =     46.0000 THE COUNT IS   36
FOR GRAY VALUE =     55.0000 THE COUNT IS   45
FOR GRAY VALUE =     64.0000 THE COUNT IS   38
FOR GRAY VALUE =     73.0000 THE COUNT IS   41
FOR GRAY VALUE =     82.0000 THE COUNT IS   47
FOR GRAY VALUE =     91.0000 THE COUNT IS   39
FOR GRAY VALUE =    100.0000 THE COUNT IS   41
FOR GRAY VALUE =     106.000 THE COUNT IS   33
FOR GRAY VALUE =     111.000 THE COUNT IS   44
FOR GRAY VALUE =     117.000 THE COUNT IS   34
FOR GRAY VALUE =     122.000 THE COUNT IS   32
FOR GRAY VALUE =     128.000 THE COUNT IS   20
FOR GRAY VALUE =     133.000 THE COUNT IS   26
FOR GRAY VALUE =     139.000 THE COUNT IS   21
FOR GRAY VALUE =     144.000 THE COUNT IS   10
FOR GRAY VALUE =     150.000 THE COUNT IS   12
FOR GRAY VALUE =     156.000 THE COUNT IS   16
FOR GRAY VALUE =     161.000 THE COUNT IS   15
FOR GRAY VALUE =     167.000 THE COUNT IS   14
FOR GRAY VALUE =     172.000 THE COUNT IS   12
FOR GRAY VALUE =     178.000 THE COUNT IS   10
FOR GRAY VALUE =     183.000 THE COUNT IS   16
FOR GRAY VALUE =     189.000 THE COUNT IS   7
FOR GRAY VALUE =     194.000 THE COUNT IS   7
FOR GRAY VALUE =     200.000 THE COUNT IS   5
FOR GRAY VALUE =     201.000 THE COUNT IS   21
```

```
FOR GRAY VALUE =     202.000 THE COUNT IS   6
FOR GRAY VALUE =     203.000 THE COUNT IS   9
FOR GRAY VALUE =     204.000 THE COUNT IS   5
FOR GRAY VALUE =     205.000 THE COUNT IS   11
FOR GRAY VALUE =     206.000 THE COUNT IS   5
FOR GRAY VALUE =     207.000 THE COUNT IS   5
FOR GRAY VALUE =     208.000 THE COUNT IS   4
FOR GRAY VALUE =     209.000 THE COUNT IS   6
FOR GRAY VALUE =     210.000 THE COUNT IS   2
FOR GRAY VALUE =     211.000 THE COUNT IS   2
FOR GRAY VALUE =     212.000 THE COUNT IS   2
```

F.4 FORTRAN Program: Gray Value Count

```
*************************************************************************
*                                                                       *
*                      GRAY VALUE COUNT PROGRAM                         *
*                                                                       *
* AUTHOR: CAPT JOHN M. AMRINE                                           *
* CREATED: NOV 1992                                                     *
*                                                                       *
*************************************************************************
* THIS PROGRAM READS THE GRAY VALUES OF THE THREE IMAGES (CHANNELS 1-3)  *
* AND OUTPUTS THE GRAY VALUE COUNT BY CHANNEL.  CHANNEL 1 IS CONTAINED IN *
* ROWS 1-50, CHANNEL 2 IN ROWS 51-100 AND CHANNEL 3 IN ROWS 101-150.  EACH *
* CHANNEL HAS A BORDER COMPOSED OF 999s.  THIS IS NOT A REQUIREMENT FOR THE*
* THE SUCCESSFUL RUNNING OF THIS PROGRAM BUT RATHER A REQUIREMENT FOR    *
* ADDITIONAL IMAGE PROCESSING THAT WILL OCCUR OUTSIDE OF THIS PROGRAM. THE *
* 999s ARE NOT COUNTED IN THIS PROGRAM SINCE THE ARE NOT PART OF THE ACTUAL*
* IMAGE.  THE DATA OF THE THREE CHANNELS ARE READ INTO THE PROGRAM VIA THE *
* UNIX COMMAND "a.out < filename.ext" (QUOTES ARE NOT INCLUDED IN THE     *
* COMMAND).  THIS PROGRAM CAN BE EASILY EXTENDED TO DATA SETS COMPOSED OF *
* MORE OR FEWER THAN THREE CHANNELS.  THIS PROGRAM ASSUMES EIGHT BIT DATA. *
*                                                                       *
*************************************************************************
* VARIABLES:                                                            *
* COLS:   NUMBER OF COLUMNS IN DATA SET                                 *
* ROWS:   NUMBER OF ROWS IN DATA SET                                    *
* I:      COUNTER FOR ROWS                                              *
* J:      COUNTER FOR COLUMNS                                           *
* COUNT:  COUNTER FOR NUMBER OF A PARTICULAR GRAY VALUE                 *
* GRAY:  GRAY VALUE                                                     *
* MAX:    MAXIMUM GRAY VALUE ALLOWED                                    *
* ARRAY1: 150 X 20 ARRAY CONTAINING THE GRAY VALUES BY CHANNEL          *
*                                                                       *
*************************************************************************

      INTEGER COLS, ROWS, I, J, COUNT
      REAL GRAY, MAX
      REAL ARRAY1(150,20)

* Data will be read from external data file

      READ *, ROWS, COLS

* Fill the real array
```

```
      DO 10 I = 1, ROWS
 READ *, (ARRAY1 (I, J), J = 1, COLS)

10    CONTINUE

* Subroutine to get count of individual pixel values in Channel 1

* Print title for Channel 1
        PRINT *
        PRINT *, '          CHANNEL 1 PIXEL COUNT BY GRAY VALUE'
        PRINT *, '          ----------------------------------'
        PRINT *


* INITIALIZE MAX, GRAY, AND COUNT

      MAX = 255.0
      GRAY = 0.00
130   COUNT = 0
      DO 110 I = 2, 49
         DO 120 J = 2, COLS - 1
            IF (ARRAY1(I, J) .LT. GRAY+.0001 .AND. ARRAY1(I,J)
     +                       .GT. GRAY-.0001)  THEN
               COUNT = COUNT + 1
            ENDIF
120      CONTINUE
110   CONTINUE

      IF (COUNT .GT. 0)
     + PRINT *, '          FOR GRAY VALUE = ', GRAY,' THE COUNT IS ',
     + COUNT

      GRAY = GRAY + 1.0

      IF (GRAY .LE. MAX) GOTO 130

* Print title for Channel 2
        PRINT *
        PRINT *, '          CHANNEL 2 PIXEL COUNT BY GRAY VALUE'
        PRINT *, '          ----------------------------------'
        PRINT *

* Subroutine to get count of individual pixel values in Channel 2
```

```
* INITIALIZE MAX, GRAY, AND COUNT

      MAX = 255.0
      GRAY = 0.00
230   COUNT = 0
      DO 210 I = 52, 99
         DO 220 J = 2, COLS - 1
            IF (ARRAY1(I, J) .LT. GRAY+.0001 .AND. ARRAY1(I,J)
     +                       .GT. GRAY-.0001)  THEN
                COUNT = COUNT + 1
            ENDIF
220      CONTINUE
210   CONTINUE

      IF (COUNT .GT. 0)
     + PRINT *, '          FOR GRAY VALUE = ', GRAY,' THE COUNT IS ',
     + COUNT

      GRAY = GRAY + 1.0

      IF (GRAY .LE. MAX) GOTO 230

* Print title for Channel 3
        PRINT *
        PRINT *, '          CHANNEL 3 PIXEL COUNT BY GRAY VALUE'
        PRINT *, '          ----------------------------------'
        PRINT *

* Subroutine to get count of individual pixel values in channel 3

* INITIALIZE MAX, GRAY, AND COUNT

      MAX = 255.0
      GRAY = 0.00
330   COUNT = 0
      DO 310 I = 102, 149
         DO 320 J = 2, COLS - 1
            IF (ARRAY1(I, J) .LT. GRAY+.0001 .AND. ARRAY1(I,J)
     +                       .GT. GRAY-.0001)  THEN
                COUNT = COUNT + 1
            ENDIF
320      CONTINUE
310   CONTINUE

      IF (COUNT .GT. 0)
```

```
+ PRINT *, '          FOR GRAY VALUE = ', GRAY,' THE COUNT IS ',
+ COUNT

 GRAY = GRAY + 1.0

 IF (GRAY .LE. MAX) GOTO 330


 END
```

Appendix G. *SPOT Subimage Water Regions and Program*

In this appendix are contained the water regions based on spectral vaulues alone. Each channel of data has been filtered using the spectral range for water identified in Chapter 4. In addition to identifying the bodies of water, other nonwater areas were characterized as water. This was particularily evident in Channels 2 and 3. The reason for this labeling of nonwater areas as water is that the other areas were spectrally similar to water. In these character representations, the pixels within the water spectral range have been labelled with a W. The 999 border has been labeled with an X and the other gray values with a space.

G.1 Channel 1 Water Regions

CHARACTER REPRESENTATION OF CHANNEL 1 WATER

```
XXXXXXXXXXXXXXXXXXXX
X                  X
X                  X
X                  X
X                  X
X                  X
X                  X
XW                 X
XWW                X
XWWW               X
XWWW               X
XWWW               X
XWWW               X
XWWW               X
XWW                X
XW                 X
X                  X
X                  X
X                  X
X                  X
X                  X
X                  X
X                  X
X                  X
X                  X
X                  X
X                  X
X      WW          X
XW    WWWW         X
XW    WWWW         X
XW    WWWW   W     X
XW    WWWW         X
XW    WWWW         X
X      WW          X
X  WW              X
X    W             X
X    W   W         X
X                  X
X                  X
X                  X
X                  X
X                  X
X                  X
X                  X
X                 WX
X                  X
X                  X
X             WWWWWX
X         WWWWWWWWWX
X      WWWWWWWWWWWWX
XXXXXXXXXXXXXXXXXXXX
```

G.2 Channel 2 Water Regions

CHARACTER REPRESENTATION OF CHANNEL 2 WATER

```
XXXXXXXXXXXXXXXXXXXX
X          WWWWWWWWX
XWWWWWWW WWWWWWWWWWX
XWWWWWWWWWWWWWWWWWWX
XWWWWWWWWWWWWWWWWW X
XWWWWWWWWWWWWWWWWWWX
XWWWWWWWWWWWWWWWWWWX
XWW WWWWWWWWWWWWWWWX
XWW  WWWWWWWWWWWWWX
XWWW WW   WWWWWWWWWX
XWWW      WWWWWWWWWX
XWWW      WWWWWWWWWX
XWWW      WWWWWWWWWX
XWWW  W   WWWWWWWWWX
XWW  WWWWWWWWWWWWWWX
XW  WWWWWWWWWWWWWWWX
X  WWWWWWWWWWWWWWWWX
XWWWWWWWWWWWWWWWWWWX
XWWWWWWWWWWWWWWWWWWX
XWWWWWWWWWWWWWWWWWWX
XWWWWWWWWWWWWWWWWWWX
XWWWWWWWWWWWWWWWWWWX
XWWWWWWWWWWWWWWWWWWX
XWWWWWWWWWWWWWWW    X
XWWWWWWWWWWWWWW    X
XWWWWW W  WWWWWWWWWX
XW WW WWW  WWWW WWWX
XW    WWWW WWWW WWWX
XWW  WWWWW WWWW WWWX
XW W WWWWW W  WWWWWX
XWW  WWWWW WWWWWWWWX
XW    WWWW  WW  WWWX
XW WW WWWW WWWW WWWX
XWW WWWWWWWWWWWWWWWX
XWWWWWWWWWWWWWWWWWWX
XWWWWWWWWWWWWWWW  X
XWWWWWWWWWWWWWWWWWWX
XWWWWWWWWWWWWWWWWWWX
XWWWWWWWWWWWWWWWWWWX
XWWWWWWWWWWWWWWWWWWX
XWWWWWWWWWWWWWWWWWWX
XWWWWWWWWWWWWWWWWWWX
XWWWWWWWWWWWWWW WWX
XWWWWWWWWWWWWWWWWWWX
XWWWWWWWWWWWWWWWWWWX
XWWWWWWWWWWW   WWWX
XWWWWWWWWWWWWWWWWWWX
XWWWWWWWWWWWWWWWWWWX
XWWW WWWWWWWWWWWWWWX
XXXXXXXXXXXXXXXXXXXX
```

G.3 Channel 3 Water Regions

CHARACTER REPRESENTATION OF CHANNEL 3 WATER

```
XXXXXXXXXXXXXXXXXXXX
X   W    W WWWWWWWWX
XWWWWWWWWWWWWWWWWWWX
XWWWWWWWWWWWWWWWWWWX
XWWWWWWWWWWWWWWWWWWX
XWWWWWWWWWWWWWWWWWWX
XWWWWWWWWWWWWWWWWWWX
XW  WWWWWWWWWWWWWWX
XWW  WWWWWWWWWWWWWX
XWWW WW  WWWWWWWWWX
XWWW      WWWWWWWWWX
XWWW      WWWWWWWWWX
XWWW      WWWWWWWWWX
XWWW WW  WWWWWWWWWX
XWW WWWWWWWWWWWWWWX
XW  WWWWWWWWWWWWWWX
X  WWWWWWWWWWWWWWWX
XWWWWWWWWWWWWWWWWWX
XWWWWWWWWWWWWWWWWWX
XWWWWWWWWWWWWWWWWWX
XWWWWWWWWWWWWWWWWWX
XWWWWWWWWWWWWWWWWWX
XWWWWWWWWWWWWWWWWWX
XWWWWWWWWWWWWWWWW  X
XWWWWWWWWWWWWWW   X
XWWWWW    WWWWWWWWWX
XW WW WWW  WWWWWWWWX
XW    WWW WWW WWWX
XWW  WWWW WWWWWWWWX
XW WWWWWW W  WWWWWX
XWWWWWWWW WWWWWWWWX
XWW   WWW  WWW WWWX
XWWWW WWW WWW WWWX
XWW WWWWWWWWWWWWWWX
XWWWWWWWWWWWWWWWWWX
XWWWWWWWWWWWWWWWW X
XWWWWWWWWWWWWWWWWWX
XWWWWWWWWWWWWWWWWWX
XWWWWWWWWWWWWWWWWWX
XWWWWWWWWWWWWWWWWWX
XWWWWWWWWWWWWWWWWWX
XWWWWWWWWWWWWWWWWWX
XWWWWWWWWWWWWWWWWWX
XWWWWWWWWWWWWWWWWWX
XWWWWWWWWWWWWWWWWWX
XWWWWWWWWWWWWW  WWWX
XWWWWWWWWWWWWWWWWWX
XWWWWWWWWWWWWWWWWWX
XWWWWWWWWWWWWWWWWWX
XXXXXXXXXXXXXXXXXXXX
```

G.4 FORTRAN Program: Characterization of Water

```
************************************************************************
*                    SPECTRAL FILTERING PROGRAM                        *
*  Author: John M. Amrine                                              *
*  Created: Nov 1992                                                   *
* ***********************************************************************
*  PROGRAM READS IN THE SPECTRAL VALUES OF AN IMAGE AND CONVERTS THE GRAY  *
*  VALUE  TO A CHARCTER.  IN THIS PROGRAM, THE SPECTRAL FILTER WAS SET TO  *
*  WATER CHANNEL ONE SPOT IMAGERY.  THE PIXELS WITH A GRAY VALUE IN THE    *
*  WATER SPECTRAL RANGE (0-22) WERE SET TO W.  THE BORDER OF 999s IN THE   *
*  INPUT FILE WAS SET TO X.  THE REMAINING PIXELS WERE DESIGNATED BY A     *
*  SPACE.  THIS PROGRAM CAN BE EASILY CHANGED TO FILTER FOR OTHER OBJECTS  *
*  IF THE SPECTRAL RANGE IS KNOWN.  THIS PROGRAM WAS MODIFIED TO PROCESS   *
*  CHANNELS SPOT CHANNELS TWO AND THREE.  THE SPECTRAL RANGE FOR WATER IN  *
*  THESE CHANNELS ARE (5-155) AND (5-200), RESPECTIVELY.                   *
************************************************************************
*  Variables:                                                              *
*          I    :Counter for rows                                          *
*          J    :Counter for columns                                       *
*          COLS :Max number of columns                                     *
*          ROWS :Max number of rows                                        *
*                                                                          *
*          ARRAY: Input matrix (150 X 20)                                  *
*          ARRAY6: Output Character Matrix                                 *
************************************************************************

      INTEGER COLS, ROWS, I, J
      INTEGER ARRAY (150,20)
      CHARACTER ARRAY6 (50,20)

* Files where output will be stored.  To activate this line,
* remove the * from the OPEN line of code.  The * will also have
* to be removed from the WRITE line and the CLOSE line.  Add an *
* to the PRINT 910 line since both the WRITE and PRINT can not both
* be active.

*      OPEN (UNIT = 28, FILE = 'image8')

* Data will be read from external data file
      READ *, ROWS, COLS

* Fill the integer array
      DO 10 I = 1, ROWS
```

```
 READ *, (ARRAY (I, J), J = 1, COLS)

10    CONTINUE

* Subroutine to plot W characters based on Array  gray values
* Print a  title for Channel 1 water representation
        PRINT *
        PRINT *, 'CHARACTER REPRESENTATION OF CHANNEL 1 WATER'
        PRINT *, '-------------------------------------------------'
        PRINT *

      DO 610 I = 1, 50
         DO 620 J = 1, COLS
            IF (ARRAY (I,J) .GE. 999) THEN
                ARRAY6 (I,J) = 'X'

              ELSE IF (ARRAY (I,J) .LT. 256 .AND. ARRAY (I,J)
     +                                 .GE. 23) THEN
                ARRAY6 (I,J) = ' '

            ELSE

                ARRAY6 (I,J) = 'W'

            ENDIF
620      CONTINUE
610   CONTINUE

      DO 630 I = 1, 50
         PRINT 910, (ARRAY6 (I,J), J = 1, COLS)
*         WRITE (28, 910) (ARRAY6 (I,J), J = 1, COLS)
630   CONTINUE

910   FORMAT (1X, 50A)

*      CLOSE (28)

      END
```

Appendix H. *GAMS Program with Modified Objective Function*

In this appendix is contained a modified objective function to bias selection of pixels that are close to the mean gray value of a given land cover type. The objective function also contains lambda weights to perform multicriteria optimization of two channels simultaneously. The objective fuction was not successfully run in GAMS.

```
GAMS 2.20 VAX VMS                                    15-NOV-1992 02:44  PAGE     1
G E N E R A L   A L G E B R A I C   M O D E L I N G   S Y S T E M
C O M P I L A T I O N

   1
   2       SETS
   3          I PIXEL LOCATION / 1*49 /
   4          K NUMBER OF SUBREGIONS TO FIND / 1 /
   5          G IMAGING CHANNEL / A, B /
   6       ALIAS(I,J)
   7       SETS
   8          ADJ(I,J)  ADJACENCY MATRIX
   9              /1.(2,8)
  10               2.(1,3,9)
  11               3.(2,4,10)
  12               4.(3,5,11)
  13               5.(4,6,12)
  14               6.(5,7,13)
  15               7.(6,14)
  16               8.(1,9,15)
  17               9.(2,8,10,16)
  18              10.(3,9,11,17)
  19              11.(4,10,12,18)
  20              12.(5,11,13,19)
  21              13.(6,12,14,20)
  22              14.(7,13,21)
  23              15.(8,16,22)
  24              16.(9,15,17,23)
  25              17.(10,16,18,24)
  26              18.(11,17,19,25)
  27              19.(12,18,20,26)
  28              20.(13,19,21,27)
  29              21.(14,20,28)
```

```
 30              22.(15,23,29)
 31              23.(16,22,24,30)
 32              24.(17,23,25,31)
 33              25.(18,24,26,32)
 34              26.(19,25,27,33)
 35              27.(20,26,28,34)
 36              28.(21,27,35)
 37              29.(22,30,36)
 38              30.(23,29,31,37)
 39              31.(24,30,32,38)
 40              32.(25,31,33,39)
 41              33.(26,32,34,40)
 42              34.(27,33,34,41)
 43              35.(28,34,42)
 44              36.(29,37,43)
 45              37.(30,36,38,44)
 46              38.(31,37,39,45)
 47              39.(32,38,40,46)
 48              40.(33,39,41,47)
 49              41.(34,40,42,48)
 50              42.(35,41,49)
 51              43.(36,44)
 52              44.(37,43,45)
 53              45.(38,44,46)
 54              46.(39,45,47)
```



```
 55              47.(40,46,48)
 56              48.(41,47,49)
 57              49.(42,48)/
 58
 59          TABLE
 60
 61              C(I,G) COST OF EACH PIXEL
 62
 63                      A    B
 64
 65                  1   0    0
 66                  2   0    0
 67                  3   0    0
 68                  4   0    0
 69                  5   0    0
```

70	6	0	0
71	7	0	0
72	8	0	0
73	9	192	48
74	10	163	77
75	11	115	48
76	12	145	39
77	13	151	20
78	14	0	0
79	15	0	0
80	16	145	39
81	17	222	149
82	18	230	161
83	19	210	58
84	20	157	9
85	21	0	0
86	22	0	0
87	23	180	58
88	24	240	183
89	25	249	188
90	26	221	96
91	27	210	29
92	28	0	0
93	29	0	0
94	30	186	20
95	31	216	96
96	32	216	67
97	33	198	10
98	34	174	9
99	35	0	0
100	36	0	0
101	37	145	8
102	38	139	9
103	39	145	7
104	40	168	7
105	41	168	8
106	42	0	0
107	43	0	0
108	44	0	0

109	45	0	0

```
110                 46    0    0
111                 47    0    0
112                 48    0    0
113                 49    0    0  ;
114
115
116
117
118
119          VARIABLES
120             Z          TOTAL COST OF ACQUIRED PIXELS
121             X(I,K)     BINARY VARIABLE
122             P(I,J,K)   BINARY VARIABLE
123             N(I,J,K)   BINARY VARIABLE
124
125          BINARY VARIABLES X,P,N;
126
127          EQUATIONS
128             OBJ        OBJECTIVE FUNCTION
129             C2(I,J,K)  CONSTRAINT FOR CONTIGUITY
130             C4(I,J,K)  CONSTRAINT FOR P&N MUTUAL EXCLUSION
131             C5(I)      CONSTRAINT FOR SUBREGIONS;
132
133          OBJ..
134             Z =E= SUM((I,K), 0.1/(ABS(240 - C(I,"A")*X(I,K)) + 1)
135                        + 0.9/(ABS(180 - C(I,"B")*X(I,K)) + 1));
136
137
138          C2(I,J,K) $ADJ(I,J)..
139             X(I,K) - X(J,K) - P(I,J,K) + N(I,J,K) =E= 0;
140
141
142          C4(I,J,K) $ADJ(I,J)..
143             P(I,J,K) + N(I,J,K) =L= 1;
144
145          C5(I)..
146             SUM(K, X(I,K)) =L= 1;
147
148          X.FX(I,"1")$(C(I,"A") LT 240 OR C(I,"A") GT 249) = 0;
149          X.FX(I,"1")$(C(I,"B") LT 180 OR C(I,"B") GT 190) = 0;
150
151          MODEL SUBREGION /ALL/;
152
153
154          SOLVE SUBREGION USING MIP MAXIMIZING Z;
```

```
****                                                $51,256
**** THE FOLLOWING MIP ERRORS WERE DETECTED IN MODEL SUBREGION:
**** 51 IN EQUATION OBJ        .. ENDOG ARGUMENT(S) IN FUNCTION
**** 51 IN EQUATION OBJ        .. ENDOG ARGUMENT(S) IN FUNCTION
 155
 156
 157          DISPLAY X.L,P.L,N.L,Z.L;
 158
 159
GAMS 2.20 VAX VMS                                15-NOV-1992 02:44  PAGE      5
G E N E R A L   A L G E B R A I C   M O D E L I N G   S Y S T E M
ERROR MESSAGES


 51  ENDOGENOUS FUNCTION ARGUMENT(S) NOT ALLOWED IN LINEAR MODELS
256  ERROR(S) IN ANALYZING SOLVE STATEMENT. MORE DETAIL APPEARS
     BELOW THE SOLVE STATEMENT ABOVE

**** FILE SUMMARY

INPUT   GS092D:[JAMRINE]PRAC2.GMS;1
OUTPUT  GS092D:[JAMRINE]PRAC2.LIS;1

**** USER ERROR(S) ENCOUNTERED

COMPILATION TIME     =        0.810 SECONDS
```

Appendix I. *GAMS Program: Three-Channel Filter*

In this appendix, a three channel filter was performed on the SPOT subimage. The three filters for the gray values were (0-22) for Channel 1, (5-155) for Channel 2, and (5-200) for Channel 3. In the objective function, no lambda weights were used since each channel was equal in weight. No area or border length constraint was used since I wanted to locate all the water regions based on the spectral information contained in the three channels. In the character representation of water, the water areas have been labelled with a W and the other pixels with a space. The 999 border is replaced by an X border.

I.1 GAMS Program to Locate Water Regions

```
GAMS 2.20 VAX VMS                                17-NOV-1992 19:53  PAGE     1
G E N E R A L   A L G E B R A I C   M O D E L I N G   S Y S T E M
C O M P I L A T I O N


   1
   2
   3       SETS
   4               I PIXEL LOCATION / 1*1000 /
   5               K NUMBER OF SUBREGIONS TO  FIND / 1 /
   6               G IMAGING CHANNEL / A, B, C /;
   7
   8         ALIAS (I,J)
   9         SETS
  10           ADJ (I,J) ADJACENCY MATRIX
  11               /1.  (2,21)
  Remaining 999 entries for adjaceny matrix deleted to save space.

 1012       TABLE
 1013
 1014         C(I,G) COST OF EACH PIXEL
 1015
 1016                   A    B    C
 1017
```

```
1018              1  999  999  999
  Remaining 999 entries for cost matrix deleted to save space.
2023
2024       VARIABLES
2025            Z          TOTAL COST OF ACQUIRED PIXELS
2026            X(I,K)     BINARY VARIABLE
2027            P(I,J,K)   BINARY VARIABLE
2028            N(I,J,K)   BINARY VARIABLE
2029
2030       BINARY VARIABLES X,P,N;
2031
2032       EQUATIONS
2033            OBJ          OBJECTIVE FUNCTION
2034            C2(I,J,K) CONSTRAINT FOR CONTIGUITY
2035            C4(I,J,K) CONSTRAINT FOR P&N MUTUAL EXCLUSION
2036            C5(I)        CONSTRAINT FOR SUBREGIONS;
2037
2038       OBJ ..
2039                Z =E= SUM((I,K,G), C(I,G)*X(I,K));
2040
2041       C2(I,J,K) $ADJ(I,J) ..
2042                X(I,K) - X(J,K) - P(I,J,K) + N(I,J,K) =E= 0;
2043
2044       C4(I,J,K) $ADJ(I,J) ..
2045                P(I,J,K) + N(I,J,K) =L= 1;
2046
2047       C5(I) ..
2048               SUM(K, X(I,K)) =L= 1;
2049
2050
2051
2052       X.FX(I,"1")$(C(I,"A") GT 22) = 0;
```



```
2053       X.FX(I,"1")$(C(I,"B") LT 5 OR C(I,"B") GT 155) = 0;
2054       X.FX(I,"1")$(C(I,"C") LT 5 OR C(I,"C") GT 200) = 0;
2055
2056
2057       MODEL SUBREGION /ALL/;
2058
2059       OPTIONS ITERLIM = 5000, RESLIM = 5000, WORK = 25000;
2060
```

```
 2061       SOLVE SUBREGION USING MIP MAXIMIZING Z;
 2062
 2063       DISPLAY X.L,P.L,N.L,Z.L;
 2064
 2065
 2066
GAMS 2.20 VAX VMS                                17-NOV-1992 19:53  PAGE     54
G E N E R A L   A L G E B R A I C   M O D E L I N G   S Y S T E M
MODEL STATISTICS    SOLVE SUBREGION USING MIP FROM LINE 2061

MODEL STATISTICS

BLOCKS OF EQUATIONS     4     SINGLE EQUATIONS   8721
BLOCKS OF VARIABLES     4     SINGLE VARIABLES   8721
NON ZERO ELEMENTS   25161     DISCRETE VARIABLES 7800

GENERATION TIME      =       79.690 SECONDS


EXECUTION TIME       =       82.310 SECONDS
GAMS 2.20 VAX VMS                                17-NOV-1992 20:41  PAGE     55
G E N E R A L   A L G E B R A I C   M O D E L I N G   S Y S T E M
SOLUTION REPORT     SOLVE SUBREGION USING MIP FROM LINE 2061


              S O L V E     S U M M A R Y

     MODEL   SUBREGION           OBJECTIVE  Z
     TYPE    MIP                 DIRECTION  MAXIMIZE
     SOLVER  ZOOM                FROM LINE  2061

**** SOLVER STATUS     2 ITERATION INTERRUPT
**** MODEL STATUS      9 INTERMEDIATE NON-INTEGER
**** OBJECTIVE VALUE           11710.0000

 RESOURCE USAGE, LIMIT       1345.760     5000.000
 ITERATION COUNT, LIMIT      2681         5000

 Z O O M / X M P    ---    Version 2.1 Oct 1988

 Courtesy of Dr Roy E. Marsten,
             Department of Management Information Systems,
             University of Arizona,
             Tucson     Arizona  85721,  U.S.A.
```

```
PROBLEM SPECIFICATIONS
----------------------
BEGIN
   HEURISTIC NO
   DIVE NO
END
Work space needed (estimate)   --  923463 words.
Work space available           --  923463 words.

ZOOM: Due to the model status, the following
solution is probably meaningless.

The branch and bound tree contained   480 nodes (max. 25000 nodes).

Iterations: Initial LP          1180, Time:  325.410
            Heuristic              0,        1.70996
            Branch and bound    1501,        984.340
            Final LP               0,       0.000000E+00

GAMS 2.20 VAX VMS                              17-NOV-1992 20:41  PAGE    406
G E N E R A L   A L G E B R A I C   M O D E L I N G   S Y S T E M
E X E C U T I N G

----   2063 VARIABLE  X.L           BINARY VARIABLE

              1

142        1.000
162        1.000
163        1.000
182        1.000
183        1.000
184        1.000
202        1.000
203        1.000
204        1.000
222        1.000
223        1.000
224        1.000
242        1.000
243        1.000
244        1.000
262        1.000
263        1.000
264        1.000
```

```
282        1.000
283        1.000
302        1.000
528        1.000
529        1.000
542        1.000
547        1.000
548        1.000
549        1.000
550        1.000
562        1.000
567        1.000
568        1.000
569        1.000
570        1.000
582        1.000
587        1.000
588        1.000
589        1.000
590        1.000
607        1.000
608        1.000
609        1.000
610        1.000
622        1.000
627        1.000
628        1.000
629        1.000
630        1.000
648        1.000
649        1.000
665        1.000
GAMS 2.20 VAX VMS                                17-NOV-1992 20:41  PAGE    407
G E N E R A L   A L G E B R A I C   M O D E L I N G   S Y S T E M
E X E C U T I N G

      2063 VARIABLE  X.L          BINARY VARIABLE

                1

686        1.000
706        1.000
710        1.000
879        1.000
935        1.000
```

```
936        1.000
937        1.000
938        1.000
951        1.000
952        1.000
953        1.000
954        1.000
955        1.000
956        1.000
957        1.000
958        1.000
959        1.000
968        1.000
969        1.000
970        1.000
971        1.000
972        1.000
973        1.000
974        1.000
975        1.000
976        1.000
977        1.000
978        1.000
979        1.000

----   2063 VARIABLE  Z.L                    =    11710.000 TOTAL COST OF
                                                            ACQUIRED PIXELS
GAMS 2.20 VAX VMS                              17-NOV-1992 20:41  PAGE    413
G E N E R A L   A L G E B R A I C   M O D E L I N G   S Y S T E M
E X E C U T I N G

**** FILE SUMMARY

INPUT   GS092D:[JAMRINE]R1003.GMS;6
OUTPUT  GS092D:[JAMRINE]R1003.LIS;4

EXECUTION TIME       =       34.340 SECONDS
```

I.2 Character Representation of Water Based on Three-Channel Filtering

```
XXXXXXXXXXXXXXXXXXXX
X                  X
X                  X
X                  X
X                  X
X                  X
X                  X
XW                 X
XWW                X
XWWW               X
XWWW               X
XWWW               X
XWWW               X
XWWW               X
XWW                X
XW                 X
X                  X
X                  X
X                  X
X                  X
X                  X
X                  X
X                  X
X                  X
X                  X
X                  X
X                  X
X      WW          X
XW    WWWW         X
XW    WWWW         X
XW    WWWW         X
X     WWWW         X
XW    WWWW         X
X      WW          X
X    W             X
X     W            X
X    W   W         X
X                  X
X                  X
X                  X
X                  X
X                  X
X                  X
X                  X
X                 WX
X                  X
X                  X
X             WWW X
X         WWWWWWWWX
X      WWWWWWWWWWWX
XXXXXXXXXXXXXXXXXXXX
```

Appendix J. *Channel 1 - Channel 2 Program*

In this appendix is contained a partial listing of a weighted, two-channel, multiobjective GAMS program. In this program, Channel 1 has weight of 0.1 and Channel 2 has a weight of 0.9. This program was repeated nine times with different lambda weights. The same program was also modified to run the other two-channel combinations: Channel 1 - Channel 3 and Channel 2 - Channel 3. In the partial listing, the adjacency and cost matices have been edited due to their length. Due to the length of these input tables, they were generated with two separate FORTRAN programs. These programs are also included in this appendix.

J.1 GAMS Program: Channel 1 - Channel 2

```
GAMS 2.20 VAX VMS                                13-NOV-1992 10:38  PAGE      1
G E N E R A L   A L G E B R A I C   M O D E L I N G   S Y S T E M
C O M P I L A T I O N

   1
   2
   3       SETS
   4               I PIXEL LOCATION / 1*1000 /
   5               K NUMBER OF SUBREGIONS TO  FIND / 1 /
   6               G IMAGING CHANNEL / A, B /;
   7
   8         ALIAS (I,J)
   9         SETS
  10           ADJ (I,J) ADJACENCY MATRIX
  11               /1.  (2,21)
Remaining 999 entries for adjacency matrix were deleted.
 1013
 1014
 1015       TABLE
 1016
 1017         C(I,G) COST OF EACH PIXEL
 1018
 1019                    A    B
 1020
```

```
1021              1  999  999
Remaining 999 entries for cost matrix were deleted.
2021
2022       PARAMETERS
2023             M(K) PIXEL SIZE OF KTH SUBREGION
2024                /1 24/
2025
2026            L(K) BORDER OF KTH SUBREGION
2027                /1 44/ ;
2028
2029
2030       VARIABLES
2031            Z          TOTAL COST OF ACQUIRED PIXELS
2032            X(I,K)     BINARY VARIABLE
2033            P(I,J,K)   BINARY VARIABLE
2034            N(I,J,K)   BINARY VARIABLE
2035
2036       BINARY VARIABLES X,P,N;
2037
2038       EQUATIONS
2039            OBJ          OBJECTIVE FUNCTION
2040            C1(K)      CONSTRAINT FOR SIZE OF KTH SUBREGION
2041            C2(I,J,K) CONSTRAINT FOR CONTIGUITY
2042            C3(K)      CONSTRAINT FOR BORDER LENGTH
2043            C4(I,J,K) CONSTRAINT FOR P&N MUTUAL EXCLUSION
2044            C5(I)        CONSTRAINT FOR SUBREGIONS;
2045
2046       OBJ ..
2047                Z =E= SUM((I,K), 0.1*C(I,"A")*X(I,K)
2048                           + 0.9*C(I,"B")*X(I,K));
2049
2050       C1(K)..
2051              SUM(I,X(I,K)) =E= M(K);
2052
GAMS 2.20 VAX VMS                                13-NOV-1992 10:38  PAGE     39
G E N E R A L   A L G E B R A I C   M O D E L I N G   S Y S T E M
C O M P I L A T I O N


2053       C2(I,J,K) $ADJ(I,J) ..
2054                X(I,K) - X(J,K) - P(I,J,K) + N(I,J,K) =E= 0;
2055
2056       C3(K)..
2057              SUM((I,J) $ADJ(I,J), P(I,J,K) + N(I,J,K)) =E= L(K);
2058
```

```
2059        C4(I,J,K) $ADJ(I,J) ..
2060                P(I,J,K) + N(I,J,K) =L= 1;
2061
2062        C5(I) ..
2063                SUM(K, X(I,K)) =L= 1;
2064
2065        X.FX(I,"1")$(C(I,"A") GT 22) = 0;
2066        X.FX(I,"1")$(C(I,"B") LT 5 OR C(I,"B") GT 149) = 0;
2067
2068
2069
2070        MODEL SUBREGION /ALL/;
2071
2072        OPTIONS ITERLIM = 5000, RESLIM = 5000, WORK = 25000;
2073
2074        SOLVE SUBREGION USING MIP MAXIMIZING Z;
2075
2076        DISPLAY X.L,P.L,N.L,Z.L;
2077
2078
2079
```


MODEL STATISTICS SOLVE SUBREGION USING MIP FROM LINE 2074

MODEL STATISTICS

BLOCKS OF EQUATIONS	6	SINGLE EQUATIONS	8723
BLOCKS OF VARIABLES	4	SINGLE VARIABLES	8721
NON ZERO ELEMENTS	33881	DISCRETE VARIABLES	7799

GENERATION TIME = 117.100 SECONDS

EXECUTION TIME = 119.430 SECONDS

SOLUTION REPORT SOLVE SUBREGION USING MIP FROM LINE 2074

S O L V E S U M M A R Y

MODEL	SUBREGION	OBJECTIVE	Z
TYPE	MIP	DIRECTION	MAXIMIZE
SOLVER	ZOOM	FROM LINE	2074

```
**** SOLVER STATUS     1 NORMAL COMPLETION
**** MODEL STATUS      1 OPTIMAL
**** OBJECTIVE VALUE            1706.0000

 RESOURCE USAGE, LIMIT        554.890     5000.000
 ITERATION COUNT, LIMIT      1544         5000

 Z O O M / X M P    ---    Version 2.1 Oct 1988

 Courtesy of Dr Roy E. Marsten,

            Department of Management Information Systems,
            University of Arizona,
            Tucson     Arizona  85721,  U.S.A.

 PROBLEM SPECIFICATIONS
 ----------------------
 BEGIN
    HEURISTIC NO
    DIVE NO
 END

 Work space needed (estimate)   -- 1041383 words.
 Work space available           -- 1041383 words.

 Iterations: Initial LP          1544, Time:  516.930
             Heuristic              0,       0.000000E+00
             Branch and bound       0,       0.000000E+00
             Final LP               0,       0.000000E+00

GAMS 2.20 VAX VMS                              13-NOV-1992 11:05  PAGE    486
G E N E R A L   A L G E B R A I C   M O D E L I N G   S Y S T E M
E X E C U T I N G


----   2076 VARIABLE  X.L          BINARY VARIABLE

               1

528         1.000
529         1.000
547         1.000
548         1.000
549         1.000
```

```
550        1.000
567        1.000
568        1.000
569        1.000
570        1.000
587        1.000
588        1.000
589        1.000
590        1.000
607        1.000
608        1.000
609        1.000
610        1.000
627        1.000
628        1.000
629        1.000
630        1.000
648        1.000
649        1.000

GAMS 2.20 VAX VMS                                13-NOV-1992 11:05  PAGE    487
G E N E R A L   A L G E B R A I C   M O D E L I N G   S Y S T E M
E X E C U T I N G

----   2076 VARIABLE  Z.L                 =     1706.000 TOTAL COST OF
                                                         ACQUIRED PIXELS

**** FILE SUMMARY

INPUT   GS092D:[JAMRINE]R1K19ASB.GMS;1
OUTPUT  GS092D:[JAMRINE]R1K19ASB.LIS;1

EXECUTION TIME       =       37.810 SECONDS
```

J.2 FORTRAN ProgramL: GAMS Adjacency Matrix Generation

```
*****************************************************************************
*     1ST ORDER NEIGHBOR MATRIX (ADJACENCY MATRIX) GENERATION PROGRAM       *
*                                                                           *
*  AUTHOR: John M. Amrine                                                   *
*  CREATED: Nov 1992                                                        *
*****************************************************************************
```

```
*  THIS PROGRAM CREATES A GAMS INPUT FILE.  THE PROGRAM LISTS THE FIRST    *
*  ORDER NEIGHBORS FOR A 50 X 20 (ROWS X COLUMNS) MATRIX.  THIS PROGRAM    *
*  CAN BE EASILY MODIFIED TO ACCOMODATE ANY SIZE MATRIX.                   *
*                                                                          *
****************************************************************************
*  Variables:                                                              *
*           I    :Counter for pixel location                               *
*           MAX  :Total number of pixels in matrix                         *
****************************************************************************


      INTEGER  I, MAX


      MAX = 1000
      I = 1
      PRINT *, '       ALIAS (I,J)'
      PRINT *, '       SETS'
      PRINT *, '         ADJ (I,J) ADJACENCY MATRIX'
      PRINT *
      DO 10 I = 1, MAX
         IF (I .EQ. 1) THEN
            PRINT 1100, I, I+1, I+20
          ELSE IF (I .EQ. 20) THEN
            PRINT 1200, I, I-1, I+20
          ELSE IF (I .EQ. 981) THEN
            PRINT 1200, I, I-20, I+1
          ELSE IF (I .EQ. 1000) THEN
            PRINT 1250, I, I-20, I-1
          ELSE IF (I .GT. 1 .AND. I .LT. 20) THEN
            PRINT 1300, I, I-1, I+1, I+20
   ELSE IF (I .GT. 981 .AND. I .LT. 1000) THEN
            PRINT 1300, I, I-20, I-1, I+1
   ELSE IF (I .GT. 20 .AND. I .LT. 981 .AND.
     +          MOD(I,20) .EQ. 1) THEN
                    PRINT 1300, I, I-20, I+1, I+20
   ELSE IF (I .GT. 20 .AND. I .LT. 981 .AND.
     +          MOD(I,20) .EQ. 0) THEN
                    PRINT 1300, I, I-20, I-1, I+20
          ELSE
            PRINT 1400, I, I-20, I-1, I+1, I+20
          END IF
```

```
10    CONTINUE

1100  FORMAT (13X, '/', I1, '.', 2X, '(', I1, ',', I2, ')' )

1200   FORMAT (10X, I5, '.', 2X, '(', I4, ',', I4, ')' )

1250  FORMAT (10X, I5, '.', 2X, '(', I4, ',', I4, ') /' )

1300  FORMAT (10X, I5, '.', 2X, '(', 2(I4, ','), I4,  ')' )

1400  FORMAT (10X, I5, '.', 2X, '(', 3(I4, ','), I4,  ')' )

      END
```

J.3 FORTRAN Program: GAMS Cost Matrix Generation

```
*************************************************************************
*                                                                       *
*                GAMS COST MATRIX GENERATION PROGRAM                    *
* AUTHOR: John M. Amrine                                                *
* CREATED: Nov 1992                                                     *
*************************************************************************
*  THIS PROGRAM CREATES A GAMS INPUT FILE.  THE PROGRAM READS THE GRAY  *
*  VALUES FOR ONE LOCATION AND LISTS THE GRAY VALUES FOR EACH CHANNEL.  *
*  CURRENTLY THE PROGRAM ONLY READS THREE CHANNELS SINCE SPOT IMAGERY   *
*  IS USED.  THE THREE CHANNELS ARE DESIGNATED BY A, B, AND C.  THE PROGRAM*
*  CAN BE EASILY MODIFIED TO ACCOMODATE MORE OR LESS CHANNELS.  THE INPUT *
*  MATRIX IS 150 (ROWS) X 20 (COLUMNS).  THIS MATRIX ACTUALLY CONTAINS  *
*  THREE IMAGES OF THE SAME AREA, CORRESPONDING TO THE THREE CHANNELS.  THE*
*  FIRST FIFTY ROWS CORRESPONDS TO THE FIRST CHANNEL, THE SECOND FIFTY  *
*  ROWS (51-100) TO THE SECOND CHANNEL, AND THE LAST FIFTY ROWS TO THE  *
*  THIRD CHANNEL.  THE INPUT FILE FOR THIS FORTRAN PROGRAM IS READ IN   *
*  USING THE a.out < filename.ext UNIX COMMAND.                         *
*                                                                       *
*************************************************************************
*  Variables:                                                           *
*          I    :Counter for rows                                       *
*          J    :Counter for columns                                    *
*          A    :Gray value of pixel in Channel 1 at location COUNT     *
*          B    :Gray value of pixel in Channel 2 at location COUNT     *
```

```
*           C    :Gray value of pixel in Channel 3 at location COUNT       *
*           COLS :Max number of columns                                    *
*           ROWS :Max number of rows                                       *
*           COUNT:Counter for pixel location                               *
*                                                                          *
*           ARRAY: Input matrix (150 X 20)                                 *
****************************************************************************


      INTEGER COLS, ROWS, I, J, COUNT, A, B
      INTEGER ARRAY (150,20)


* Data will be read from external data file

      READ *, ROWS, COLS

* Fill the integer array

      DO 10 I = 1, ROWS
 READ *, (ARRAY (I, J), J = 1, COLS)

10    CONTINUE

      PRINT *, '     TABLE'
      PRINT *
      PRINT *, '       C(I,G) COST OF EACH PIXEL'
      PRINT *
      PRINT *, '                    A    B'
      PRINT *

* Read gray values of each channel by position

      COUNT = 1
      DO 20 I = 1, 50
 DO 30 J = 1, COLS
            A = ARRAY (I, J)
            B = ARRAY (I + 50, J)
*            C = ARRAY (I + 100, J)
               IF (COUNT .EQ. 1000) THEN
                  GO TO 40
               ELSE
```

```
                PRINT 1000, COUNT, A, B
             END IF
             GO TO 50
40         PRINT 1100, COUNT, A, B
50         COUNT = COUNT + 1
30      CONTINUE
20   CONTINUE



1000 FORMAT (10X, I4, 2I5)
1100 FORMAT (10X, I4, 2I5, ' ;' )

     END
```

Appendix K. *Channel 1 - Channel 3 Program*

In this appendix is contained the partial listing of Channel 1 - Channel 3, weighted, multiobjective GAMS program. In this program, Channel 1 has weight of 0.1 and Channel 3 has a weight of 0.9. This program was repeated nine times with different lambda weights. In the partial listing, the adjacency and cost matices have been edited due to their length. Due to the length of these input tables, they were generated with two separate FORTRAN programs. These programs are included in Appendix J.

K.1 GAMS Program: Channel 1 - Channel 3

```
GAMS 2.20 VAX VMS                                    18-NOV-1992 03:23  PAGE      1
G E N E R A L   A L G E B R A I C   M O D E L I N G   S Y S T E M
C O M P I L A T I O N


   1
   2
   3       SETS
   4               I PIXEL LOCATION / 1*1000 /
   5               K NUMBER OF SUBREGIONS TO  FIND / 1 /
   6               G IMAGING CHANNEL / A, C /;
   7
   8         ALIAS (I,J)
   9         SETS
  10           ADJ (I,J) ADJACENCY MATRIX
  11               /1.  (2,21)
Remaining 999 entries deleted.
 1013
 1014
 1015       TABLE
 1016
 1017         C(I,G) COST OF EACH PIXEL
 1018
 1019                    A    C
 1020
 1021               1  999  999
```

```
Remaining 999 entries deleted.
 2022
 2023       PARAMETERS
 2024              M(K) PIXEL SIZE OF KTH SUBREGION
 2025                 /1 24/
 2026
 2027             L(K) BORDER OF KTH SUBREGION
 2028                 /1 44/ ;
 2029
 2030
 2031
 2032       VARIABLES
 2033            Z          TOTAL COST OF ACQUIRED PIXELS
 2034            X(I,K)     BINARY VARIABLE
 2035            P(I,J,K)   BINARY VARIABLE
 2036            N(I,J,K)   BINARY VARIABLE
 2037
 2038       BINARY VARIABLES X,P,N;
 2039
 2040       EQUATIONS
 2041            OBJ          OBJECTIVE FUNCTION
 2042            C1(K)      CONSTRAINT FOR SIZE OF KTH SUBREGION
 2043            C2(I,J,K) CONSTRAINT FOR CONTIGUITY
 2044            C3(K)      CONSTRAINT FOR BORDER LENGTH
 2045            C4(I,J,K) CONSTRAINT FOR P&N MUTUAL EXCLUSION
 2046            C5(I)        CONSTRAINT FOR SUBREGIONS;
 2047
 2048       OBJ ..
 2049                Z =E= SUM((I,K), 0.1*C(I,"A")*X(I,K)
 2050                          + 0.9*C(I,"C")*X(I,K));
 2051
 2052       C1(K)..
GAMS 2.20 VAX VMS                                18-NOV-1992 03:23  PAGE     39
G E N E R A L   A L G E B R A I C   M O D E L I N G   S Y S T E M
C O M P I L A T I O N


 2053              SUM(I,X(I,K)) =E= M(K);
 2054
 2055       C2(I,J,K) $ADJ(I,J) ..
 2056                X(I,K) - X(J,K) - P(I,J,K) + N(I,J,K) =E= 0;
 2057
 2058       C3(K)..
 2059              SUM((I,J) $ADJ(I,J), P(I,J,K) + N(I,J,K)) =E= L(K);
 2060
```

```
2061        C4(I,J,K) $ADJ(I,J) ..
2062                P(I,J,K) + N(I,J,K) =L= 1;
2063
2064        C5(I) ..
2065                SUM(K, X(I,K)) =L= 1;
2066
2067        X.FX(I,"1")$(C(I,"A") GT 22) = 0;
2068        X.FX(I,"1")$(C(I,"C") LT 5 OR C(I,"C") GT 200) = 0;
2069
2070
2071        MODEL SUBREGION /ALL/;
2072
2073        OPTIONS ITERLIM = 5000, RESLIM = 5000, WORK = 25000;
2074
2075        SOLVE SUBREGION USING MIP MAXIMIZING Z;
2076
2077        DISPLAY X.L,P.L,N.L,Z.L;
2078
2079
2080
GAMS 2.20 VAX VMS                                18-NOV-1992 03:23  PAGE    134
G E N E R A L   A L G E B R A I C   M O D E L I N G   S Y S T E M
MODEL STATISTICS    SOLVE SUBREGION USING MIP FROM LINE 2075


MODEL STATISTICS

BLOCKS OF EQUATIONS     6     SINGLE EQUATIONS   8723
BLOCKS OF VARIABLES     4     SINGLE VARIABLES   8721
NON ZERO ELEMENTS   33881     DISCRETE VARIABLES 7801

GENERATION TIME      =      118.890 SECONDS


EXECUTION TIME       =      121.320 SECONDS
GAMS 2.20 VAX VMS                                18-NOV-1992 04:13  PAGE    135
G E N E R A L   A L G E B R A I C   M O D E L I N G   S Y S T E M
SOLUTION REPORT     SOLVE SUBREGION USING MIP FROM LINE 2075

              S O L V E     S U M M A R Y

     MODEL   SUBREGION           OBJECTIVE  Z
     TYPE    MIP                 DIRECTION  MAXIMIZE
     SOLVER  ZOOM                FROM LINE  2075
```

```
**** SOLVER STATUS     1 NORMAL COMPLETION
**** MODEL STATUS      1 OPTIMAL
**** OBJECTIVE VALUE            2253.2000

 RESOURCE USAGE, LIMIT        511.530     5000.000
 ITERATION COUNT, LIMIT      1380         5000

 Z O O M / X M P    ---    Version 2.1 Oct 1988

 Courtesy of Dr Roy E. Marsten,
             Department of Management Information Systems,
             University of Arizona,
             Tucson     Arizona  85721,  U.S.A.

 PROBLEM SPECIFICATIONS
 ----------------------
 BEGIN
    HEURISTIC NO
    DIVE NO
 END

 Work space needed (estimate)   -- 1041389 words.
 Work space available           -- 1041389 words.

 Iterations: Initial LP          1380, Time:  471.460
             Heuristic              0,       0.000000E+00
             Branch and bound       0,       0.000000E+00
             Final LP               0,       0.000000E+00

GAMS 2.20 VAX VMS                              18-NOV-1992 04:13  PAGE    486
G E N E R A L   A L G E B R A I C   M O D E L I N G   S Y S T E M
E X E C U T I N G

----   2077 VARIABLE  X.L          BINARY VARIABLE

               1

528        1.000
529        1.000
547        1.000
548        1.000
549        1.000
550        1.000
567        1.000
568        1.000
```

```
569        1.000
570        1.000
587        1.000
588        1.000
589        1.000
590        1.000
607        1.000
608        1.000
609        1.000
610        1.000
627        1.000
628        1.000
629        1.000
630        1.000
648        1.000
649        1.000

GAMS 2.20 VAX VMS                                18-NOV-1992 04:13  PAGE    487
G E N E R A L   A L G E B R A I C   M O D E L I N G   S Y S T E M
E X E C U T I N G

----    2077 VARIABLE  Z.L                  =     2253.200 TOTAL COST OF
                                                           ACQUIRED PIXELS

**** FILE SUMMARY

INPUT   GS092D:[JAMRINE]R19BSB.GMS;1
OUTPUT  GS092D:[JAMRINE]R19BSB.LIS;1

EXECUTION TIME       =       41.100 SECONDS
```

Appendix L. *Vegetation Indicies Results & Programs*

In this appendix are contained the following matrices: vegetation index, normalized vegetation index, and near-IR/red. These matrices were calculated using the FORTRAN programs included in this appendix. When using the SPOT multispectral imagery for these matrices, only Channels 2 (red) and 3 (near-IR) were used.

L.1 VI Matrix

0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.
0.	-1.	-2.	18.	17.	26.	8.	-1.	-2.	6.	9.	28.	11.	6.	12.	17.	23.	17.	11.	0.
0.	1.	1.	12.	12.	23.	1.	-10.	0.	18.	12.	17.	16.	15.	4.	1.	17.	11.	1.	0.
0.	-10.	-10.	-20.	-12.	-12.	-1.	9.	4.	-5.	-10.	-11.	8.	0.	-10.	15.	15.	14.	18.	0.
0.	-11.	5.	5.	-22.	35.	7.	6.	12.	-3.	0.	17.	0.	-1.	25.	25.	24.	5.	17.	0.
0.	12.	6.	-4.	-1.	9.	19.	6.	20.	6.	-1.	-1.	10.	6.	15.	5.	33.	20.	17.	0.
0.	12.	28.	-5.	-11.	1.	7.	4.	5.	7.	16.	-3.	6.	12.	5.	-3.	33.	14.	6.	0.
0.	21.	40.	24.	12.	-3.	5.	17.	-5.	7.	15.	17.	6.	1.	-1.	5.	6.	24.	-11.	0.
0.	52.	45.	6.	18.	-14.	-14.	11.	6.	6.	6.	12.	25.	7.	-1.	14.	-10.	6.	12.	0.
0.	25.	1.	23.	0.	12.	17.	7.	-2.	6.	12.	29.	-3.	16.	7.	5.	15.	20.	-5.	0.
0.	53.	1.	14.	0.	-3.	-2.	-2.	-2.	-5.	-11.	8.	-3.	6.	7.	25.	6.	1.	6.	0.
0.	30.	43.	43.	1.	-2.	-2.	-3.	-5.	-5.	-4.	24.	33.	16.	7.	-3.	7.	10.	-14.	0.
0.	43.	26.	11.	-1.	-5.	-4.	-5.	-5.	-5.	0.	24.	15.	8.	18.	15.	6.	0.	4.	0.
0.	74.	36.	17.	-5.	-22.	-10.	-6.	-5.	6.	11.	10.	7.	0.	-1.	23.	10.	17.	11.	0.
0.	4.	40.	-4.	-11.	-4.	-31.	-11.	-5.	-3.	12.	6.	16.	2.	1.	24.	14.	23.	11.	0.
0.	25.	29.	-2.	-11.	-21.	-30.	-16.	0.	-10.	-2.	-5.	33.	7.	17.	15.	-4.	0.	12.	0.
0.	29.	18.	-4.	-2.	-12.	-12.	6.	21.	-3.	-1.	16.	26.	33.	14.	15.	29.	6.	14.	0.
0.	28.	5.	-12.	-2.	-2.	-1.	-3.	15.	-12.	8.	1.	-1.	21.	6.	4.	10.	-5.	14.	0.
0.	-12.	-10.	-30.	-20.	-1.	-1.	-2.	5.	6.	-12.	-10.	1.	-12.	20.	-11.	1.	12.	15.	0.
0.	-2.	-11.	0.	-1.	-1.	16.	6.	5.	10.	24.	44.	35.	7.	-12.	6.	6.	23.	15.	0.
0.	-5.	7.	17.	35.	27.	9.	8.	-13.	5.	5.	-3.	20.	15.	9.	15.	6.	6.	22.	0.
0.	6.	6.	1.	7.	-1.	9.	15.	-2.	16.	26.	16.	15.	15.	2.	14.	-5.	17.	23.	0.
0.	-23.	-5.	1.	24.	4.	6.	15.	-20.	8.	-2.	25.	6.	15.	16.	11.	0.	6.	28.	0.
0.	14.	-3.	16.	15.	-5.	-3.	7.	7.	-1.	-2.	-2.	6.	33.	12.	11.	6.	18.	19.	0.
0.	20.	1.	16.	9.	-12.	6.	28.	22.	6.	-12.	18.	8.	35.	6.	6.	-4.	4.	-3.	0.
0.	1.	6.	5.	-1.	15.	19.	35.	14.	7.	0.	0.	1.	0.	8.	11.	0.	-5.	6.	0.
0.	17.	-1.	1.	18.	24.	11.	1.	17.	10.	4.	10.	17.	2.	15.	1.	-5.	-5.	15.	0.
0.	6.	4.	-2.	0.	24.	6.	39.	25.	45.	3.	23.	23.	17.	17.	-1.	1.	1.	1.	0.
0.	1.	28.	18.	-1.	17.	24.	40.	44.	34.	2.	23.	24.	1.	6.	6.	0.	0.	23.	0.
0.	1.	13.	1.	-11.	6.	15.	23.	28.	17.	7.	5.	0.	0.	1.	-5.	14.	17.	12.	0.
0.	0.	-11.	-16.	0.	28.	12.	37.	34.	6.	9.	15.	18.	0.	-2.	18.	6.	17.	23.	0.
0.	7.	0.	-5.	-4.	13.	14.	39.	52.	6.	-1.	18.	22.	23.	0.	-1.	17.	11.	31.	0.
0.	17.	11.	-5.	12.	6.	45.	24.	26.	28.	0.	23.	17.	23.	12.	2.	6.	15.	44.	0.
0.	18.	-14.	-1.	-1.	1.	33.	42.	34.	39.	17.	-1.	0.	1.	15.	18.	-5.	24.	10.	0.
0.	17.	-4.	-12.	-12.	0.	23.	21.	14.	7.	17.	-1.	-2.	15.	-5.	6.	1.	17.	17.	0.
0.	15.	-3.	-5.	1.	-2.	-2.	7.	-1.	0.	25.	31.	0.	6.	6.	18.	12.	17.	-1.	0.
0.	-1.	0.	17.	-4.	7.	-2.	-3.	-3.	14.	10.	23.	8.	9.	-14.	6.	6.	11.	17.	0.
0.	1.	2.	1.	-1.	27.	23.	25.	10.	5.	-3.	19.	19.	1.	-12.	11.	-5.	17.	17.	0.
0.	19.	2.	1.	10.	19.	25.	-12.	-3.	-2.	1.	2.	-2.	20.	14.	20.	4.	6.	14.	0.
0.	4.	1.	1.	0.	-1.	18.	-3.	25.	25.	9.	-1.	24.	6.	10.	-5.	15.	12.	15.	0.
0.	1.	1.	1.	0.	2.	27.	16.	26.	17.	0.	0.	5.	17.	15.	15.	17.	17.	11.	0.
0.	18.	2.	2.	27.	7.	8.	18.	17.	25.	24.	9.	8.	7.	15.	17.	23.	17.	6.	0.
0.	15.	26.	27.	26.	-10.	1.	1.	24.	28.	28.	25.	18.	7.	7.	11.	11.	15.	23.	0.
0.	34.	17.	16.	0.	9.	26.	23.	24.	15.	24.	20.	15.	23.	-2.	14.	-5.	-1.	-19.	0.
0.	11.	5.	0.	20.	33.	14.	6.	34.	15.	6.	6.	12.	12.	6.	15.	-23.	1.	6.	0.
0.	4.	24.	24.	14.	15.	15.	6.	5.	0.	6.	23.	17.	22.	11.	29.	11.	6.	17.	0.
0.	33.	50.	33.	-12.	14.	0.	17.	18.	23.	34.	29.	40.	23.	45.	29.	17.	51.	34.	0.
0.	17.	31.	42.	17.	18.	12.	6.	17.	12.	17.	35.	44.	25.	1.	18.	15.	16.	25.	0.
0.	6.	17.	12.	11.	12.	18.	14.	25.	6.	10.	17.	24.	15.	18.	52.	33.	34.	35.	0.
0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.

L.2 NVI Matrix

0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.05	0.04	0.07	0.02	0.00	0.00	0.02	0.02	0.08	0.05	0.02	0.05	0.07	0.07	0.07	0.05	0.00
0.00	0.00	0.00	0.05	0.05	0.09	0.00	-0.04	0.00	0.06	0.04	0.07	0.17	0.10	0.02	0.00	0.06	0.05	0.00	0.00
0.00	-0.33	-0.33	-0.34	-0.10	-0.12	-0.02	0.31	0.02	-0.03	-0.21	-0.16	0.17	0.00	-0.21	0.10	0.07	0.08	0.06	0.00
0.00	-0.05	0.03	0.03	-0.15	0.38	0.08	0.04	0.05	-0.03	0.00	0.30	0.00	-0.06	0.24	0.13	0.15	0.03	0.05	0.00
0.00	0.04	0.02	-0.02	-0.02	0.31	0.51	0.04	0.10	0.05	-0.03	-0.02	0.36	0.05	0.07	0.03	0.20	0.10	0.05	0.00
0.00	0.04	0.08	-0.02	-0.38	0.05	0.07	0.02	0.03	0.08	0.17	-0.02	0.03	0.05	0.03	-0.03	0.20	0.08	0.02	0.00
0.00	0.10	0.11	0.06	0.05	-0.02	0.03	0.07	-0.02	0.08	0.10	0.07	0.02	0.00	-0.06	0.03	0.03	0.17	-0.04	0.00
0.00	0.35	0.16	0.01	0.05	-0.07	-0.07	0.04	0.02	0.04	0.03	0.04	0.13	0.08	-0.05	0.08	-0.04	0.03	0.04	0.00
0.00	0.24	0.07	0.08	0.00	0.04	0.05	0.02	0.00	0.02	0.04	0.16	-0.03	0.17	0.07	0.03	0.07	0.10	-0.02	0.00
0.00	0.57	0.07	0.08	0.00	-0.01	0.00	0.00	0.00	-0.01	-0.04	0.17	-0.02	0.04	0.07	0.21	0.04	0.00	0.02	0.00
0.00	0.68	0.31	0.31	0.00	0.00	0.00	-0.01	-0.01	-0.01	-0.02	0.15	0.22	0.17	0.08	-0.03	0.07	0.05	-0.07	0.00
0.00	0.36	0.31	0.05	0.00	-0.01	-0.01	-0.01	-0.01	-0.01	0.00	0.15	0.11	0.12	0.47	0.11	0.04	0.00	0.02	0.00
0.00	0.44	0.64	0.05	-0.01	-0.06	-0.03	-0.01	-0.01	0.02	0.04	0.05	0.07	0.00	-0.03	0.13	0.05	0.07	0.05	0.00
0.00	0.25	0.21	-0.01	-0.03	-0.02	-0.22	-0.04	-0.02	-0.02	0.05	0.03	0.14	0.14	0.05	0.17	0.08	0.09	0.05	0.00
0.00	0.21	0.08	0.00	-0.05	-0.22	-0.29	-0.07	0.00	-0.21	-0.02	-0.03	0.20	0.08	0.30	0.11	-0.03	0.00	0.05	0.00
0.00	0.08	0.05	-0.03	-0.02	-0.12	-0.12	0.04	0.10	-0.02	-0.02	0.17	0.12	0.13	0.08	0.11	0.16	0.03	0.08	0.00
0.00	0.08	0.03	-0.43	-0.02	-0.03	-0.03	-0.03	0.07	-0.12	0.12	0.05	-0.03	0.10	0.03	0.02	0.05	-0.02	0.08	0.00
0.00	-0.10	-0.21	-0.62	-0.34	-0.05	-0.03	-0.02	0.03	0.04	-0.12	-0.33	0.05	-0.12	0.10	-0.05	0.00	0.05	0.07	0.00
0.00	-0.02	-0.13	0.00	-0.02	-0.02	0.17	0.05	0.03	0.05	0.17	0.52	0.38	0.07	-0.10	0.03	0.02	0.09	0.07	0.00
0.00	-0.02	0.07	0.23	0.38	0.57	0.31	0.12	-0.09	0.03	0.03	-0.02	0.10	0.10	0.31	0.07	0.02	0.02	0.08	0.00
0.00	0.03	0.02	0.00	0.08	-0.02	0.31	0.11	-0.02	0.14	0.31	0.14	0.11	0.11	0.11	0.08	-0.02	0.07	0.08	0.00
0.00	-0.14	-0.02	0.00	0.15	0.02	0.04	0.11	-0.26	0.12	-0.02	0.21	0.05	0.10	0.14	0.05	0.00	0.03	0.10	0.00
0.00	0.08	-0.02	0.17	0.07	-0.02	-0.03	0.07	0.07	-0.02	-0.03	-0.02	0.04	0.22	0.05	0.03	0.02	0.05	0.05	0.00
0.00	0.10	0.00	0.14	0.31	-0.12	0.03	0.10	0.06	0.02	-0.12	0.47	0.12	0.38	0.03	0.02	-0.01	0.01	-0.01	0.00
0.00	0.00	0.02	0.03	-0.08	0.07	0.05	0.10	0.04	0.02	0.00	0.00	0.07	0.00	0.17	0.04	0.00	-0.02	0.02	0.00
0.00	0.05	0.00	0.00	0.06	0.06	0.03	0.00	0.07	0.03	0.01	0.05	0.30	0.14	0.10	0.00	-0.02	-0.02	0.11	0.00
0.00	0.03	0.01	0.00	0.00	0.06	0.04	0.23	0.21	0.14	0.01	0.07	0.08	0.07	0.05	0.00	0.00	0.00	0.00	0.00
0.00	0.09	0.08	0.05	0.00	0.06	0.17	0.15	0.43	0.12	0.00	0.08	0.15	0.07	0.04	0.02	0.00	0.00	0.10	0.00
0.00	0.05	0.03	0.00	-0.03	0.02	0.11	0.10	0.12	0.06	0.02	0.03	0.00	[illegible]	0.07	-0.02	0.08	0.07	0.05	0.00
0.00	0.00	-0.04	-0.04	0.00	0.08	0.46	0.16	0.18	0.03	0.02	0.10	0.47	0.00	-0.03	0.06	0.03	0.06	0.10	0.00
0.00	0.08	0.00	-0.01	-0.01	0.03	0.58	0.23	0.40	0.03	0.00	0.05	0.06	0.07	0.00	0.00	0.07	0.04	0.15	0.00
0.00	0.07	0.03	-0.02	0.03	0.01	0.16	0.17	0.31	0.10	0.00	0.06	0.07	0.13	0.04	0.00	0.03	0.10	0.25	0.00
0.00	0.06	-0.07	-0.20	-0.07	0.00	0.09	0.27	0.18	0.11	0.06	-0.03	0.00	0.08	0.11	0.06	-0.03	0.17	0.05	0.00
0.00	0.06	-0.02	-0.12	-0.12	0.00	0.10	0.10	0.08	0.07	0.30	-0.06	-0.02	0.07	-0.02	0.02	0.00	0.05	0.05	0.00
0.00	0.07	-0.03	-0.02	0.00	-0.11	-0.03	0.07	-0.02	0.00	0.24	0.15	0.00	0.03	0.02	0.06	0.05	0.05	0.00	0.00
0.00	-0.06	0.00	0.30	-0.02	0.08	-0.02	-0.02	-0.02	0.08	0.05	0.13	0.12	0.31	-0.08	0.03	0.03	0.04	0.05	0.00
0.00	0.07	0.12	0.09	-0.06	0.57	0.13	0.13	0.05	0.03	-0.03	0.51	0.51	0.05	-0.10	0.05	-0.02	0.07	0.07	0.00
0.00	0.51	0.12	0.06	0.36	0.51	0.24	-0.10	-0.03	-0.03	0.05	0.14	-0.03	0.10	0.08	0.10	0.02	0.03	0.08	0.00
0.00	0.25	0.06	0.07	0.00	-0.07	0.47	-0.03	0.21	0.24	0.31	-0.06	0.15	0.03	0.05	-0.02	0.07	0.05	0.07	0.00
0.00	0.07	0.07	0.07	0.00	0.11	0.57	0.17	0.31	0.23	0.00	0.00	0.03	0.07	0.07	0.07	0.07	0.07	0.04	0.00
0.00	0.47	0.12	0.14	0.57	0.08	0.17	0.47	0.30	0.24	0.17	0.31	0.17	0.07	0.10	0.06	0.08	0.07	0.02	0.00
0.00	0.11	0.39	0.57	0.31	-0.21	0.06	0.06	0.17	0.11	0.11	0.24	0.47	0.07	0.08	0.04	0.03	0.07	0.13	0.00
0.00	0.18	0.07	0.17	0.00	0.31	0.31	0.13	0.17	0.11	0.17	0.10	0.07	0.13	-0.03	0.08	-0.02	-0.02	-0.49	0.00
0.00	0.04	0.03	0.00	0.56	0.20	0.08	0.04	0.26	0.10	0.04	0.03	0.04	0.04	0.03	0.07	-0.12	0.00	0.02	0.00
0.00	0.02	0.17	0.15	0.08	0.10	0.10	0.05	0.03	0.00	0.02	0.07	0.05	0.06	0.03	0.08	0.03	0.02	0.05	0.00
0.00	0.22	0.27	0.22	-0.10	0.08	0.00	0.05	0.06	0.07	0.10	0.10	0.12	0.07	0.14	0.10	0.06	0.17	0.09	0.00
0.00	0.07	0.15	0.27	0.07	0.06	0.04	0.02	0.06	0.04	0.07	0.38	0.52	0.24	0.05	0.47	0.11	0.14	0.21	0.00
0.00	0.02	0.07	0.04	0.03	0.04	0.06	0.08	0.21	0.04	0.36	0.23	0.17	0.07	0.47	0.35	0.22	0.26	0.38	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

L.3 Near-IR/Red Matrix

1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
1.00	1.00	0.99	1.10	1.09	1.15	1.04	1.00	0.99	1.03	1.05	1.18	1.10	1.05	1.10	1.14	1.15	1.15	1.10	1.00
1.00	1.01	1.01	1.09	1.09	1.20	1.01	0.93	1.00	1.12	1.09	1.15	1.41	1.22	1.04	1.01	1.12	1.10	1.01	1.00
1.00	0.50	0.50	0.49	0.82	0.79	0.97	1.90	1.04	0.95	0.66	0.72	1.40	1.00	0.66	1.22	1.16	1.18	1.12	1.00
1.00	0.90	1.06	1.06	0.74	2.21	1.18	1.09	1.11	0.95	1.00	1.85	1.00	0.89	1.64	1.29	1.36	1.06	1.10	1.00
1.00	1.08	1.05	0.95	0.97	1.90	3.11	1.09	1.23	1.10	0.95	0.97	2.11	1.10	1.16	1.06	1.49	1.23	1.11	1.00
1.00	1.08	1.17	0.97	0.45	1.11	1.15	1.04	1.06	1.18	1.41	0.96	1.06	1.10	1.06	0.95	1.49	1.18	1.05	1.00
1.00	1.22	1.25	1.14	1.10	0.96	1.06	1.16	0.96	1.18	1.22	1.15	1.05	1.01	0.89	1.06	1.05	1.41	0.92	1.00
1.00	2.08	1.37	1.03	1.10	0.87	0.87	1.09	1.04	1.09	1.05	1.09	1.29	1.18	0.90	1.16	0.92	1.05	1.08	1.00
1.00	1.64	1.14	1.17	1.00	1.08	1.11	1.04	0.99	1.03	1.09	1.38	0.95	1.41	1.15	1.06	1.16	1.23	0.96	1.00
1.00	3.65	1.14	1.18	1.00	0.99	0.99	0.99	0.99	0.98	0.92	1.40	0.96	1.09	1.15	1.52	1.09	1.01	1.05	1.00
1.00	5.29	1.90	1.90	1.00	0.99	0.99	0.99	0.98	0.98	0.95	1.36	1.57	1.41	1.18	0.95	1.15	1.10	0.87	1.00
1.00	2.10	1.90	1.10	1.00	0.98	0.98	0.98	0.98	0.98	1.00	1.36	1.26	1.28	2.80	1.26	1.09	1.00	1.04	1.00
1.00	2.54	4.60	1.11	0.98	0.89	0.94	0.97	0.98	1.03	1.08	1.10	1.15	1.00	0.95	1.30	1.10	1.15	1.10	1.00
1.00	1.67	1.52	0.98	0.94	0.95	0.64	0.92	0.96	0.96	1.11	1.05	1.33	1.33	1.11	1.41	1.16	1.20	1.10	1.00
1.00	1.52	1.17	0.99	0.91	0.64	0.55	0.86	1.00	0.66	0.96	0.95	1.49	1.18	1.85	1.26	0.95	1.00	1.11	1.00
1.00	1.17	1.10	0.95	0.96	0.79	0.79	1.09	1.22	0.96	0.97	1.41	1.27	1.30	1.18	1.26	1.38	1.06	1.18	1.00
1.00	1.18	1.06	0.40	0.96	0.95	0.95	0.95	1.16	0.79	1.28	1.11	0.95	1.22	1.06	1.04	1.10	0.95	1.16	1.00
1.00	0.82	0.66	0.23	0.49	0.90	0.95	0.96	1.06	1.09	0.79	0.50	1.11	0.79	1.23	0.91	1.01	1.11	1.16	1.00
1.00	0.96	0.77	1.00	0.97	0.97	1.41	1.10	1.06	1.10	1.41	3.20	2.21	1.15	0.82	1.05	1.05	1.20	1.16	1.00
1.00	0.95	1.15	1.59	2.21	3.70	1.90	1.28	0.83	1.06	1.06	0.96	1.23	1.22	1.90	1.16	1.05	1.05	1.18	1.00
1.00	1.06	1.05	1.01	1.18	0.97	1.90	1.26	0.96	1.33	1.90	1.33	1.26	1.26	1.25	1.16	0.95	1.16	1.18	1.00
1.00	0.76	0.96	1.01	1.36	1.04	1.09	1.26	0.58	1.28	0.96	1.52	1.10	1.22	1.33	1.10	1.00	1.06	1.23	1.00
1.00	1.18	0.96	1.41	1.16	0.96	0.95	1.15	1.15	0.97	0.95	0.96	1.09	1.57	1.10	1.06	1.03	1.10	1.10	1.00
1.00	1.23	1.01	1.33	1.90	0.79	1.05	1.21	1.14	1.05	0.79	2.80	1.28	2.21	1.05	1.03	0.98	1.02	0.99	1.00
1.00	1.01	1.04	1.06	0.86	1.16	1.10	1.21	1.07	1.04	1.00	1.00	1.14	1.00	1.40	1.08	1.00	0.96	1.04	1.00
1.00	1.11	1.00	1.01	1.13	1.14	1.07	1.01	1.15	1.05	1.02	1.10	1.85	1.33	1.22	1.01	0.96	0.95	1.26	1.00
1.00	1.06	1.02	0.99	1.00	1.14	1.09	1.58	1.52	1.33	1.01	1.15	1.17	1.15	1.11	1.00	1.01	1.01	1.01	1.00
1.00	1.20	1.18	1.10	1.00	1.13	1.41	1.34	2.52	1.27	1.01	1.17	1.36	1.14	1.09	1.03	1.00	1.00	1.22	1.00
1.00	1.11	1.07	1.01	0.94	1.05	1.26	1.22	1.27	1.12	1.04	1.06	1.00	[illegible]	1.14	0.96	1.16	1.14	1.11	1.00
1.00	1.00	0.92	0.92	1.00	1.18	2.71	1.39	1.44	1.05	1.05	1.22	2.80	1.00	0.95	1.12	1.05	1.13	1.22	1.00
1.00	1.18	1.00	0.98	0.98	1.07	3.80	1.58	2.33	1.05	1.00	1.10	1.14	1.16	1.00	1.00	1.14	1.09	1.36	1.00
1.00	1.15	1.06	0.97	1.07	1.03	1.39	1.41	1.90	1.21	1.00	1.14	1.15	1.30	1.09	1.01	1.06	1.22	1.66	1.00
1.00	1.13	0.87	0.67	0.88	1.01	1.20	1.72	1.44	1.25	1.13	0.95	1.00	1.17	1.26	1.12	0.95	1.41	1.10	1.00
1.00	1.13	0.95	0.79	0.79	1.00	1.22	1.22	1.16	1.15	1.85	0.89	0.96	1.16	0.95	1.04	1.01	1.11	1.11	1.00
1.00	1.16	0.95	0.96	1.01	0.80	0.95	1.15	0.97	1.00	1.64	1.36	1.00	1.06	1.04	1.13	1.11	1.10	1.00	1.00
1.00	0.89	1.00	1.85	0.95	1.18	0.96	0.96	0.96	1.16	1.10	1.30	1.28	1.90	0.85	1.05	1.06	1.09	1.10	1.00
1.00	1.14	1.29	1.20	0.89	3.70	1.30	1.29	1.10	1.06	0.95	3.11	3.11	1.11	0.82	1.10	0.96	1.14	1.14	1.00
1.00	3.11	1.29	1.12	2.11	3.11	1.64	0.82	0.95	0.95	1.11	1.33	0.95	1.23	1.18	1.23	1.04	1.06	1.16	1.00
1.00	1.67	1.12	1.14	1.00	0.88	2.80	0.95	1.52	1.64	1.90	0.89	1.36	1.05	1.10	0.95	1.16	1.10	1.16	1.00
1.00	1.14	1.14	1.14	1.00	1.25	3.70	1.41	1.90	1.59	1.00	1.00	1.06	1.14	1.16	1.16	1.15	1.15	1.08	1.00
1.00	2.80	1.29	1.33	3.70	1.18	1.40	2.80	1.85	1.64	1.41	1.90	1.40	1.15	1.22	1.13	1.17	1.16	1.05	1.00
1.00	1.26	2.30	3.70	1.90	0.66	1.12	1.12	1.41	1.24	1.25	1.64	2.80	1.15	1.18	1.09	1.07	1.16	1.30	1.00
1.00	1.44	1.16	1.41	1.00	1.90	1.90	1.30	1.41	1.26	1.41	1.23	1.16	1.30	0.95	1.18	0.95	0.97	0.34	1.00
1.00	1.09	1.06	1.00	3.50	1.49	1.16	1.09	1.71	1.22	1.09	1.05	1.08	1.08	1.05	1.16	0.78	1.01	1.05	1.00
1.00	1.04	1.41	1.36	1.16	1.22	1.22	1.10	1.06	1.00	1.05	1.15	1.10	1.13	1.06	1.17	1.07	1.04	1.11	1.00
1.00	1.57	1.75	1.57	0.82	1.18	1.00	1.11	1.13	1.15	1.22	1.21	1.27	1.15	1.33	1.21	1.13	1.40	1.20	1.00
1.00	1.16	1.36	1.72	1.15	1.13	1.08	1.04	1.12	1.08	1.16	2.21	3.20	1.64	1.11	2.80	1.26	1.33	1.52	1.00
1.00	1.05	1.15	1.08	1.06	1.08	1.13	1.18	1.52	1.09	2.11	1.59	1.41	1.16	2.80	2.08	1.57	1.71	2.21	1.00
1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00

L.4 FORTRAN Program: VI and NVI

```
*******************************************************************
*                                                                 *
*  PROGRAM TO FIND THE DIFFERENCE BETWEEN CHANNEL 2 AND 3,        *
*  VI: (CH3 - CH2) AND TO FIND THE NVI: (CH3 - CH2)/(CH3 + CH2)   *
*******************************************************************

      INTEGER COLS, ROWS, I, J
      REAL ARRAY (150,20), ARRAY10 (50,20), ARRAY11 (50, 20)
      REAL SUM

* Files where output will be stored

      OPEN (UNIT = 31, FILE = 'image11')
      OPEN (UNIT = 32, FILE = 'image12')

* Data will be read from external data file

      READ *, ROWS, COLS

* Fill the integer array

      DO 10 I = 1, ROWS
 READ *, (ARRAY (I, J), J = 1, COLS)

10    CONTINUE

* Subtract Channel 2 from Channel 3

      DO 20 I = 1, 50
 DO 30 J = 1, COLS
    ARRAY10 (I, J) = ARRAY (I + 100, J) - ARRAY (I + 50, J)
30       CONTINUE
20    CONTINUE

* Print a title for Matrix1
        PRINT *
        PRINT *, 'VI: CHAN 3 - CHAN 2 OUTPUT'
        PRINT *, '-------------------------------------------------'
        PRINT *
* Display Matrix1 elements, row by row

      DO 60 I = 1, 50
```

```
PRINT 920, (ARRAY10 (I, J), J = 1, COLS)
        WRITE (31, 920) (ARRAY10 (I,J), J = 1, COLS)

60    CONTINUE

* Calculate NVI: (Ch 3 - Ch 2)/(Ch 3 + Ch2)

      DO 120 I = 1, 50
 DO 130 J = 1, COLS
            SUM = ARRAY (I + 100, J) +  ARRAY (I + 50, J)
    ARRAY11 (I, J) = ARRAY10 (I , J) / SUM
 130      CONTINUE
120   CONTINUE

* Print a title for Matrix1
        PRINT *
        PRINT *, 'NVI: (CH 3 - CH 2)/(CH 3 + CH 2) OUTPUT'
        PRINT *, '-------------------------------------------------'
        PRINT *
* Display Matrix1 elements, row by row

      DO 160 I = 1, 50
PRINT 930, (ARRAY11 (I, J), J = 1, COLS)
        WRITE (32, 930) (ARRAY11 (I,J), J = 1, COLS)

160   CONTINUE

920   FORMAT (10X, 50(F4.0,1X))
930   FORMAT (10X, 50(F5.2,1X))

      CLOSE (31)
      CLOSE (32)

      END
```

L.5 FORTRAN Program: Near-IR/Red

```
*****************************************************************************
* Program to compute Ch3/Ch2
*******************************

      INTEGER COLS, ROWS, I, J, COUNT
      REAL RATIO, MAX
      REAL ARRAY (150,50), ARRAY12 (50,20)

* Files where output will be stored

      OPEN (UNIT = 33, FILE = 'image13')
      OPEN (UNIT = 34, FILE = 'image14')

* Data will be read from external data file

      READ *, ROWS, COLS

* Fill the integer array

      DO 10 I = 1, ROWS
 READ *, (ARRAY (I, J), J = 1, COLS)

10    CONTINUE

* Divide Channel 2 by Channel 3

      DO 20 I = 1, 50
 DO 30 J = 1, COLS
    ARRAY12 (I, J) = ARRAY (I + 100, J) / ARRAY (I + 50, J)
30       CONTINUE
20    CONTINUE

* Print a title for Matrix1
        PRINT *
        PRINT *, 'CHAN 3 / CHAN 2 OUTPUT'
        PRINT *, '-----------------------------------------------------'
        PRINT *
* Display Matrix1 elements, row by row

      DO 60 I = 1, 50
PRINT 900, (ARRAY12 (I, J), J = 1, COLS)
        WRITE (33, 910) (ARRAY12 (I,J), J = 1, COLS)
```

```
60    CONTINUE

900   FORMAT (10X, 50(F4.2,1X))
910   FORMAT (1X, 50(F4.2,1X))
920   FORMAT (10X, 50(F4.0,1X))

      CLOSE (33)
      CLOSE (34)

      END
```

Appendix M. *GAMS/OSL Program and Summary Output*

In this appendix is contained the Channel 1–3 program ran to find water regions with area and border constraints of ≥ 24 and ≤ 48, respectively. I have also included a summary of the output from this run. Note this program was not run on a VAX using the GAMS/ZOOM solver. Rather, the program was run on an AIX RS/6000P with the GAMS/OSL solver. This program not run at the Air Force Institute of Technology, but rather at the GAMS Development Corporation (17).

M.1 GAMS/OSL Program

```
GAMS 2.25.056  AIX RS/6000P                    12/07/92 16:01:50  PAGE     1
G e n e r a l   A l g e b r a i c   M o d e l i n g   S y s t e m
C o m p i l a t i o n

  42
  43       SETS
  44               I PIXEL LOCATION / 1*1000 /
  45               K NUMBER OF SUBREGIONS TO  FIND / 1 /
  46               G IMAGING CHANNEL / A, C /;
  47
  48         ALIAS (I,J)
  49         SETS
  50           ADJ (I,J) ADJACENCY MATRIX
  51               /1.  (2,21)
Remaining 999 entries deleted to save space.
1054
1055       TABLE
1056
1057         C(I,G) COST OF EACH PIXEL
1058
1059                    A    C
1060
1061               1  999  999
Remaining 999 entries deleted to save space.
2062
2063       PARAMETERS
2064              M(K) PIXEL SIZE OF KTH SUBREGION
```

```
2065                /1 24/
2066
2067            L(K) BORDER OF KTH SUBREGION
2068                /1 48/ ;
2069
2070
2071
2072       VARIABLES
```



```
2073            Z          TOTAL COST OF ACQUIRED PIXELS
2074            X(I,K)     BINARY VARIABLE
2075            P(I,J,K)   BINARY VARIABLE
2076            N(I,J,K)   BINARY VARIABLE
2077
2078       BINARY VARIABLES X,P,N;
2079
2080       EQUATIONS
2081            OBJ          OBJECTIVE FUNCTION
2082            C1(K)      CONSTRAINT FOR SIZE OF KTH SUBREGION
2083            C2(I,J,K)  CONSTRAINT FOR CONTIGUITY
2084            C3(K)      CONSTRAINT FOR BORDER LENGTH
2085            C4(I,J,K)  CONSTRAINT FOR P&N MUTUAL EXCLUSION
2086            C5(I)        CONSTRAINT FOR SUBREGIONS;
2087
2088       OBJ ..
2089                Z =E= SUM((I,K), 0.1*C(I,"A")*X(I,K)
2090                           + 0.9*C(I,"C")*X(I,K));
2091
2092       C1(K)..
2093              SUM(I,X(I,K)) =G= M(K);
2094
2095       C2(I,J,K) $ADJ(I,J) ..
2096                X(I,K) - X(J,K) - P(I,J,K) + N(I,J,K) =E= 0;
2097
2098       C3(K)..
2099              SUM((I,J) $ADJ(I,J), P(I,J,K) + N(I,J,K)) =L= L(K);
2100
2101       C4(I,J,K) $ADJ(I,J) ..
2102                P(I,J,K) + N(I,J,K) =L= 1;
2103
2104       C5(I) ..
```

```
2105              SUM(K, X(I,K)) =L= 1;
2106
2107       X.FX(I,"1")$(C(I,"A") GT 22) = 0;
2108       X.FX(I,"1")$(C(I,"C") LT 5 OR C(I,"C") GT 200) = 0;
2109
2110
2111       MODEL SUBREGION /ALL/;
2112
2113  *     OPTIONS ITERLIM = 5000, RESLIM = 5000, WORK = 25000;
2114       option iterlim = 100000, reslim=5000;
2115
2116       SOLVE SUBREGION USING MIP MAXIMIZING Z;
2117
2118       DISPLAY X.L,P.L,N.L,Z.L;
2119
2120
2121
2122
```

M.2 GAMS/OSL Summary Output

```
GAMS 2.25.056  AIX RS/6000P                    12/07/92 16:01:50  PAGE    121
G e n e r a l   A l g e b r a i c   M o d e l i n g   S y s t e m
Model Statistics    SOLVE SUBREGION USING MIP FROM LINE 2116



MODEL STATISTICS

BLOCKS OF EQUATIONS       6     SINGLE EQUATIONS     8723
BLOCKS OF VARIABLES       4     SINGLE VARIABLES     8721
NON ZERO ELEMENTS     33881     DISCRETE VARIABLES   7801

GENERATION TIME      =       57.220 SECONDS


EXECUTION TIME       =       57.520 SECONDS        VERID AIX-00-056

STEP SUMMARY:      0.070 STARTUP
                   2.690 COMPILATION
                  57.520 EXECUTION
                   0.450 CLOSEDOWN
```

```
                60.730 TOTAL SECONDS
GAMS 2.25.056  AIX RS/6000P                    12/07/92 16:01:50  PAGE    122
G e n e r a l   A l g e b r a i c   M o d e l i n g   S y s t e m
Solution Report     SOLVE SUBREGION USING MIP FROM LINE 2116


              S O L V E     S U M M A R Y

     MODEL   SUBREGION           OBJECTIVE  Z
     TYPE    MIP                 DIRECTION  MAXIMIZE
     SOLVER  OSL                 FROM LINE  2116

**** SOLVER STATUS     1 NORMAL COMPLETION
**** MODEL STATUS      8 INTEGER SOLUTION
**** OBJECTIVE VALUE             2253.2000

 RESOURCE USAGE, LIMIT        465.120     5000.000
 ITERATION COUNT, LIMIT      2813       100000

 --- GAMS/OSL   2.001


 Work space allocated           --    9.95 Mb


 Relaxed optimum objective value:     2466.1000
 Bound on best integer solution:      2466.1000
 Objective value of this solution:    2253.2000

 Relative gap: .08633 Absolute gap:   212.90000
 Optcr       : .10000 Optca:           .0

 The solution satisfies the termination tolerances

GAMS 2.25.056  AIX RS/6000P                    12/07/92 16:01:50  PAGE    430
G e n e r a l   A l g e b r a i c   M o d e l i n g   S y s t e m
E x e c u t i o n


----   2118 VARIABLE  X.L          BINARY VARIABLE

               1

528         1.000
529         1.000
```

547	1.000
548	1.000
549	1.000
550	1.000
567	1.000
568	1.000
569	1.000
570	1.000
587	1.000
588	1.000
589	1.000
590	1.000
607	1.000
608	1.000
609	1.000
610	1.000
627	1.000
628	1.000
629	1.000
630	1.000
648	1.000
649	1.000

Appendix N. *Network Formulation of B&W Model*

In this appendix, the B&W model is reformulated. This time, the model was set up as a network with side constraints problem.[1] The software used to run this problem was a combination of the following: SAS/OR NETFLOW, GAMS, and Pascal programs (26:125-144). This formulation will attempt to improve upon the performance of the modified model discussed in Chapters 3 and 4.

N.1 Computer Runs

A total of ten computer runs were made using the new formulation of the B&W model. With these runs, I will be able to show if there is an improvement in the model. A summary of the results are contained in Table 36. In this table the nomenclature for the "Title" column needs to be explained. The "S" and the first number indicates the number of subregions to be found. The following numbers represent the size of the subregion(s). For example, S2-1-5 indicates that two subregions are to be found — one of size one and the second of size five. A complete listing of the detailed results are included in this appendix.

N.1.1 Analysis of Computer Runs The new formulation of the B&W model resulted in a measurable increase in performance. The solution time required on the VAX computer was much quicker than the pure GAMS implementation of the B&W model. However, this improvement resulted in another problem. In Table 36 several of the runs contain flows that did not remain integer. When this occurred, the solution was not an optimal one. In the runs that did not maintain unitary flows, the following combinations of the split flow were observed: (.5/.5), (.8/.2), (.6/.2/.2), and (.6/.4). See the output of the computer runs for details.

[1]For an explaination of the network with side constraints, and Operations Research textbook should be consulted. This appendix assumes the reader is familiar with this concept.

Table 36. Results of Ten Runs

Run #	*Title*	*Pixels Selected*	*Total Cost*
1	S1-2	18,19	2
2	S2-1-3	8/14,18,19	9
3	S2-1-4	*	*
4	S2-1-4	18/8,9,14,19	15
5	S2-1-5	*	*
6	S2-1-5	17/8,9,14,18,19	20
7	S2-2-2	14,19/17,18	9
8	s2-2-4	*	*
9	s2-2-4	*	*
10	s2-2-4	8,9/14,17,18,19	20
* Indicates flow did not remain integer			

N.1.2 Conclusion on Network Formulation While the model efficiency in solving the pixel subregion problem was improved, the flow at times did not remaind integer (unitary). More work is required to resolve this deficiency in the network formualtion of the model.

N.1.3 Run #1 (S1-2)

```
                          The SAS System                  20:36 Tuesday, August 25, 1992   1

 'N'  'FROM'   'TO'        'COST'    'CAPAC'      'LO'  'NAME'      'SUPPLY'

  1   ss      a18           1         1         0   SSA18          1
  2   a18     b19           1         1         0   A18B19          .
  3   b19     d1            0         1         0   B19D1           .

     'N'    'DEMAND'     'FLOW'     'FCOST'     'RCOST'  'STATUS'

      1        .          1         1         .   KEY'ARC   BASIC
      2        .          1         1         .   KEY'ARC   BASIC
      3        1          1         0         .   KEY'ARC   BASIC
```

```
1                          The SAS System                  20:36 Tuesday, August 25, 1992

NOTE: Copyright(c) 1989 by SAS Institute Inc., Cary, NC USA.
NOTE: SAS (r) Proprietary Software Release 6.06.01
    Licensed to AIR FORCE INSTITUTE OF TECHNOLOGY, Site 0003357011.

NOTE: Running on VAX Model 6000-420 Serial Number 0B000006.


 Welcome to the new SAS System, Release 6.06.

 This message is seen by users when the NEWS option is specified.
 You can replace this message with your own by editing the NEWS file.

 Changes and enhancements available in SAS Release 6.06 are documented
 in the online Host Help.

1       data noded;
2          input  'node'$ 'sd';
3          cards;

NOTE: The data set WORK.NODED has 2 observations and 2 variables.

6       ;
7       data arcd;
8       input 'from'$ 'to'$ 'cost'  'capac' 'name'$;
9       cards;

NOTE: The data set WORK.ARCD has 42 observations and 5 variables.

52      ;
53      data cond;
54      input
55      ssa7    ssa8    ssa9    ssa12   ssa13   ssa14   ssa17   ssa18   ssa19   a7b8
56      a7b12   a8b9    a8b13   a8b7    a9b14   a9b8    a12b7   a12b13  a12b17  a13b8
57      a13b14  a13b18  a13b12  a14b9   a14b19  a14b13  a17b12  a17b18  a18b13  a18b19
58      a18b17  a19b14  a19b18  b7d1    b8d1    b9d1    b12d1   b13d1   b14d1   b17d1
59      b18d1   b19d1   'type'$ 'rhs';
60      cards;
```

```
NOTE: SAS went to a new line when INPUT statement reached past the end of a line.
NOTE: The data set WORK.COND has 9 observations and 44 variables.

88        ;
89        proc netflow
90          nodedata=noded
91          arcdata=arcd
92          condata=cond
93          conout=solution;
94        run;

NOTE: Number of nodes= 20 .
NOTE: Number of supply nodes= 1 .
NOTE: Number of demand nodes= 1 .
NOTE: Total supply= 1 , total demand= 1 .
NOTE: Number of arcs= 42 .
NOTE: Number of iterations performed (neglecting any constraints)= 23 .
NOTE: Of these, 21 were degenerate.
NOTE: Optimum (neglecting any constraints) found.
NOTE: Minimal total cost= 2 .
```



```
NOTE: Number of ¡= side constraints= 9 .
NOTE: Number of == side constraints= 0 .
NOTE: Number of ¿= side constraints= 0 .
NOTE: Number of arc and nonarc variable side constraint coefficients= 33 .
NOTE: Number of iterations, optimizing with constraints= 1 .
NOTE: Of these, 0 were degenerate.
NOTE: Optimum reached.
NOTE: Minimal total cost= 2 .
NOTE: The data set WORK.SOLUTION has 42 observations and 14 variables.
95         print arcs/nonzero; run;

NOTE: The PROCEDURE NETFLOW printed page 1.

NOTE: SAS Institute Inc., SAS Circle, PO Box 8000, Cary, NC 27512-8000
```

N.1.4 Run #2 (S2-1-3)

```
                          The SAS System                  20:54 Tuesday, August 25, 1992   1

 'N'  'FROM'   'TO'      'COST'    'CAPAC'     'LO'  'NAME'     'SUPPLY'

  1   ss       a14        2         1         0   SSA14          2
  2   ss       a8         5         1         0   SSA8           2
  3   a14      b19        1         1         0   A14B19         .
  4   b19      c18        1         1         0   B19C18         .
  5   a8       d1         0         1         0   A8D1           .
  6   c18      d2         0         1         0   C18D2          .

      'N'    'DEMAND'    'FLOW'    'FCOST'    'RCOST'  'STATUS'

       1        .         1         2         .   KEY'ARC   BASIC
       2        .         1         5         .   NONKEY ARC BASIC
       3        .         1         1         0   UPPERBD NONBASIC
       4        .         1         1         .   KEY'ARC   BASIC
       5        1         1         0         .   KEY'ARC   BASIC
       6        1         1         0         .   KEY'ARC   BASIC


1                         The SAS System                  20:54 Tuesday, August 25, 1992

NOTE: Copyright(c) 1989 by SAS Institute Inc., Cary, NC USA.
NOTE: SAS (r) Proprietary Software Release 6.06.01
    Licensed to AIR FORCE INSTITUTE OF TECHNOLOGY, Site 0003357011.

NOTE: Running on VAX Model 6000-420 Serial Number 0B000006.


 Welcome to the new SAS System, Release 6.06.

 This message is seen by users when the NEWS option is specified.
 You can replace this message with your own by editing the NEWS file.

 Changes and enhancements available in SAS Release 6.06 are documented
 in the online Host Help.

1       data noded;
2          input  'node'$ 'sd';
3          cards;

NOTE: The data set WORK.NODED has 3 observations and 2 variables.

7       ;
8       data arcd;
9       input 'from'$ 'to'$ 'cost'  'capac' 'name'$;
10      cards;

NOTE: The data set WORK.ARCD has 75 observations and 5 variables.

86      ;
87      data cond;
88      input
89      ssa7    ssa8    ssa9    ssa12   ssa13   ssa14   ssa17   ssa18   ssa19   a7b8
```

```
90      a7b12   a8b9    a8b13   a8b7    a9b14   a9b8    a12b7   a12b13  a12b17  a13b8
91      a13b14  a13b18  a13b12  a14b9   a14b19  a14b13  a17b12  a17b18  a18b13  a18b19
92      a18b17  a19b14  a19b18  b7c8    b7c12   b8c9    b8c13   b8c7    b9c14   b9c8
93      b12c7   b12c13  b12c17  b13c8   b13c14  b13c18  b13c12  b14c9   b14c19  b14c13
94      b17c12  b17c18  b18c13  b18c19  b18c17  b19c14  b19c18  a7d1    a8d1    a9d1
95      a12d1   a13d1   a14d1   a17d1   a18d1   a19d1   c7d2    c8d2    c9d2    c12d2
96      c13d2   c14d2   c17d2   c18d2   c19d2   'type'$ 'rhs';
97      cards;

NOTE: SAS went to a new line when INPUT statement reached past the end of a line.
NOTE: The data set WORK.COND has 9 observations and 77 variables.

125     ;
126     proc netflow
127       nodedata=noded
128       arcdata=arcd
129       condata=cond
130       conout=solution;
131     run;

NOTE: Number of nodes= 30 .
NOTE: Number of supply nodes= 1 .
NOTE: Number of demand nodes= 2 .
NOTE: Total supply= 2 , total demand= 2 .
NOTE: Number of arcs= 75 .
NOTE: Number of iterations performed (neglecting any constraints)= 49 .
```



```
NOTE: Of these, 44 were degenerate.
NOTE: Optimum (neglecting any constraints) found.
NOTE: Minimal total cost= 4 .
NOTE: Number of ¡= side constraints= 9 .
NOTE: Number of == side constraints= 0 .
NOTE: Number of ¿= side constraints= 0 .
NOTE: Number of arc and nonarc variable side constraint coefficients= 66 .
NOTE: Number of iterations, optimizing with constraints= 35 .
NOTE: Of these, 32 were degenerate.
NOTE: Optimum reached.
NOTE: Minimal total cost= 9 .
NOTE: The data set WORK.SOLUTION has 75 observations and 14 variables.
132     print arcs/nonzero; run;

NOTE: The PROCEDURE NETFLOW printed page 1.

NOTE: SAS Institute Inc., SAS Circle, PO Box 8000, Cary, NC 27512-8000
```

N.1.5 Run #3 (S2-1-4)

The SAS System 22:08 Tuesday, August 25, 1992 1

'N'	'FROM'	'TO'	'COST'	'CAPAC'	'LO'	'NAME'	'SUPPLY'
1	ss	a14	2	1	0	SSA14	2
2	ss	a18	1	1	0	SSA18	2
3	ss	a8	5	1	0	SSA8	2
4	a18	b17	5	1	0	A18B17	.
5	a14	b19	1	1	0	A14B19	.
6	b19	c14	2	1	0	B19C14	.
7	b17	c18	1	1	0	B17C18	.
8	a8	d1	0	1	0	A8D1	.
9	c18	d17	5	1	0	C18D17	.
10	c14	d19	1	1	0	C14D19	.
11	d17	d2	0	1	0	D17D2	.
12	d19	d2	0	1	0	D19D2	.

'N'	'DEMAND'	'FLOW'	'FCOST'	'RCOST'	'STATUS'
1	.	0.5	1	.	NONKEY ARC BASIC
2	.	0.5	0.5	.	NONKEY ARC BASIC
3	.	1	5	.	KEY'ARC BASIC
4	.	0.5	2.5	.	KEY'ARC BASIC
5	.	0.5	0.5	.	KEY'ARC BASIC
6	.	0.5	1	.	NONKEY ARC BASIC
7	.	0.5	0.5	.	KEY'ARC BASIC
8	1	1	0	.	KEY'ARC BASIC
9	.	0.5	2.5	.	KEY'ARC BASIC
10	.	0.5	0.5	.	KEY'ARC BASIC
11	1	0.5	0	.	KEY'ARC BASIC
12	1	0.5	0	.	KEY'ARC BASIC

1 The SAS System 22:08 Tuesday, August 25, 1992

NOTE: Copyright(c) 1989 by SAS Institute Inc., Cary, NC USA.
NOTE: SAS (r) Proprietary Software Release 6.06.01
Licensed to AIR FORCE INSTITUTE OF TECHNOLOGY, Site 0003357011.

NOTE: Running on VAX Model 6000-420 Serial Number 0B000006.

Welcome to the new SAS System, Release 6.06.

This message is seen by users when the NEWS option is specified.
You can replace this message with your own by editing the NEWS file.

Changes and enhancements available in SAS Release 6.06 are documented
in the online Host Help.

```
1        data noded;
2           input 'node'$ 'sd';
3           cards;
```

```
NOTE: The data set WORK.NODED has 3 observations and 2 variables.

7        ;
8        data arcd;
9        input 'from'$ 'to'$ 'cost'  'capac' 'name'$;
10       cards;

NOTE: The data set WORK.ARCD has 99 observations and 5 variables.

110      ;
111      data cond;
112      input
113      ssa7   ssa8   ssa9   ssa12  ssa13  ssa14  ssa17  ssa18  ssa19  a7b8
114      a7b12  a8b9   a8b13  a8b7   a9b14  a9b8   a12b7  a12b13 a12b17 a13b8
115      a13b14 a13b18 a13b12 a14b9  a14b19 a14b13 a17b12 a17b18 a18b13 a18b19
116      a18b17 a19b14 a19b18 b7c8   b7c12  b8c9   b8c13  b8c7   b9c14  b9c8
117      b12c7  b12c13 b12c17 b13c8  b13c14 b13c18 b13c12 b14c9  b14c19 b14c13
118      b17c12 b17c18 b18c13 b18c19 b18c17 b19c14 b19c18 c7d8   c7d12  c8d9
119      c8d13  c8d7   c9d14  c9d8   c12d7  c12d13 c12d17 c13d8  c13d14 c13d18
120      c13d12 c14d9  c14d19 c14d13 c17d12 c17d18 c18d13 c18d19 c18d17 c19d14
121      c19d18 a7d1   a8d1   a9d1   a12d1  a13d1  a14d1  a17d1  a18d1  a19d1
122      d7d2   d8d2   d9d2   d12d2  d13d2  d14d2  d17d2  d18d2  d19d2  'type'$ 'rhs';
123      cards;

NOTE: SAS went to a new line when INPUT statement reached past the end of a line.
NOTE: The data set WORK.COND has 9 observations and 101 variables.

160      ;
161      proc netflow
162        nodedata=noded
163        arcdata=arcd
164        condata=cond
165        conout=solution;
166      run;

NOTE: Number of nodes= 39 .
NOTE: Number of supply nodes= 1 .
NOTE: Number of demand nodes= 2 .
NOTE: Total supply= 2 , total demand= 2 .
```



```
NOTE: Number of arcs= 99 .
NOTE: Number of iterations performed (neglecting any constraints)= 67 .
NOTE: Of these, 60 were degenerate.
NOTE: Optimum (neglecting any constraints) found.
NOTE: Minimal total cost= 5 .
NOTE: Number of ¡= side constraints= 9 .
NOTE: Number of == side constraints= 0 .
NOTE: Number of ¿= side constraints= 0 .
NOTE: Number of arc and nonarc variable side constraint coefficients= 90 .
NOTE: Number of iterations, optimizing with constraints= 15 .
NOTE: Of these, 12 were degenerate.
NOTE: Optimum reached.
NOTE: Minimal total cost= 14 .
NOTE: The data set WORK.SOLUTION has 99 observations and 14 variables.
167      print arcs/nonzero; run;
```

NOTE: The PROCEDURE NETFLOW printed page 1.

NOTE: SAS Institute Inc., SAS Circle, PO Box 8000, Cary, NC 27512-8000

N.1.6 Run #4 (S2-1-4)

```
                              The SAS System                  22:24 Tuesday, August 25, 1992   1

 'N'  'FROM'   'TO'        'COST'     'CAPAC'       'LO'  'NAME'      'SUPPLY'

  1   ss      a18           1          1          0   SSA18          2
  2   ss      a19           1          1          0   SSA19          2
  3   a19     b14           2          1          0   A19B14          .
  4   b14     c9            6          1          0   B14C9           .
  5   a18     d1            0          1          0   A18D1           .
  6   d8      d2            0          1          0   D8D2            .
  7   c9      d8            5          1          0   C9D8            .

      'N'     'DEMAND'     'FLOW'     'FCOST'     'RCOST'   'STATUS'

       1          .         1          1     -99999997   UPPERBD NONBASIC
       2          .         1          1          .   KEY'ARC    BASIC
       3          .         1          2          .   KEY'ARC    BASIC
       4          .         1          6          .   KEY'ARC    BASIC
       5          1         1          0          .   KEY'ARC    BASIC
       6          1         1          0          .   NONKEY ARC BASIC
       7                    1          5          .   KEY'ARC    BASIC

1                             The SAS System                  22:24 Tuesday, August 25, 1992

NOTE: Copyright(c) 1989 by SAS Institute Inc., Cary, NC USA.
NOTE: SAS (r) Proprietary Software Release 6.06.01
    Licensed to AIR FORCE INSTITUTE OF TECHNOLOGY, Site 0003357011.

NOTE: Running on VAX Model 6000-420 Serial Number 0B000006.


 Welcome to the new SAS System, Release 6.06.

 This message is seen by users when the NEWS option is specified.
 You can replace this message with your own by editing the NEWS file.

 Changes and enhancements available in SAS Release 6.06 are documented
 in the online Host Help.

1       data noded;
2          input  'node'$ 'sd';
3          cards;

NOTE: The data set WORK.NODED has 3 observations and 2 variables.

7       ;
8       data arcd;
9       input 'from'$ 'to'$ 'cost'  'capac' 'name'$;
10      cards;

NOTE: The data set WORK.ARCD has 99 observations and 5 variables.

110     ;
111     data cond;
```

```
112     input
113     ssa7    ssa8    ssa9    ssa12   ssa13   ssa14   ssa17   ssa18   ssa19   a7b8
114     a7b12   a8b9    a8b13   a8b7    a9b14   a9b8    a12b7   a12b13  a12b17  a13b8
115     a13b14  a13b18  a13b12  a14b9   a14b19  a14b13  a17b12  a17b18  a18b13  a18b19
116     a18b17  a19b14  a19b18  b7c8    b7c12   b8c9    b8c13   b8c7    b9c14   b9c8
117     b12c7   b12c13  b12c17  b13c8   b13c14  b13c18  b13c12  b14c9   b14c19  b14c13
118     b17c12  b17c18  b18c13  b18c19  b18c17  b19c14  b19c18  c7d8    c7d12   c8d9
119     c8d13   c8d7    c9d14   c9d8    c12d7   c12d13  c12d17  c13d8   c13d14  c13d18
120     c13d12  c14d9   c14d19  c14d13  c17d12  c17d18  c18d13  c18d19  c18d17  c19d14
121     c19d18  a7d1    a8d1    a9d1    a12d1   a13d1   a14d1   a17d1   a18d1   a19d1
122     d7d2    d8d2    d9d2    d12d2   d13d2   d14d2   d17d2   d18d2   d19d2   'type'$ 'rhs';
123     cards;
```

NOTE: SAS went to a new line when INPUT statement reached past the end of a line.
NOTE: The data set WORK.COND has 10 observations and 101 variables.

```
164     ;
165     proc netflow
166       nodedata=noded
167       arcdata=arcd
168       condata=cond
169       conout=solution;
170     run;
```

NOTE: Number of nodes= 39 .
NOTE: Number of supply nodes= 1 .
NOTE: Number of demand nodes= 2 .
NOTE: Total supply= 2 , total demand= 2 .

NOTE: Number of arcs= 99 .
NOTE: Number of iterations performed (neglecting any constraints)= 67 .
NOTE: Of these, 60 were degenerate.
NOTE: Optimum (neglecting any constraints) found.
NOTE: Minimal total cost= 5 .
NOTE: Number of ¡= side constraints= 9 .
NOTE: Number of == side constraints= 1 .
NOTE: Number of ¿= side constraints= 0 .
NOTE: Number of arc and nonarc variable side constraint coefficients= 91 .
NOTE: Number of iterations, optimizing with constraints= 16 .
NOTE: Of these, 14 were degenerate.
NOTE: Optimum reached.
NOTE: Minimal total cost= 15 .
NOTE: The data set WORK.SOLUTION has 99 observations and 14 variables.

```
171       print arcs/nonzero; run;
```

NOTE: The PROCEDURE NETFLOW printed page 1.

NOTE: SAS Institute Inc., SAS Circle, PO Box 8000, Cary, NC 27512-8000

N.1.7 Run #5 (S2-1-5)

'N'	'FROM'	'TO'	'COST'	'CAPAC'	'LO'	'NAME'	'SUPPLY'
1	ss	a14	2	1	0	SSA14	2
2	ss	a17	5	1	0	SSA17	2
3	ss	a18	1	1	0	SSA18	2
4	ss	a8	5	1	0	SSA8	2
5	ss	a9	6	1	0	SSA9	2
6	a18	b17	5	1	0	A18B17	.
7	a17	b18	1	1	0	A17B18	.
8	a14	b19	1	1	0	A14B19	.
9	a9	b8	5	1	0	A9B8	.
10	a8	b9	6	1	0	A8B9	.
11	b19	c14	2	1	0	B19C14	
12	b17	c18	1	1	0	B17C18	.
13	b18	c19	1	1	0	B18C19	.
14	b9	c8	5	1	0	B9C8	.
15	b8	c9	6	1	0	B8C9	.
16	a17	d1	0	1	0	A17D1	.
17	a18	d1	0	1	0	A18D1	.
18	c19	d14	2	1	0	C19D14	.
19	c14	d19	1	1	0	C14D19	.
20	c18	d19	1	1	0	C18D19	.
21	e8	d2	0	1	0	E8D2	.
22	e9	d2	0	1	0	E9D2	.
23	e14	d2	0	1	0	E14D2	.
24	e19	d2	0	1	0	E19D2	.
25	c9	d8	5	1	0	C9D8	.
26	c8	d9	6	1	0	C8D9	.
27	d19	e14	2	1	0	D19E14	.
28	d14	e19	1	1	0	D14E19	.
29	d9	e8	5	1	0	D9E8	.
30	d8	e9	6	1	0	D8E9	.

'N'	'DEMAND'	'FLOW'	'FCOST'	'RCOST'	'STATUS'
1	.	0.2	0.4	.	KEY'ARC BASIC
2	.	0.8	4	.	KEY'ARC BASIC
3	.	0.6	0.6	.	KEY'ARC BASIC
4	.	0.2	1	.	KEY'ARC BASIC
5	.	0.2	1.2	.	KEY'ARC BASIC
6	.	0.2	1	.	NONKEY ARC BASIC
7	.	0.2	0.2	.	KEY'ARC BASIC
8	.	0.2	0.2	.	NONKEY ARC BASIC
9	.	0.2	1	.	KEY'ARC BASIC
10	.	0.2	1.2	.	KEY'ARC BASIC
11	.	0.2	0.4	.	KEY'ARC BASIC
12	.	0.2	0.2	.	KEY'ARC BASIC
13	.	0.2	0.2	.	KEY'ARC BASIC
14	.	0.2	1	.	KEY'ARC BASIC
15	.	0.2	1.2	.	KEY'ARC BASIC
16	1	0.6	0	.	NONKEY ARC BASIC
17	1	0.4	0	.	KEY'ARC BASIC
18	.	0.2	0.4	.	KEY'ARC BASIC
19	.	0.2	0.2	.	KEY'ARC BASIC

20	.	0.2	0.2	.	KEY'ARC BASIC
21	1	0.2	0	.	NONKEY ARC BASIC
22	1	0.2	0	.	KEY'ARC BASIC
23	1	0.4	0	.	KEY'ARC BASIC

'N'	'DEMAND'	'FLOW'	'FCOST'	'RCOST'	'STATUS'
24	1	0.2	0	.	NONKEY ARC BASIC
25	.	0.2	1	.	NONKEY ARC BASIC
26	.	0.2	1.2	.	KEY'ARC BASIC
27	.	0.4	0.8	.	KEY'ARC BASIC
28	.	0.2	0.2	.	KEY'ARC BASIC
29	.	0.2	1	.	KEY'ARC BASIC
30	.	0.2	1.2	.	KEY'ARC BASIC

NOTE: Copyright(c) 1989 by SAS Institute Inc., Cary, NC USA.
NOTE: SAS (r) Proprietary Software Release 6.06.01
Licensed to AIR FORCE INSTITUTE OF TECHNOLOGY, Site 0003357011.

NOTE: Running on VAX Model 6000-420 Serial Number 0B000006.

Welcome to the new SAS System, Release 6.06.

This message is seen by users when the NEWS option is specified.
You can replace this message with your own by editing the NEWS file.

Changes and enhancements available in SAS Release 6.06 are documented
in the online Host Help.

```
1        data noded;
2           input 'node'$ 'sd';
3           cards;
```

NOTE: The data set WORK.NODED has 3 observations and 2 variables.

```
7        ;
8        data arcd;
9        input 'from'$ 'to'$ 'cost' 'capac' 'name'$;
10       cards;
```

NOTE: The data set WORK.ARCD has 123 observations and 5 variables.

```
134      ;
135      data cond;
136      input
137      ssa7   ssa8   ssa9   ssa12  ssa13  ssa14  ssa17  ssa18  ssa19  a7b8
138      a7b12  a8b9   a8b13  a8b7   a9b14  a9b8   a12b7  a12b13 a12b17 a13b8
139      a13b14 a13b18 a13b12 a14b9  a14b19 a14b13 a17b12 a17b18 a18b13 a18b19
140      a18b17 a19b14 a19b18 b7c8   b7c12  b8c9   b8c13  b8c7   b9c14  b9c8
141      b12c7  b12c13 b12c17 b13c8  b13c14 b13c18 b13c12 b14c9  b14c19 b14c13
142      b17c12 b17c18 b18c13 b18c19 b18c17 b19c14 b19c18 c7d8   c7d12  c8d9
143      c8d13  c8d7   c9d14  c9d8   c12d7  c12d13 c12d17 c13d8  c13d14 c13d18
```

```
144     c13d12  c14d9   c14d19  c14d13  c17d12  c17d18  c18d13  c18d19  c18d17  c19d14
145     c19d18  d7e8    d7e12   d8e9    d8e13   d8e7    d9e14   d9e8    d12e7   d12e13
146     d12e17  d13e8   d13e14  d13e18  d13e12  d14e9   d14e19  d14e13  d17e12  d17e18
147     d18e13  d18e19  d18e17  d19e14  d19e18  a7d1    a8d1    a9d1    a12d1   a13d1
148     a14d1   a17d1   a18d1   a19d1   e7d2    e8d2    e9d2    e12d2   e13d2   e14d2
149     e17d2   e18d2   e19d2   'type'$ 'rhs';
150     cards;
```

NOTE: SAS went to a new line when INPUT statement reached past the end of a line.
NOTE: The data set WORK.COND has 9 observations and 125 variables.

```
196     ;
197     proc netflow
198       nodedata=noded
199       arcdata=arcd
200       condata=cond
201       conout=solution;
202     run;
```

NOTE: Number of nodes= 48 .

NOTE: Number of supply nodes= 1 .
NOTE: Number of demand nodes= 2 .
NOTE: Total supply= 2 , total demand= 2 .
NOTE: Number of arcs= 123 .
NOTE: Number of iterations performed (neglecting any constraints)= 74 .
NOTE: Of these, 66 were degenerate.
NOTE: Optimum (neglecting any constraints) found.
NOTE: Minimal total cost= 6 .
NOTE: Number of ¡= side constraints= 9 .
NOTE: Number of == side constraints= 0 .
NOTE: Number of ¿= side constraints= 0 .
NOTE: Number of arc and nonarc variable side constraint coefficients= 114 .
NOTE: Number of iterations, optimizing with constraints= 37 .
NOTE: Of these, 26 were degenerate.
NOTE: Optimum reached.
NOTE: Minimal total cost= 20 .
NOTE: The data set WORK.SOLUTION has 123 observations and 14 variables.

```
203     print arcs/nonzero; run;
```

NOTE: The PROCEDURE NETFLOW printed pages 1-2.

NOTE: SAS Institute Inc., SAS Circle, PO Box 8000, Cary, NC 27512-8000

N.1.8 Run #6 (S2-1-5)

```
                              The SAS System                  22:42 Tuesday, August 25, 1992   1

 'N'  'FROM'   'TO'        'COST'     'CAPAC'      'LO'   'NAME'      'SUPPLY'

  1   ss       a17          5          1          0   SSA17          2
  2   ss       a18          1          1          0   SSA18          2
  3   a18      b19          1          1          0   A18B19         .
  4   b19      c14          2          1          0   B19C14         .
  5   a17      d1           0          1          0   A17D1          .
  6   e8       d2           0          1          0   E8D2           .
  7   c14      d9           6          1          0   C14D9          .
  8   d9       e8           5          1          0   D9E8           .

      'N'    'DEMAND'     'FLOW'     'FCOST'     'RCOST'   'STATUS'

       1        .          1          5          .   KEY'ARC    BASIC
       2        .          1          1         -3   UPPERBD NONBASIC
       3        .          1          1          .   KEY'ARC    BASIC
       4        .          1          2          .   KEY'ARC    BASIC
       5        1          1          0          0   UPPERBD NONBASIC
       6        1          1          0          .   KEY'ARC    BASIC
       7        .          1          6          .   KEY'ARC    BASIC
       8        .          1          5          0   UPPERBD NONBASIC

1                             The SAS System                  22:42 Tuesday, August 25, 1992

NOTE: Copyright(c) 1989 by SAS Institute Inc., Cary, NC USA.
NOTE: SAS (r) Proprietary Software Release 6.06.01
    Licensed to AIR FORCE INSTITUTE OF TECHNOLOGY, Site 0003357011.

NOTE: Running on VAX Model 6000-420 Serial Number 0B000006.


 Welcome to the new SAS System, Release 6.06.

 This message is seen by users when the NEWS option is specified.
 You can replace this message with your own by editing the NEWS file.

 Changes and enhancements available in SAS Release 6.06 are documented
 in the online Host Help.

1       data noded;
2          input  'node'$ 'sd';
3          cards;

NOTE: The data set WORK.NODED has 3 observations and 2 variables.

7       ;
8       data arcd;
9       input 'from'$ 'to'$ 'cost'  'capac' 'name'$;
10      cards;

NOTE: The data set WORK.ARCD has 123 observations and 5 variables.

134     ;
```

```
135     data cond;
136     input
137     ssa7   ssa8   ssa9   ssa12  ssa13  ssa14  ssa17  ssa18  ssa19  a7b8
138     a7b12  a8b9   a8b13  a8b7   a9b14  a9b8   a12b7  a12b13 a12b17 a13b8
139     a13b14 a13b18 a13b12 a14b9  a14b19 a14b13 a17b12 a17b18 a18b13 a18b19
140     a18b17 a19b14 a19b18 b7c8   b7c12  b8c9   b8c13  b8c7   b9c14  b9c8
141     b12c7  b12c13 b12c17 b13c8  b13c14 b13c18 b13c12 b14c9  b14c19 b14c13
142     b17c12 b17c18 b18c13 b18c19 b18c17 b19c14 b19c18 c7d8   c7d12  c8d9
143     c8d13  c8d7   c9d14  c9d8   c12d7  c12d13 c12d17 c13d8  c13d14 c13d18
144     c13d12 c14d9  c14d19 c14d13 c17d12 c17d18 c18d13 c18d19 c18d17 c19d14
145     c19d18 d7e8   d7e12  d8e9   d8e13  d8e7   d9e14  d9e8   d12e7  d12e13
146     d12e17 d13e8  d13e14 d13e18 d13e12 d14e9  d14e19 d14e13 d17e12 d17e18
147     d18e13 d18e19 d18e17 d19e14 d19e18 a7d1   a8d1   a9d1   a12d1  a13d1
148     a14d1  a17d1  a18d1  a19d1  e7d2   e8d2   e9d2   e12d2  e13d2  e14d2
149     e17d2  e18d2  e19d2  'type'$ 'rhs';
150     cards;
```

NOTE: SAS went to a new line when INPUT statement reached past the end of a line.
NOTE: The data set WORK.COND has 10 observations and 125 variables.

```
201     ;
202     proc netflow
203       nodedata=noded
204       arcdata=arcd
205       condata=cond
206       conout=solution;
207     run;
```

NOTE: Number of nodes= 48 .

NOTE: Number of supply nodes= 1 .
NOTE: Number of demand nodes= 2 .
NOTE: Total supply= 2 , total demand= 2 .
NOTE: Number of arcs= 123 .
NOTE: Number of iterations performed (neglecting any constraints)= 74 .
NOTE: Of these, 66 were degenerate.
NOTE: Optimum (neglecting any constraints) found.
NOTE: Minimal total cost= 6 .
NOTE: Number of ¡= side constraints= 9 .
NOTE: Number of ==≈ side constraints= 1 .
NOTE: Number of ¿= side constraints= 0 .
NOTE: Number of arc and nonarc variable side constraint coefficients= 116 .
NOTE: Number of iterations, optimizing with constraints= 73 .
NOTE: Of these, 64 were degenerate.
NOTE: Optimum reached.
NOTE: Minimal total cost= 20 .
NOTE: The data set WORK.SOLUTION has 123 observations and 14 variables.

```
208       print arcs/nonzero; run;
```

NOTE: The PROCEDURE NETFLOW printed page 1.

NOTE: SAS Institute Inc., SAS Circle, PO Box 8000, Cary, NC 27512-8000

N.1.9 Run #7 (S2-2-2)

```
1                                  The SAS System                  22:52 Tuesday, August 25, 1992

        'N'  'FROM'   'TO'        'COST'    'CAPAC'      'LO'  'NAME'      'SUPPLY'

         1   ss      a14          2         1          0  SSA14          2
         2   ss      a17          5         1          0  SSA17          2
         3   a17     b18          1         1          0  A17B18          .
         4   a14     b19          1         1          0  A14B19          .
         5   b18     d1           0         1          0  B18D1          .
         6   b19     d2           0         1          0  B19D2          .

            'N'    'DEMAND'    'FLOW'    'FCOST'    'RCOST'  'STATUS'

             1        .         1         2          .  KEY'ARC   BASIC
             2        .         1         5          .  KEY'ARC   BASIC
             3        .         1         1          .  NONKEY ARC BASIC
             4        .         1         1          .  KEY'ARC   BASIC
             5        1         1         0          .  KEY'ARC   BASIC
             6        1         1         0          0  UPPERBD NONBASIC

1                                  The SAS System                  22:52 Tuesday, August 25, 1992

NOTE: Copyright(c) 1989 by SAS Institute Inc., Cary, NC USA.
NOTE: SAS (r) Proprietary Software Release 6.06.01
    Licensed to AIR FORCE INSTITUTE OF TECHNOLOGY, Site 0003357011.

NOTE: Running on VAX Model 6000-420 Serial Number 0B000006.


 Welcome to the new SAS System, Release 6.06.

 This message is seen by users when the NEWS option is specified.
 You can replace this message with your own by editing the NEWS file.

 Changes and enhancements available in SAS Release 6.06 are documented
 in the online Host Help.

1       data noded;
2          input  'node'$ 'sd';
3          cards;

NOTE: The data set WORK.NODED has 3 observations and 2 variables.

7       ;
8       data arcd;
9       input 'from'$ 'to'$ 'cost'  'capac' 'name'$;
10      cards;

NOTE: The data set WORK.ARCD has 51 observations and 5 variables.

62      ;
63      data cond;
64      input
65      ssa7    ssa8    ssa9    ssa12   ssa13   ssa14   ssa17   ssa18   ssa19   a7b8
66      a7b12   a8b9    a8b13   a8b7    a9b14   a9b8    a12b7   a12b13  a12b17  a13b8
```

```
67      a13b14  a13b18  a13b12  a14b9   a14b19  a14b13  a17b12  a17b18  a18b13  a18b19
68      a18b17  a19b14  a19b18  b7d1    b8d1    b9d1    b12d1   b13d1   b14d1   b17d1
69      b18d1   b19d1   b7d2    b8d2    b9d2    b12d2   b13d2   b14d2   b17d2   b18d2
70      b19d2   'type'$ 'rhs';
71      cards;
```

NOTE: SAS went to a new line when INPUT statement reached past the end of a line.
NOTE: The data set WORK.COND has 9 observations and 53 variables.

```
99      ;
100     proc netflow
101       nodedata=noded
102       arcdata=arcd
103       condata=cond
104       conout=solution;
105     run;
```

NOTE: Number of nodes= 21 .
NOTE: Number of supply nodes= 1 .
NOTE: Number of demand nodes= 2 .
NOTE: Total supply= 2 , total demand= 2 .
NOTE: Number of arcs= 51 .
NOTE: Number of iterations performed (neglecting any constraints)= 33 .
NOTE: Of these, 29 were degenerate.
NOTE: Optimum (neglecting any constraints) found.

NOTE: Minimal total cost= 4 .
NOTE: Number of ¡= side constraints= 9 .
NOTE: Number of == side constraints= 0 .
NOTE: Number of ¿= side constraints= 0 .
NOTE: Number of arc and nonarc variable side constraint coefficients= 42 .
NOTE: Number of iterations, optimizing with constraints= 10 .
NOTE: Of these, 6 were degenerate.
NOTE: Optimum reached.
NOTE: Minimal total cost= 9 .
NOTE: The data set WORK.SOLUTION has 51 observations and 14 variables.

```
106     print arcs/nonzero; run;
```

NOTE: The PROCEDURE NETFLOW printed page 1.

NOTE: SAS Institute Inc., SAS Circle, PO Box 8000, Cary, NC 27512-8000

N.1.10 Run #8 (S2-2-4)

'N'	'FROM'	'TO'	'COST'	'CAPAC'	'LO'	'NAME'	'SUPPLY'
1	ss	a14	2	1	0	SSA14	2
2	ss	a17	5	1	0	SSA17	2
3	ss	a8	5	1	0	SSA8	2
4	a17	b18	1	1	0	A17B18	.
5	a14	b19	1	1	0	A14B19	.
6	a8	b9	6	1	0	A8B9	.
7	b18	c17	5	1	0	B18C17	.
8	b9	c8	5	1	0	B9C8	.
9	b19	d1	0	1	0	B19D1	.
10	c17	d18	1	1	0	C17D18	.
11	d9	d2	0	1	0	D9D2	.
12	d18	d2	0	1	0	D18D2	.
13	c8	d9	6	1	0	C8D9	.

'N'	'DEMAND'	'FLOW'	'FCOST'	'RCOST'	'STATUS'
1	.	1	2	.	KEY'ARC BASIC
2	.	0.5	2.5	.	KEY'ARC BASIC
3	.	0.5	2.5	.	KEY'ARC BASIC
4	.	0.5	0.5	.	NONKEY ARC BASIC
5	.	1	1	.	KEY'ARC BASIC
6	.	0.5	3	.	NONKEY ARC BASIC
7	.	0.5	2.5	.	KEY'ARC BASIC
8	.	0.5	2.5	.	NONKEY ARC BASIC
9	1	1	0	.	NONKEY ARC BASIC
10	.	0.5	0.5	.	KEY'ARC BASIC
11	1	0.5	0	.	KEY'ARC BASIC
12	1	0.5	0	.	KEY'ARC BASIC
13	.	0.5	3	.	KEY'ARC BASIC

NOTE: Copyright(c) 1989 by SAS Institute Inc., Cary, NC USA.
NOTE: SAS (r) Proprietary Software Release 6.06.01
Licensed to AIR FORCE INSTITUTE OF TECHNOLOGY, Site 0003357011.

NOTE: Running on VAX Model 6000-420 Serial Number 0B000006.

Welcome to the new SAS System, Release 6.06.

This message is seen by users when the NEWS option is specified.
You can replace this message with your own by editing the NEWS file.

Changes and enhancements available in SAS Release 6.06 are documented
in the online Host Help.

```
1        data noded;
2           input 'node'$ 'sd';
3           cards;
```

```
NOTE: The data set WORK.NODED has 3 observations and 2 variables.

7        ;
8        data arcd;
9        input 'from'$ 'to'$ 'cost' 'capac' 'name'$;
10       cards;

NOTE: The data set WORK.ARCD has 99 observations and 5 variables.

110      ;
111      data cond;
112      input
113      ssa7   ssa8   ssa9   ssa12  ssa13  ssa14  ssa17  ssa18  ssa19  a7b8
114      a7b12  a8b9   a8b13  a8b7   a9b14  a9b8   a12b7  a12b13 a12b17 a13b8
115      a13b14 a13b18 a13b12 a14b9  a14b19 a14b13 a17b12 a17b18 a18b13 a18b19
116      a18b17 a19b14 a19b18 b7c8   b7c12  b8c9   b8c13  b8c7   b9c14  b9c8
117      b12c7  b12c13 b12c17 b13c8  b13c14 b13c18 b13c12 b14c9  b14c19 b14c13
118      b17c12 b17c18 b18c13 b18c19 b18c17 b19c14 b19c18 c7d8   c7d12  c8d9
119      c8d13  c8d7   c9d14  c9d8   c12d7  c12d13 c12d17 c13d8  c13d14 c13d18
120      c13d12 c14d9  c14d19 c14d13 c17d12 c17d18 c18d13 c18d19 c18d17 c19d14
121      c19d18 b7d1   b8d1   b9d1   b12d1  b13d1  b14d1  b17d1  b18d1  b19d1
122      d7d2   d8d2   d9d2   d12d2  d13d2  d14d2  d17d2  d18d2  d19d2  'type'$ 'rhs';
123      cards;

NOTE: SAS went to a new line when INPUT statement reached past the end of a line.
NOTE: The data set WORK.COND has 9 observations and 101 variables.

160      ;
161      proc netflow
162        nodedata=noded
163        arcdata=arcd
164        condata=cond
165        conout=solution;
166      run;

NOTE: Number of nodes= 39 .
NOTE: Number of supply nodes= 1 .
NOTE: Number of demand nodes= 2 .
NOTE: Total supply= 2 , total demand= 2 .
```



```
NOTE: Number of arcs= 99 .
NOTE: Number of iterations performed (neglecting any constraints)= 70 .
NOTE: Of these, 64 were degenerate.
NOTE: Optimum (neglecting any constraints) found.
NOTE: Minimal total cost= 6 .
NOTE: Number of ¡= side constraints= 9 .
NOTE: Number of == side constraints= 0 .
NOTE: Number of ¿= side constraints= 0 .
NOTE: Number of arc and nonarc variable side constraint coefficients= 90 .
NOTE: Number of iterations, optimizing with constraints= 30 .
NOTE: Of these, 20 were degenerate.
NOTE: Optimum reached.
NOTE: Minimal total cost= 20 .
NOTE: The data set WORK.SOLUTION has 99 observations and 14 variables.
167      print arcs/nonzero; run;
```

NOTE: The PROCEDURE NETFLOW printed page 1.

NOTE: SAS Institute Inc., SAS Circle, PO Box 8000, Cary, NC 27512-8000

N.1.11 Run #9 (S2-2-4)

'N'	'FROM'	'TO'	'COST'	'CAPAC'	'LO'	'NAME'	'SUPPLY'
1	ss	a14	2	1	0	SSA14	2
2	ss	a18	1	1	0	SSA18	2
3	ss	a8	5	1	0	SSA8	2
4	a18	b17	5	1	0	A18B17	.
5	a14	b19	1	1	0	A14B19	.
6	a8	b9	6	1	0	A8B9	.
7	b19	c14	2	1	0	B19C14	.
8	b17	c18	1	1	0	B17C18	.
9	b9	d1	0	1	0	B9D1	.
10	c18	d17	5	1	0	C18D17	.
11	c14	d19	1	1	0	C14D19	.
12	d17	d2	0	1	0	D17D2	.
13	d19	d2	0	1	0	D19D2	.

'N'	'DEMAND'	'FLOW'	'FCOST'	'RCOST'	'STATUS'
1	.	0.5	1	.	NONKEY ARC BASIC
2	.	0.5	0.5	.	KEY'ARC BASIC
3	.	1	5	-99999999	UPPERBD NONBASIC
4	.	0.5	2.5	.	NONKEY ARC BASIC
5	.	0.5	0.5	.	KEY'ARC BASIC
6	.	1	6	0	UPPERBD NONBASIC
7	.	0.5	1	.	KEY'ARC BASIC
8	.	0.5	0.5	.	KEY'ARC BASIC
9	1	1	0	.	KEY'ARC BASIC
10	.	0.5	2.5	.	NONKEY ARC BASIC
11	.	0.5	0.5	.	KEY'ARC BASIC
12	1	0.5	0	.	KEY'ARC BASIC
13	1	0.5	0	.	KEY'ARC BASIC

NOTE: Copyright(c) 1989 by SAS Institute Inc., Cary, NC USA.
NOTE: SAS (r) Proprietary Software Release 6.06.01
Licensed to AIR FORCE INSTITUTE OF TECHNOLOGY, Site 0003357011.

NOTE: Running on VAX Model 6000-420 Serial Number 0B000006.

Welcome to the new SAS System, Release 6.06.

This message is seen by users when the NEWS option is specified.
You can replace this message with your own by editing the NEWS file.

Changes and enhancements available in SAS Release 6.06 are documented
in the online Host Help.

```
1        data noded;
2           input 'node'$ 'sd';
3           cards;
```

```
NOTE: The data set WORK.NODED has 3 observations and 2 variables.

7       ;
8       data arcd;
9       input 'from'$ 'to'$ 'cost' 'capac' 'name'$;
10      cards;

NOTE: The data set WORK.ARCD has 99 observations and 5 variables.

110     ;
111     data cond;
112     input
113     ssa7   ssa8   ssa9   ssa12  ssa13  ssa14  ssa17  ssa18  ssa19  a7b8
114     a7b12  a8b9   a8b13  a8b7   a9b14  a9b8   a12b7  a12b13 a12b17 a13b8
115     a13b14 a13b18 a13b12 a14b9  a14b19 a14b13 a17b12 a17b18 a18b13 a18b19
116     a18b17 a19b14 a19b18 b7c8   b7c12  b8c9   b8c13  b8c7   b9c14  b9c8
117     b12c7  b12c13 b12c17 b13c8  b13c14 b13c18 b13c12 b14c9  b14c19 b14c13
118     b17c12 b17c18 b18c13 b18c19 b18c17 b19c14 b19c18 c7d8   c7d12  c8d9
119     c8d13  c8d7   c9d14  c9d8   c12d7  c12d13 c12d17 c13d8  c13d14 c13d18
120     c13d12 c14d9  c14d19 c14d13 c17d12 c17d18 c18d13 c18d19 c18d17 c19d14
121     c19d18 b7d1   b8d1   b9d1   b12d1  b13d1  b14d1  b17d1  b18d1  b19d1
122     d7d2   d8d2   d9d2   d12d2  d13d2  d14d2  d17d2  d18d2  d19d2  'type'$ 'rhs';
123     cards;

NOTE: SAS went to a new line when INPUT statement reached past the end of a line.
NOTE: The data set WORK.COND has 10 observations and 101 variables.

164     ;
165     proc netflow
166       nodedata=noded
167       arcdata=arcd
168       condata=cond
169       conout=solution;
170     run;

NOTE: Number of nodes= 39 .
NOTE: Number of supply nodes= 1 .
NOTE: Number of demand nodes= 2 .
NOTE: Total supply= 2 , total demand= 2 .
```



```
NOTE: Number of arcs= 99 .
NOTE: Number of iterations performed (neglecting any constraints)= 70 .
NOTE: Of these, 64 were degenerate.
NOTE: Optimum (neglecting any constraints) found.
NOTE: Minimal total cost= 6 .
NOTE: Number of ¡= side constraints= 9 .
NOTE: Number of == side constraints= 1 .
NOTE: Number of ¿= side constraints= 0 .
NOTE: Number of arc and nonarc variable side constraint coefficients= 91 .
NOTE: Number of iterations, optimizing with constraints= 38 .
NOTE: Of these, 32 were degenerate.
NOTE: Optimum reached.
NOTE: Minimal total cost= 20 .
NOTE: The data set WORK.SOLUTION has 99 observations and 14 variables.
171       print arcs/nonzero; run;
```

NOTE: The PROCEDURE NETFLOW printed page 1.

NOTE: SAS Institute Inc., SAS Circle, PO Box 8000, Cary, NC 27512-8000

N.1.12 Run #10 (S2-2-4)

```
                              The SAS System                  23:51 Tuesday, August 25, 1992   1

 'N'  'FROM'   'TO'        'COST'     'CAPAC'       'LO'  'NAME'       'SUPPLY'

  1   ss       a14          2          1          0   SSA14            2
  2   ss       a8           5          1          0   SSA8             2
  3   a14      b19          1          1          0   A14B19           .
  4   a8       b9           6          1          0   A8B9             .
  5   b19      c18          1          1          0   B19C18           .
  6   b9       d1           0          1          0   B9D1             .
  7   c18      d17          5          1          0   C18D17           .
  8   d17      d2           0          1          0   D17D2            .

      'N'    'DEMAND'     'FLOW'     'FCOST'      'RCOST'   'STATUS'

       1        .          1          2          .   KEY'ARC   BASIC
       2        .          1          5          .   KEY'ARC   BASIC
       3        .          1          1          .   KEY'ARC   BASIC
       4        .          1          6   3.885781E-16   UPPERBD NONBASIC
       5        .          1          1          .   KEY'ARC   BASIC
       6        1          1          0          .   KEY'ARC   BASIC
       7        .          1          5   3.330669E-16   UPPERBD NONBASIC
       8        1          1          0          .   KEY'ARC   BASIC

1                             The SAS System                  23:51 Tuesday, August 25, 1992

NOTE: Copyright(c) 1989 by SAS Institute Inc., Cary, NC USA.
NOTE: SAS (r) Proprietary Software Release 6.06.01
    Licensed to AIR FORCE INSTITUTE OF TECHNOLOGY, Site 0003357011.

NOTE: Running on VAX Model 6000-420 Serial Number 0B000006.


 Welcome to the new SAS System, Release 6.06.

 This message is seen by users when the NEWS option is specified.
 You can replace this message with your own by editing the NEWS file.

 Changes and enhancements available in SAS Release 6.06 are documented
 in the online Host Help.

1       data noded;
2          input 'node'$ 'sd';
3          cards;

NOTE: The data set WORK.NODED has 3 observations and 2 variables.

7       ;
8       data arcd;
9       input 'from'$ 'to'$ 'cost' 'capac' 'name'$;
10      cards;

NOTE: The data set WORK.ARCD has 99 observations and 5 variables.

110     ;
```

```
111     data cond;
112     input
113     ssa7   ssa8   ssa9   ssa12  ssa13  ssa14  ssa17  ssa18  ssa19  a7b8
114     a7b12  a8b9   a8b13  a8b7   a9b14  a9b8   a12b7  a12b13 a12b17 a13b8
115     a13b14 a13b18 a13b12 a14b9  a14b19 a14b13 a17b12 a17b18 a18b13 a18b19
116     a18b17 a19b14 a19b18 b7c8   b7c12  b8c9   b8c13  b8c7   b9c14  b9c8
117     b12c7  b12c13 b12c17 b13c8  b13c14 b13c18 b13c12 b14c9  b14c19 b14c13
118     b17c12 b17c18 b18c13 b18c19 b18c17 b19c14 b19c18 c7d8   c7d12  c8d9
119     c8d13  c8d7   c9d14  c9d8   c12d7  c12d13 c12d17 c13d8  c13d14 c13d18
120     c13d12 c14d9  c14d19 c14d13 c17d12 c17d18 c18d13 c18d19 c18d17 c19d14
121     c19d18 b7d1   b8d1   b9d1   b12d1  b13d1  b14d1  b17d1  b18d1  b19d1
122     d7d2   d8d2   d9d2   d12d2  d13d2  d14d2  d17d2  d18d2  d19d2  'type'$ 'rhs';
123     cards;
```

NOTE: SAS went to a new line when INPUT statement reached past the end of a line.
NOTE: The data set WORK.COND has 10 observations and 101 variables.

```
164     ;
165     proc netflow
166       nodedata=noded
167       arcdata=arcd
168       condata=cond
169       conout=solution;
170     run;
```

NOTE: Number of nodes= 39 .
NOTE: Number of supply nodes= 1 .
NOTE: Number of demand nodes= 2 .
NOTE: Total supply= 2 , total demand= 2 .

NOTE: Number of arcs= 99 .
NOTE: Number of iterations performed (neglecting any constraints)= 70 .
NOTE: Of these, 64 were degenerate.
NOTE: Optimum (neglecting any constraints) found.
NOTE: Minimal total cost= 6 .
NOTE: Number of ¡= side constraints= 9 .
NOTE: Number of == side constraints= 1 .
NOTE: Number of ¿= side constraints= 0 .
NOTE: Number of arc and nonarc variable side constraint coefficients= 92 .
NOTE: Number of iterations, optimizing with constraints= 29 .
NOTE: Of these, 24 were degenerate.
NOTE: Optimum reached.
NOTE: Minimal total cost= 20 .
NOTE: The data set WORK.SOLUTION has 99 observations and 14 variables.

```
171     print arcs/nonzero; run;
```

NOTE: The PROCEDURE NETFLOW printed page 1.

NOTE: SAS Institute Inc., SAS Circle, PO Box 8000, Cary, NC 27512-8000

Bibliography

1. Benabdullah, Salah and Jeff R. Wright. "Multiple Subregion Allocation Models." Unpublished manuscript, 1990.

2. Brooke, Anthony and others. *GAMS: A User's Guide.* Redwood City, California: The Scientific Press, 1988.

3. Cipperly, George E. Personal Interviews. Instructor of Physics, Engineering Physics Department, Air Force Institute of Technology (AU), Wright-Patterson AFB, OH, Fall 1992.

4. Chan, Yupo. Class handout distributed in OPER7.61, Multiple Criteria Decision Making, Air Force Institute of Technology (AU), Wright-Patterson AFB, OH, Fall 1992.

5. Chan, Dr Yupo. *Multicriteria Decision Making: A Tutorial.* Tutorial given at the 1991 Military Operations Research Society Symposium at West Point, N.Y., Working Papers, Air Force Institute of Technology (AU), Wright-Patterson AFB, OH, August 1991.

6. Chan, Yupo. "Remote Sensing and Geographic Information Systems: Applications to Spatial-Temporal Analysis." Unpublished manuscript, 1992.

7. Chan, Yupo and Thomas S. Kelso. Class handout in OPER6.37, Analysis and Modelling of Spatial-Temporal Information, Air Force Institute of Technology (AU), Wright-Patterson AFB, OH, Summer 1992.

8. Courtois, Michel and Gilbert Weill. "The SPOT Satellite System," *Monitoring Earth's Ocean, Land, And Atmosphere From Space — Sensors, Systems, and Applications,* Volume 97, edited by Abraham Schnapf. New York: American Institute of Aeronautics and Astronautics, Inc., 1985.

9. Department of the Navy, Office of Naval Research. *The Infrared Handbook,* edited by William L. Wolfe and George J. Zissis. Contracts: N00014-73-A-0321-0002, N00014-74-C-285, N00014-76-C-0607, N00014-77-C-0125 with The Infrared Information and Analysis (IRIA) Center, Environmental Research Institute of Michigan. Washington: Government Printing Office, 1978.

10. de Neufville, Richard. *Applied Systems Analysis.* New York: McGraw Hill Publishing Company, 1990.

11. Gonzalez, Rafael C. and Richard E. Woods. *Digital Image Processing.* Reading, Massachusetts: Addison–Wesley Publishing Company, Inc., 1992.

12. Harlow, C.A. and others. "On Radiographic Image Analysis," Chapter 3, *Digital Picture Analysis.* A. Rosenfeld editor, Berlin: Springer-Verlang, 1976.

13. Harris, Ray. *Satellite Remote Sensing, An Introduction.* London: Routledge & Kegan Paul, 1987.

14. Hord, R. Michael. *Digital Image Processing of Remotely Sensed Data.* New York: Academic Press, 1982.

15. Jensen, John R. *Introductory Digital Image Processing - A Remote Sensing Perspective.* Englewood Cliffs, New Jersey: Prentice - Hall, 1986.

16. Kalvelagen, Erwin, Programmer. Telephone interview. GAMS Development Corporation, Washington DC, 16 November 1992.

17. Kalvelagen, Erwin, Programmer. Telephone interview and Electronic Mail Correspondence. GAMS Development Corporation, Washington DC, 7-9 Dec 1992.

18. Kelly, Glen G., Cartographer. Telephone interview. U.S. Geological Survey, EROS Data Center, Science and Applications Branch, Sioux Falls, South Dakota, 10 and 16 November 1992.

19. Kelso, Dr. Thomas S. Electronic Mail Correspondence. Air Force Institute of Technology/Operational Sciences Department, Wright-Patterson AFB, OH, 5 and 11 November 1992.

20. Lillesand, Thomas M. and Ralph W. Kiefer. *Remote Sensing and Image Interpretation.* New York: John Wiley & Sons, 1979.

21. Lillesand, Thomas M. and Ralph W. Kiefer. *Remote Sensing and Image Interpretation* (2nd Edition). New York: John Wiley & Sons, 1987.

22. *Manual of Remote Sensing.* Volume 1, editor-in-chief Robert G. Reeves. Falls Church, Virginia: American Society of Photogrammetry, 1975.

23. Mather, Paul M. *Computer Processing of Remotely-Sensed Images, An Introduction.* Chichester, England: John Wiley & Sons, 1987.

24. Moik, Johannes G. *Digital Processing of Remotely Sensed Images.* National Aeronautics & Space Administration - Scientific & Technical Branch, NASA SP-431, Washington: U.S. Government Printing Office, 1980.

25. Perry, C.R. and L.F. Lautenschlager. "Functional Equivalence of Spectral Vegetation Indices," *Remote Sensing of Environment,* 14:169-182 (January 1984).

26. Reed, Thomas G. *Binary Programming Models of Spatial Pattern Recognition: Applications in Remote Sensing Image Analysis.* MS thesis, Air Force Institute of Technology, December 1991. DTIC # AD-A243 797.

27. Richardson, A.J. and C.L. Weigand. "Distinguishing Vegetation from Soil Background Information," *Remote Sensing of Environment,* 8:307-312 (December 1979).

28. Rosenfeld, Azriel and Avinash C. Kak. *Digital Picture Processing, 2nd Edition, Volume 2.* New York: Academic Press, 1982.

29. Sabins, Floyd F. *Remote Sensing* San Francisco: W. H. Freeman and Company, 1978.

30. Sabins, Floyd F., Jr. *Remote Sensing Principles and Interpretation* (2nd Edition). New York: W. H. Freeman & Company, 1978.

31. Schrange, Linus. *User's Manual for Linear, Integer and Quadratic Programming with LINDO* (3rd Edition), The Scientific Press: Redwood City, CA. 1987.

32. Stares, Paul B. *The Militarization of Space.* Ithaca, New York: Cornell University Press, 1985.

33. Steuer, Ralph E. *Multiple Criteria Optimization: Theory, Computation, and Application.* New York: John Wiley & Sons, 1986.

34. Szekielda, Karl-Heinz. *Satellite Monitoring Of The Earth.* New York: John Wiley & Sons, 1988.

35. Thomas, Ian L. and others. *Classification of Remotely Sensed Images.* Bristol, England: Adam Hilger, 1987.

36. Tou, J.T. and R.C. Gonzalez. *Pattern Recognition Principles.* Reading, Massachusetts: Addison-Wesley, 1974.

37. "Washington Inside the Beltway," *National Geographic Magazine*, Map Insert, January 1983.

Vita

Captain John M. Amrine was born on 31 January 1960 in Honolulu, Hawaii. He graduated from Little Rock Central High in Little Rock, Arkansas in May of 1978. Following graduation from high school, he accepted an appointment to the United States Air Force Academy. On 2 June 1982, he graduated from the Academy with a Bachelor of Science degree in General Science and also received his commission into the United States Air Force. His career in space began in 1983 at the Satellite Control Facility in Sunnyvale, California. At Sunnyvale AFS, he was assigned to Secretary of the Air Force/Special Projects, Operations Detachment 4. He served first as a Space Operations Officer and finally as Chief, Performance Assessment Branch. Following five years in Sunnyvale, he was assigned to the Pentagon where he continued to support Secretary of the Air Force/Special Projects. At the Pentagon, he served in a joint space program office. During his three year tour, he served as a Senior Data Systems Analyst and as a Space Systems Program Manager. He entered the School of Engineering, Air Force Institute of Technology, in May 1991 in pursuit of a Master's degree in Space Operations.

Permanent address: 1919 Rainwood Cove
Little Rock, Arkansas 72212