MULTIPLE AND ALTERNATE OPTIMA OF LP PROBLEMS VIA RECURSIVE MILP: A MATLAB IMPLEMENTATION

by

Ridade Sayın

B.S., Chemical Engineering, Boğaziçi University, 2010

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
requirements for the degree of
Master of Science

Graduate Program in Chemical Engineering
Boğaziçi University
2012

MULTIPLE AND ALTERNATE OPTIMA OF LP PROBLEMS VIA RECURSIVE MILP: A MATLAB IMPLEMENTATION

APPROVED BY:

Prof. Uğur Akman (Thesis Supervisor)	.....
Assoc. Prof. Berna Sarıyar Akbulut	.....
Assist. Prof. Kerem Uğuz	.....

DATE OF APPROVAL:

to my family

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my supervisor Professor Uğur Akman for his endless patience, valuable guidance and advice during my BSc. and MSc. years, and throughout the preparation of this thesis.

I am grateful to my family for their continuous support and encouragement in all steps of my life.

ABSTRACT

MULTIPLE AND ALTERNATE OPTIMA OF LP PROBLEMS VIA RECURSIVE MILP: A MATLAB IMPLEMENTATION

In this thesis, the Recursive Mixed Integer Linear Programming (RMILP) algorithm invented by Prof. Grossman's group at the Carnegie Mellon University is studied. The algorithm is guaranteed to find all the alternate optima of the LP problems. The main advantage of the proposed algorithm is that it can be built on efficient tools such as GAMS algebraic modeling system and does not require any modification of the LP solver. But the RMILP code written in GAMS was not generic. Since it was problem specific coding, it must be changed for each different problem. Prof. Akman generated a generic GAMS code that can be used even by a novice LP user without modification of the recursive MILP part by the user. In this thesis, final tests of Prof. Akman's work and implementation in MATLAB were performed. With this purpose, a MATLAB code consisting of two parts namely, a function part and a main part is developed. In the main part, the user enters the problem specific data and termination criterion. Then the function part is called, where LP and MILP problems are solved recursively. The same function part can be used for any standard LP problem without any modification. The MATLAB code is generic; it can accommodate any LP/MILP solver. The MATLAB code developed provides the user with all the alternate optimal solutions, as well as next K best vertex solutions of an LP problem. With this capability, the decision maker will see the difference between the optimal value and the next best objective function value, or next K best objective function values with a single run of the code. Alternate solutions enable decision maker to make choice by considering other incremental factors that are not explicitly incorporated into the optimization model. Several LP problems were solved using GAMS and MATLAB and the same results were obtained on both software platforms. In both GAMS and MATLAB, some solutions were replicated depending on the LP/MILP solvers. A dendrogram, representing hierarchical solution clusters, was used to visualize the proximity of the alternate solutions and to eliminate replicates.

ÖZET

DP PROBLEMLERİNDE ÖZYİNELİ KTDP İLE ÇOKLU VE ALTERNATİF OPTİMAL ÇÖZÜMLER: MATLAB UYGULAMASI

Bu çalışma, Carnegie Mellon Üniversitesi'nde Prof. Grosmann'ın ekibi tarafindan geliştirilen Özyineli Karma Tamsayılı Doğrusal Programlama (ÖKTDP) algoritması üzerinedir. Bu algoritma, Doğrusal Programlama (DP) problemlerinin tüm alternatif optimal çözümlerini kesin olarak bulmaktadır. Algoritmanın temel avantajı, GAMS cebirsel molelleme sistemi gibi verimli bir dilde yazılabilmesi ve DP çözücüsünde değişiklik gerektirmemesidir. GAMS'te yazılmış olan ÖKTDP kodu genel olmayıp probleme özel olduğundan, her farklı problem için değiştirilmesi gerekmekteydi. Prof. Akman başlangıç seviyesindeki bir kullanıcı tarafından bile ÖKTDP kısmının değiştirilmesi gerekmeden kolayca kullanılabilecek genel bir GAMS kodu yazmıştır. Bu tez çalışmasında, Prof. Akman'ın bu çalışmasının son sınamaları yapılmış ve algoritma MATLAB'a uyarlanmıştır. Ana ve fonksiyon kısımlarından oluşan bir MATLAB kodu geliştirilmiştir. Ana kısımda kullanıcı, probleme özel verileri tanıtmalı ve bitiş kriterini belirlemelidir. Bu kısmın sonunda DP ve ÖKTDP problemlerinin özyinelemeli olarak çözüldüğü, standart DP problemleri için değişiklik gerektirmeyen, fonksiyon kısmı çağırılmaktadır. Geliştirilen MATLAB kodu genel olup, her DP/ÖKTDP çözücüsü ile kullanılabilmektedir. Bu kod, kullanıcıya bir DP probleminin tüm alternatif optimal çözümlerini ve sonraki en iyi K köşe çözümü verir. Karar alacak olan kişi, kodun bir defa çalıştırılması ile amaç fonksiyonu değeri ile bir sonraki en iyi değer veya sonraki en iyi K değer arasındaki farkı görebilecektir. Alternatif çözümler, kullanıcının optimizasyon modeline açıkça eklenemeyen etmenleri de göz önünde bulundurarak karar vermesine imkan sağlamaktadır. Birtakım DP örnekleri GAMS ve MATLAB kullanılarak çözülmüş ve her iki yazılımla aynı sonuçlar elde edilmiştir. Her iki yazılımda da, kullanılan DP/ÖKTDP çözücülerine bağlı olarak, bazı sonuçların tekrarlandığı gözlemlenmiştir. Alterfatif optimal çözümlerin yakınlığını gözlemlemek ve tekrarlanan çözümleri elemek amacıyla hiyerarşik çözüm kümelerini gösteren öbekağacı çizimi kullanılmıştır.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS iv
ABSTRACT v
ÖZET vi
LIST OF FIGURES ix
LIST OF TABLES xvi
LIST OF SYMBOLS xviii
LIST OF ACRONYMS/ABBREVIATIONS xxi
1. INTRODUCTION 1
2. THE LP PROBLEM 4
3. RECURSIVE MILP METHOD FOR FINDING ALTERNATE AND NEXT BEST OPTIMA 13
4. GAMS-MATLAB INTERFACE 23
5. GAMS TO MATLAB CONVERSION 30
6. EXAMPLES 59
6.1. A Simple LP 59
6.2. A Degenerate LP 63
6.3. Refinery Blending Problem 68
6.4. Thermal Cracker Problem 74
6.5. A Metabolic Engineering Example 84
6.5.1. Hierarchical Clustering (HC) Analysis 93
7. CONCLUSIONS & RECOMMENDATIONS 99
7.1. Conclusions 99
7.2. Recommendations for Future Work 100
APPENDIX A: GAMS & MATLAB CODES FOR THE EXAMPLE PROBLEMS 102
A.1. A Simple LP 102
A.1.1. GAMS Code 102
A.1.2. MATLAB Code 103
A.2. A Degenerate LP 104
A.2.1. GAMS Code 104

A.2.2. MATLAB Code 105
A.3. Refinery Blending Problem 106
A.3.1. GAMS Code 106
A.3.2. MATLAB Code 108
A.4. Thermal Cracker Problem 109
A.4.1. GAMS Code 109
A.4.2. MATLAB Code 110
A.5. Metabolic Engineering Example 111
A.5.1. GAMS Code 111
A.5.2. MATLAB Code 114
APPENDIX B: SOLUTIONS OF THE METABOLIC ENGINEERING EXAMPLE 117
REFERENCES 123

LIST OF FIGURES

Figure 2.1. Graphical representation of the LP problem in Equation 2.4 (Phalakornkule *et al.*, 2000). 6

Figure 2.2. Graphical representation of the LP problem in Equation 2.4 with modified objective function coefficients. 7

Figure 2.3. Graphical representation of the LP problem in Equation 2.5. 8

Figure 2.4. Feasible region depicting the alternate optima and the K-best solutions. 11

Figure 3.1. Hyperplane of $x_3 = 0$ in Equation 3.8. 19

Figure 3.2. The four alternate and the next best objective function values of the LP problem in Equation 3.8. 20

Figure 3.3. Decision variable values in the 9 solutions of the LP problem in Equation 3.8. 20

Figure 3.4. Hyperplane of $x_3 = 6$ for the 4th next best solution of the LP in Equation 3.8. 21

Figure 4.1. Flowchart of GAMS-MATLAB interface. 24

Figure 4.2. Parameter definition. 25

Figure 4.3. Definition of the sets. 26

Figure 4.4. Inputs of the '`wgdx`' routine. 26

Figure 4.5. The ‘`wgdx`’ routine. 27

Figure 4.6. Determining the output of the ‘`gams`’ routine. 27

Figure 4.7. The ‘`gams`’ routine. 27

Figure 4.8. The ‘`rgdx`’ routine. 27

Figure 4.9. Definition of the sets and parameters. 28

Figure 4.10. Reading the data into GAMS. 28

Figure 4.11. Writing the results into the GDX file. 29

Figure 5.1. Definition of the sets. 33

Figure 5.2. Definition of the coefficient matrices and vectors in canonical matrix form. 34

Figure 5.3. Definition of variables and declaration of the equality constraints. 34

Figure 5.4. Model construction for the initial canonical LP. 35

Figure 5.5. Recovering the optimal solution. 35

Figure 5.6. Definition of the binary variables. 35

Figure 5.7. Definition of parameters to keep the iteration results. 36

Figure 5.8. Keeping the results of the first iteration. 36

Figure 5.9. Definition of parameters `WT(T,N)` and `NZ(T)`. 37

Figure 5.10. Definition of parameter `XSOL(T,I)`. 37

Figure 5.11. Declaration of Equations 5.4 and 5.5. 37

Figure 5.12. Declaration of Equations 5.6 and 5.7. 38

Figure 5.13. Declaration of the RMILP model. 38

Figure 5.14. Solving RMILP iteratively. 39

Figure 5.15. Keeping the iteration results. 39

Figure 5.16. Determining whether the program will continue. 40

Figure 5.17. Stopping the iterations. 40

Figure 5.18. Detecting integer infeasibility. 40

Figure 5.19. Entering the problem specific data. 41

Figure 5.20. Definition of the coefficient matrices and vectors in canonical matrix form. 42

Figure 5.21. Definition of parameter `InputType`. 42

Figure 5.22. Constructing the parameters of canonical matrix form. 43

Figure 5.23. Definition of the sets in GAMS. 44

Figure 5.24. Definition of the sets in MATLAB. 44

Figure 5.25. Constructing and solving the model in GAMS. 45

Figure 5.26. Constructing and solving the model in MATLAB. 45

Figure 5.27. Recovering original decision variables in GAMS. 46

Figure 5.28. Defining binary variables and parameters in GAMS. 46

Figure 5.29. Defining binary variables and parameters in MATLAB. 46

Figure 5.30. Defining matrices `WT` and `ZT` in GAMS. 47

Figure 5.31. Defining matrices `WT` and `ZT` in MATLAB. 47

Figure 5.32. Recovering original decision variables in GAMS. 47

Figure 5.33. Recovering original decision variables in MATLAB. 48

Figure 5.34. Declaring Equations `ACUT(T,N)` and `YSUM(T)` in GAMS. 48

Figure 5.35. Declaring Equations `ACUT(T,N)` and `YSUM(T)` in MATLAB. 48

Figure 5.36. Declaring Equations `BASISC(T)` and `MNB(T,N)` in GAMS. 49

Figure 5.37. Declaring Equations `BASISC(T)` and `MNB(T,N)` in MATLAB. 49

Figure 5.38. Constructing and solving RMILP in GAMS. 50

Figure 5.39. Constructing and solving RMILP in GAMS. 51

Figure 5.40. Constructing the overall coefficient matrix and vector of right hand sides. 51

Figure 5.41. Determining equation types. 52

Figure 5.42. Defrining parameter `xint`. 52

Figure 5.43. Solving the RMILP. 53

Figure 5.44. Defining parameters. 53

Figure 5.45. Solving RMILP using ‘`glpk`’. 54

Figure 5.46. Checking integer feasibility. 55

Figure 5.47. Checking the problem feasibility in case of ‘`glpk`’. 55

Figure 5.48. Checking the problem feasibility in case of ‘`lp_solve`’. 55

Figure 5.49. Keeping the results of the iterations in GAMS. 56

Figure 5.50. Keeping the results of the iterations in MATLAB. 56

Figure 5.51. Termination criterion in GAMS. 57

Figure 5.52. Termination criterion in MATLAB. 57

Figure 5.53. Displaying the results in MATLAB. 58

Figure 6.1. Graphical representation of the LP problem in Equation 6.1. 61

Figure 6.2. The two alternate and the next best objective function values of the LP problem in Equation 6.1. 62

Figure 6.3. Decision variables for the two alternate and the next best solutions of the LP problem in Equation 6.1. 62

Figure 6.4. Hyperplane of $x_3 = 0$ in Equation 6.5. 65

Figure 6.5. The four alternate and the next best objective function values of the LP problem in Equation 6.5. 66

Figure 6.6. Decision variable values in the 9 solutions of the LP problem in Equation 6.5. 67

Figure 6.7. Crude oil costs and product prices for the Refinery Blending problem. 68

Figure 6.8. The two alternate and the next best objective function values of the LP problem in Equation 6.11. 72

Figure 6.9. Decision variable values in the 5 solutions of the LP problem in Equation 6.11. 73

Figure 6.10. Flow diagram of thermal cracker. 75

Figure 6.11. The 6 best objective function values of the LP problem in Equation 6.34. 82

Figure 6.12. Decision variable values in the 6 solutions of the LP problem in Equation 6.34. 83

Figure 6.13. The metabolic network of E. coli (Phalakornkule *et al.*, 2001). 86

Figure 6.14. The 5 best objective function values of the metabolic engineering example. 92

Figure 6.15. An example hierarchical clustering and the corresponding dendrogram (Akman *et al.*, 2008). 94

Figure 6.16. Hierarchical clustering dendrogram of alternative solutions (nine unique 1st best solutions) of the metabolic engineering example. 95

Figure 6.17. Hierarchical clustering dendrogram of all solutions of the metabolic engineering example. 96

Figure 6.18. Hierarchical clustering dendrogram of fluxes in the metabolic engineering example. 97

LIST OF TABLES

Table 2.1. The first Simlex table for problem in Equation 2.6. 9

Table 2.2. The second Simlex table for problem Equation 2.6. 10

Table 2.3. The third Simlex table for problem Equation 2.6. 10

Table 6.1. Problem specifications for each LP. 59

Table 6.2. % yield and maximum allowable production data for the Refinery Blending problem. 68

Table 6.3. The 4-best solutions of the LP in Equation 6.11. 72

Table 6.4. Yield structure in terms of weight fractions. 75

Table 6.5. Fuel requirements to run the cracking system. 76

Table 6.6. Heating values of the recycled products. 76

Table 6.7. Price structure on the feeds and products and fuel costs. 76

Table 6.8. The 6-best solutions of the LP in Equation 6.34. 82

Table 6.9. Product flow values for the Thermal Cracker problem. 83

Table 6.10. The 5-best solutions of the metabolic engineering example. 91

Table A.1. Solutions from S1 to S9 for 54 solutions of the metabolic engineering example. 117

Table A.2.. Solutions from S10 to S18 for 54 solutions of the metabolic engineering example (cont.). 118

Table A.3. Solutions from S19 to S27 for 54 solutions of the metabolic engineering example (cont.). 119

Table A.4. Solutions from S28 to S36 for 54 solutions of the metabolic engineering example (cont.). 120

Table A.5. Solutions from S37 to S45 for 54 solutions of the metabolic engineering example (cont.). 121

Table A.6. Solutions from S45 to S54 for 54 solutions of the metabolic engineering example (cont.). 122

LIST OF SYMBOLS

A	Label of a horizontal line in Figure 6.15
$\boldsymbol{A}^1$	Coefficient matrix of equality constraints
$\boldsymbol{A}^2$	Coefficient matrix of inequality constraints
B	Label of a horizontal line in Figure 6.15
b	Right hand side values of equalities in Equation 2.6
$\boldsymbol{B}$	Coefficient matrix
$\boldsymbol{b}^1$	Right hand side vector of equality constraints
$\boldsymbol{b}^2$	Right hand side vector of inequality constraints
C	Label of a horizontal line in Figure 6.15
c	Objective function coefficients in Equation 2.6
$\boldsymbol{c}$	Coefficient vector of objective function
$\boldsymbol{c}^T$	Transpose of $\boldsymbol{c}$
f	Objective function value in Equation 2.5
f_i	Objective function value at point i in Figure 2.4.
f^*	Optimal objective function value
G	Cluster
i	Variable index
$\boldsymbol{I}$	Identity matrix
K	Number of next best solutions
k	Number of equality constraints
m	Number of inequality constraints
N	Number of data items to be clustered
n	Number of decision variables
NZ^K	Set of non-zero basic variables at iteration K
NZ^k	Set of non-zero basic variables at iteration k
p	Number of compounds involved in metabolic reactions
q	Number of metabolic reactions
$\boldsymbol{q}$	Right hand side vector of equality constraints
r_{ATP}	Rate of ATP production

r_i	ith flux in metabolic engineering example
$\boldsymbol{R}^m$	m-dimensional real coordinate space
$\boldsymbol{R}^n$	n-dimensional real coordinate space
S	Solution
s	Individual data points in Figures 6 16 and 6.17
$\boldsymbol{S}$	Stoichiometric matrix representing metabolic reactions
$\boldsymbol{s}$	Slack variables for inequality constraints
s_i	Slack variables in Equation 2.6
$\boldsymbol{s}^L$	Vector of slack variables for lower bound of $\boldsymbol{x}$
$\boldsymbol{s}^U$	Vector of slack variables for upper bound of $\boldsymbol{x}$
U	A valid upper bound for all z_i
$\boldsymbol{v}$	Vector of fluxes through all of the reactions in a network
w_i	Binary variable defined for each z_i at the current iteration
x	Decision variable in Equation 2.4
$\boldsymbol{x}$	Vector of decision variables
x_i	ith element of the decision variable vector $\boldsymbol{x}$
$x_i{}^*$	Value of decision variable x_i at the optimal solution
$(x_1, x_2)^i$	Coordinates of point i in Figure 2.4.
$\boldsymbol{x}^L$	Vector of lower bounds
$\boldsymbol{x}^U$	Vector of upper bounds
y	Decision variable in Equation 2.4
Y_{ATP}	ATP yield
Y_c	Carbon yield
y_i	Binary variable defined for each non-zero basic z_i at the previous iteration
Z	Objective function value
$\boldsymbol{z}$	$(2n + m)$-dimensional column vector
$(Z^1)^*$	Optimal objective function value at iteration 1 in RMILP Algorithm
Z^K	Objective function value at the Kth iteration in RMILP Algorithm

$\boldsymbol{\alpha}$	Coefficient vector of objective function
$\boldsymbol{\alpha}^T$	Transpose of $\boldsymbol{\alpha}$
ε	Termination criterion
μ	Specific growth rate
ϕ	Objective function value in Equation 2.4
ϕ^*	Optimal objective function value
*	Superscript indicating optimal value
$	Dollars
¢	Cents
#	Number

LIST OF ACRONYMS/ABBREVIATIONS

3PG	3-Phosphoglycerate
AcCoA	Acetyl-Coenzyme A
ATP	Adenosine Triphosphate
bbl	Barrel
Btu	British Thermal Unit
CIT	Citrate
CPU	Central Processing Unit
DNG	Debutanized Natural Gasoline
E4P	Erythrose 4-Phosphate
E. coli	*Escherichia coli*
F6P	Fructose 6-Phosphate
FADH	Flavin Adenine Dinucleotide
FBA	Flux Balance Analysis
G6P	Glucose 6-Phosphate
g	Grams
GDX	GAMS Data Exchange
h	Hour
HC	Hierarchical Clustering
HMP	Hexose Monophosphate
KG	Ketoglutarate
lb	Libra
LP	Linear Programming
MAL	Malate
max	Maximize
MILP	Mixed Integer Linear Programming
min	Minimize
MINLP	Mixed Integer Non-linear Programming
MIP	Mixed Integer Programming
mmol	Milimoles

NADH	Nicotinamide Adenine Dinucleotide
NADPH	Nicotinamide Adenine Dinucleotide Phosphate Hydrogen
NMR	Nuclear Magnetic Resonance
OAA	Oxaloacetate
P	Phosphate
PEP	Phosphoenolpyruvate
PYR	Pyruvate
R5P	Ribose 5-Phosphate
RMILP	Recursive Mixed Integer Linear Programming
SBML	Systems Biology Markup Language
s.t.	Subject to
Succ.	Succinate
TCA	tricarboxylic acid
THY	Total Fuel Heating Value
TP	Triose Phosphate

1. INTRODUCTION

Linear Programming (LP) analysis is a widely used tool applied in the solution of optimization problems (for linear models or after model linearization) in diverse areas including engineering, economics, finance, banking, management, health services, airlines, forestry, military, transportation, etc. The oldest and most widely used area of LP, as a mathematical modeling technique, deals with the problem of allocating limited resources among competing activities in the best possible (optimal) way.

Alternate optima are useful in practice since they allow the decision maker to choose from multiple solutions without experiencing any deterioration in the objective function (Tsai *et al.*, 2008). Furthermore, alternate solutions enable decision maker to make choice by considering other incremental factors (e.g., fuzzy knowledge) that are not (or cannot be) explicitly incorporated into the optimization model.

Integer cuts are often used to find multiple solutions of integer programs. In this case, the integer program is solved to optimality, an integer cut is added to the model in order to make the previous solution infeasible, and the model is solved again to find the second best integer solution. This process can then be repeated to obtain the best several solutions or all the feasible solutions of an integer program. In this approach, the user is required to explicitly model the integer cuts as part of the optimization model (Sahinidis and Tawarmalani, 2012).

Currently, only few of the commercial Mixed Integer Linear Programming (MILP) solvers (e.g., BARON and CPLEX), and only in their relatively recent versions, can find alternate, next best, or all possible solutions to MILP problems. For example, BARON (Sahinidis and Tawarmalani, 2005; Sahinidis and Tawarmalani, 2010) does not rely on integer cuts to find multiple solutions and thus completely eliminates the need for coding of integer cuts by the user. In BARON, all feasible solutions found are recorded, and the current node is deleted if it is infeasible or all integer variables are fixed. Nodes where feasible solutions are identified are branched further until they become points in the search space or infeasible. The advantage of BARON over using the integer cuts is that all

feasible solutions are identified through a single application of branch-and-reduce, thus avoiding a great deal of duplicate work in repetitive search trees (Sahinidis *et al.*, 2003). BARON's approach applies to integer (MILP) as well as continuous (MINLP) programs (Sahinidis and Tawarmalani, 2012). However, BARON's ability to find alternate or next best solutions is due to an intelligent implementation of the branch-and-bound procedure in the solution of MILPs and thus it cannot provide any alternate or next best solutions to LPs. Likewise, with CPLEX, it is also possible to enumerate all solutions of a MILP problem that are valid for a specific criterion under some limitations. One of such limitations is that only one solution for each set of discrete variables is provided even though there may exist several solutions that have the same values for all discrete variables but different values for continuous variables. With CPLEX, it is also possible to obtain solutions within a given percentage of the optimal solution, giving next best solutions (GAMS Development Corporation, 2012).

Currently, none of the commercial LP solvers can find alternate, next best, or all possible solutions to LP problems. The Recursive Mixed Integer Linear Programming (RMILP) method has been invented by Prof. Grossmann's group at the Carnegie-Mellon University with the purpose of finding all alternate optimal solutions of LP problems originating from metabolic engineering examples (Lee *et al.*, 2000; Phalakornkule *et al.*, 2001). In these studies, the RMILP algorithm was used to obtain the carbon trafficking alternatives of an *Escherichia coli* mutant lacking pyruvate kinase in order to design ^{13}C NMR experiments for maximum contrast. The method has also been used in genome-scale metabolic models (Mahadevan and Schilling, 2003; Reed and Palsson, 2004; Herrgard *et al.*, 2006), biochemistry (Vo *et al.*, 2004), and in general integer linear programs (Tsai *et al.*, 2008).

One advantage of the algorithm of Prof. Grossmann's group (Lee *et al.*, 2000, Phalakornkule *et al.*, 2001) is that it can be implemented relatively easily in algebraic modeling languages such as of the GAMS or AMPL. However, the original code written in GAMS was not generic. It was problem specific coding (GAMS' scalar coding) and must be changed for each different problem. As a product of several ChE492 Project (B.Sc. Thesis) work, Prof. Akman generated a very useful generic GAMS code (GAMS' array

coding) that can be used even by a novice LP user without any modification of the RMILP part by the user.

While it is possible to find only alternate optima with the original code, with the code generated by Prof. Akman, it is possible to obtain the next K-best vertex solutions with the re-definition of an additional parameter, ε. This way, all vertex solutions giving objective function values greater than the initial optimum value, (base solution obtained via a standard LP solver) by less than ε are found by the algorithm. With this property, the decision maker will see the difference between the optimal value and the next best objective function value, or next K best objective function values with a single run of the LP/RMILP code.

Here in this thesis, the RMILP algorithm is implemented in MATLAB. MATLAB has optimization tools which are useful to some extent for small-to-medium-scale linear models. While very powerful and robustly tested modeling languages, such as GAMS, have optimization tools very useful for very-large-scale linear models, they have extremely limited capabilities for data manipulation and visualization, for which MATLAB is much better.

In this study, first a brief background on LP problem is provided in Chapter 2. Then, backgrounds on MILP and alternate optima are provided and the RMILP Method invented by Prof. Grossmann's group is explained in detail in Chapter 3. In Chapter 4, an interface between GAMS and MATLAB, which enables the user to utilize the powerful optimization tools of GAMS and data manipulation and visualization capabilities of MATLAB at the same time, is described. The conversion of the RMILP code from GAMS to MATLAB is explained in Chapter 5. Several LP problems taken from the literature are solved using GAMS and MATLAB and the results are discussed as they are presented in Chapter 6. In Chapter 7, conclusions and recommendations for future work are provided. Finally, GAMS and MATLAB codes for the LP problems presented in Chapter 6 are provided in Appendix section.

2. THE LP PROBLEM

Linear Programming (LP) is an optimization terminology for the case where the objective function and all equality and/or inequality constraints are linear. The oldest and most widely used area of LP, as a mathematical modeling technique, deals with the problem of allocating limited resources among competing activities in the best possible (optimal) way. LP is applied in the solution of optimization problems (for linear models or after model linearization) in diverse areas including engineering, economics, finance, banking, management, health services, airlines, forestry, military, transportation, etc.

LP problem is a convex programming problem. In LP problem, the objective function is convex (linear) and the linear constraints form a convex set, which means that the only local optimum is also the global optimum. A special characteristic of LP problems is that the optimal solution of the problem must lie on some constraint or at the intersection of several constraints, and not in the interior of the convex region (Taha, 2006).

An LP problem can be presented as follows (Lee *et al.*, 2000):

$$\begin{aligned} \min \quad & Z = \boldsymbol{c}^T \boldsymbol{x} \\ \text{s.t.} \quad & \boldsymbol{A}^1 \boldsymbol{x} = \boldsymbol{b}^1 \\ & \boldsymbol{A}^2 \boldsymbol{x} \leq \boldsymbol{b}^2 \\ & \boldsymbol{x}^L \leq \boldsymbol{x} \leq \boldsymbol{x}^U; \quad \boldsymbol{x} \in \boldsymbol{R}^n \end{aligned} \tag{2.1}$$

where, Z is an objective function, $\boldsymbol{c}$ is an n-dimensional column vector of objective function cost coefficients, $\boldsymbol{x}$ is an n-dimensional column vector of decision variables, $\mathbf{A^1}$ is a $k \times n$ matrix of equality constraint coefficients, $\boldsymbol{b}^1$ is a k -dimensional column vector of right hand side values of the equality constraints, $\boldsymbol{A}^2$ is an $m \times n$ matrix of inequality constraint coefficients, $\boldsymbol{b}^2$ is an m -dimensional column vector of right hand side values of the inequality constraints, $\boldsymbol{x}^L$ and $\boldsymbol{x}^U$ are n-dimensional column vectors of lower and upper bounds of the decision variables.

Equation 2.1, which is in non-canonical form, can be formulated in canonical form with the introduction of slack variables $\boldsymbol{s}$ of dimension $m \times 1$ to the inequalities and the conversion of all variables into non-negative variables. To convert variables into non-negative ones, slack variables $\boldsymbol{s}^L$ and $\boldsymbol{s}^U$, n-dimensional column vectors, are introduced for the lower bound and upper bound of $\boldsymbol{x}$, respectively. At this point, it is assumed that $\boldsymbol{x}^L$ and $\boldsymbol{x}^U$ are finite. Using $\boldsymbol{s}^L$ and $\boldsymbol{s}^U$, $\boldsymbol{x}$ is eliminated to obtain the following canonical form for LP.

$$\begin{aligned} \min \quad & \boldsymbol{c}^T \boldsymbol{s}^L + \boldsymbol{c}^T \boldsymbol{x}^L \\ \text{s.t} \quad & \boldsymbol{A}^1 \boldsymbol{s}^L = \boldsymbol{b}^1 - \boldsymbol{A}^1 \boldsymbol{x}^L \\ & \boldsymbol{A}^2 \boldsymbol{s}^L + \boldsymbol{s} = \boldsymbol{b}^2 - \boldsymbol{A}^2 \boldsymbol{x}^L \\ & \boldsymbol{s}^L + \boldsymbol{s}^U = \boldsymbol{x}^U - \boldsymbol{x}^L \\ & \boldsymbol{s}^L,\ \boldsymbol{s}^U, \boldsymbol{s} \geq \boldsymbol{0} \\ & \boldsymbol{s}^L, \boldsymbol{s}^U \in \boldsymbol{R}^n,\ \boldsymbol{s} \in \boldsymbol{R}^m \end{aligned} \tag{2.2}$$

where, n is the number of decision variables and m is the number of inequality constraints in Equation 2.1.

The last term $\boldsymbol{c}^T \boldsymbol{x}^L$ in the objective function is constant and can be dropped. In Equation 2.2, all the constraints are in the form of equalities and all the variables are non-negative variables.

The LP obtained by writing Equation 2.2 in canonical matrix form is as follows:

$$\begin{aligned} \min \quad & Z = \boldsymbol{\alpha}^T \boldsymbol{z} \\ \text{s.t.} \quad & \boldsymbol{B}\boldsymbol{z} = \boldsymbol{q} \\ & \boldsymbol{z} \geq \boldsymbol{0} \end{aligned} \tag{2.3}$$

where,

$$\boldsymbol{z} = \begin{bmatrix} \boldsymbol{s}^L_{n\times 1} \\ \boldsymbol{s}^U_{n\times 1} \\ \boldsymbol{s}_{m\times 1} \end{bmatrix},\ \boldsymbol{\alpha} = \begin{bmatrix} \boldsymbol{c}_{n\times 1} \\ \boldsymbol{0}_{n\times 1} \\ \boldsymbol{0}_{m\times 1} \end{bmatrix},\ \boldsymbol{B} = \begin{bmatrix} \boldsymbol{A}^{\boldsymbol{1}}_{k\times n} & \boldsymbol{0}_{k\times n} & \boldsymbol{0}_{k\times m} \\ \boldsymbol{A}^{\boldsymbol{2}}_{m\times n} & \boldsymbol{0}_{m\times n} & \boldsymbol{I}_{m\times m} \\ \boldsymbol{I}_{n\times n} & \boldsymbol{I}_{n\times n} & \boldsymbol{0}_{n\times m} \end{bmatrix},\ \boldsymbol{q} = \begin{bmatrix} (\boldsymbol{b}^1 - \boldsymbol{A}^1 \boldsymbol{x}^L)_{k\times 1} \\ (\boldsymbol{b}^2 - \boldsymbol{A}^2 \boldsymbol{x}^L)_{m\times 1} \\ (\boldsymbol{x}^U - \boldsymbol{x}^L)_{n\times 1} \end{bmatrix}$$

where, $\boldsymbol{z}$ and $\boldsymbol{\alpha}$ are $(2n+m)$-dimensional column vectors, $\boldsymbol{B}$ is an $(k+m+n)\times(2n+m)$ coefficient matrix, and $\boldsymbol{q}$ is an $(k+m+n)$-dimensional column vector composed of right hand side values of the equality constraints.

A simple two variable (x, y) LP problem is provided below.

$$\begin{aligned} \max \quad & \phi = 20x + 30y \\ \text{s.t.} \quad & x \le 60 \\ & y \le 50 \\ & x + 2y \le 120 \\ & x, y \ge 0 \end{aligned} \tag{2.4}$$

Figure 2.1 is the graphical representation of the above LP problem (Phalakornkule *et al.*, 2001).

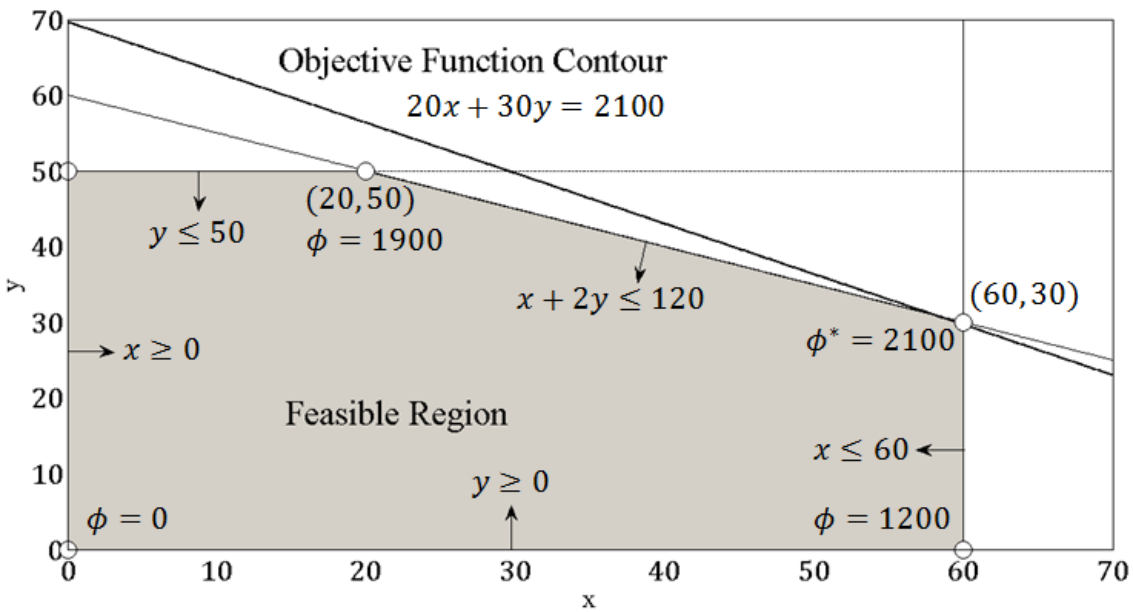

Figure 2.1. Graphical representation of the LP problem in Equation 2.4 (Phalakornkule *et al.*, 2001).

In Figure 2.1, solid lines represent the constraints and side constraints (variable bounds), and the feasible region consists of the all possible combinations of the decision variables (x and y) satisfying all of the constraints. The set of all feasible solutions is the

feasible region and the extreme points (corner points of the feasible region) constitute the basic solutions. In the case of an LP problem having a finite (unique) optimal solution, the solution is found at an extreme point (vertex); since the contours of the objective function are also linear.

As shown in Figure 2.1, the LP problem in Equation2.4 has one optimal solution at $x = 60$, $y = 30$, with an objective function value of $\phi^* = 2100$. This point is the extreme point cut by the bold line representing the objective function contour.

On the other hand, more than one (non-unique) optimal solution may exist in an LP problem. This can be the case when the gradient (slope) of a constraint is parallel to that of the objective function. In the case of infinite number of solutions, the infinite number can be reduced to a finite number of solution types by considering only the solutions at the extreme points (vertices) that are called multiple (alternate) optima. As an example, consider the problem described by Eq. 2.4 again. If the objective function coefficients are changed to $= 10x + 20y$, the contours of the objective function, as shown by the bold line in Figure 2.2, become parallel to the inequality constraint, $x + 2y \leq 120$. In this case, there are two alternate solutions at $x = 20$, $y = 50$ and at $x = 60$, $y = 30$. These two extreme vertex points and the infinite number of points connecting them have the same optimal objective function value of $\phi^* = 1200$. This case is presented in Figure 2.2 below.

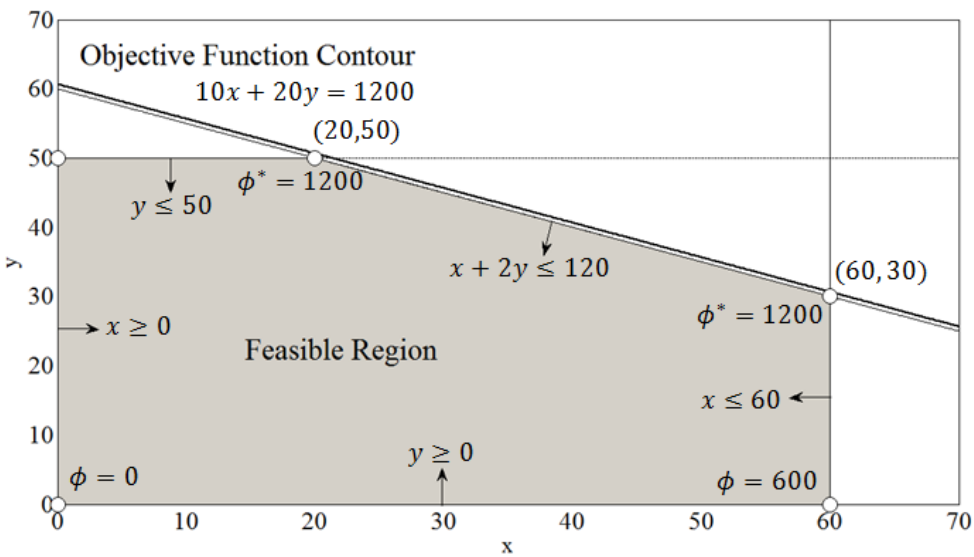

Figure 2.2. Graphical representation of the LP problem in Equation 2.4 with modified objective function coefficients.

An efficient way to solve the LP problems is the Simplex method which was developed by Dantzig in 1947 (Dantzig, 1963). The idea of the Simplex method is to proceed from one basic feasible solution of the constraint set to another, in such a way as to continually improve the value of the objective function until the (global) extremum (minimum or maximum) is reached. It is efficient in moving basic solutions to construct the (global) extremum (Dantzig and Thapa, 1997).

A simple example, including alternate optima and solved by using the Simplex method, is provided below.

$$\begin{aligned} \min \quad & f = -40x_1 - 100x_2 \\ \text{st} \quad & 10x_1 + 5x_2 \leq 2500 \\ & 4x_1 + 10x_2 \leq 2000 \\ & 2x_1 + 3x_2 \leq 900 \\ & x_1, x_2 \geq 0 \end{aligned} \tag{2.5}$$

In this problem, objective function is parallel to the second inequality constraint (cost coefficients of the objective function are negative ten times those of the inequality constraint). Graphical representation of the problem is given below.

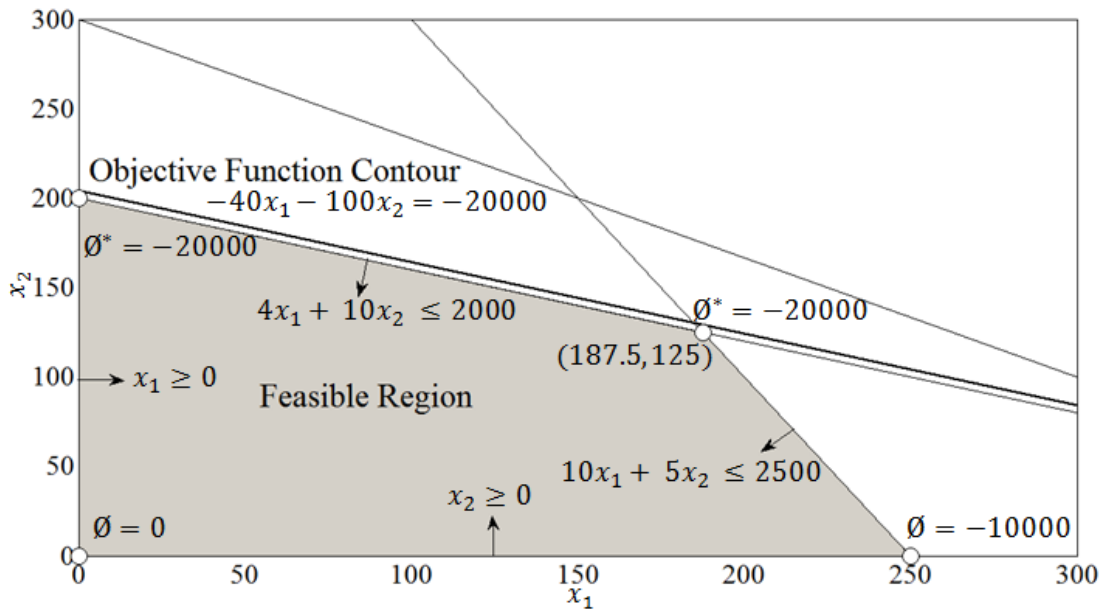

Figure 2.3. Graphical representation of the LP problem in Equation 2.5.

In Figure 2.3, solid lines represent the constraints and side constraints (variable bounds), and the feasible region consists of all possible combination of the decision variables (x_1 and x_2) satisfying all of the constraints. The set of all feasible solutions is the feasible region and the extreme points constitute the basic solutions. In the case of an LP problem having a finite optimal solution, the solution is found at an extreme point (vertex); since the contours of the objective function are also linear.

Standard form of problem in Equation 2.5 for the Simplex method is given below.

$$\begin{aligned} -40x_1 - 100x_2 - f &= 0 \\ 10x_1 + 5x_2 + s_1 &= 2500 \\ 4x_1 + 10x_2 + s_2 &= 2000 \\ 2x_1 + 3x_2 + s_3 &= 900 \\ x_1, x_2, s_1, s_2, s_3 &\geq 0 \end{aligned} \tag{2.6}$$

where s_1, s_2, s_3 are the slack variables that convert inequalities to equalities.

The solution for the problem in Equation 2.6 using Simplex method is provided below.

Table 2.1. The first Simlex table for problem in Equation 2.6.

Basic set	x_1	x_2	s_1	s_2	s_3	B	(b/c), c>0
s_1	10	5	1	0	0	2500	2500/5 = 500
s_2	4	10	0	1	0	2000	2000/10 = 200
s_3	2	3	0	0	1	900	900/3 = 300
-f	-40	-100	0	0	0	0	

In the initial Simplex table (Table 2.1), the slack variables s_1, s_2, s_3 are taken into the basic set with values 2500, 2000, and 900, respectively. At this point, all the original decision variables are zero ($\boldsymbol{x} = \mathbf{0}$), since they are not in the basic set. Since the objective function cost coefficient of x_2 is the most negative coefficient in the last row, x_2 column is seleted as the pivot column. Since the smallest (b/c) value of 200 is obtained for the row of s_2, this row is selected as the pivot row. Thus, x_2 enters the basic set replacing the slack s_2, and Table 2.2 given below is obtained after applying the Simlex method procedure which

involves making the pivot element one and the elements above and below the pivot element zero via elementary row-column operations.

Table 2.2. The second Simlex table for problem Equation 2.6.

Basic set	x_1	x_2	s_1	s_2	s_3	b	(b/c), c>0
s_1	8	0	1	-1/2	0	1500	
x_2	4/10	1	0	1/10	0	200	
s_3	8/10	0	0	-3/10	1	300	
-f	0	0	0	10	0	20000	

In Table 2.2, all entries for the candidate (non-basic) variables in the row of $-f$ are either zero or positive numbers, which means that the solution found is the optimum solution. Here, since the values of the variables that are not in the basic set are zero, the optimal values of the decision variables are $(x_1, x_2) = (0, 200)$ and the optimal value of the objective function is $f^* = -20000$ as seen in Fig. 2.3. However, the coefficient of the non-basic variable x_1 is zero, indicating that an alternative solution exists. If, for example, x_1 is forced into the basic set in place of s_1, the new solution presented in Table 2.3 is obtained by following the similar basic steps of the Simplex method. This new table is also optimal since entries in the last row are either zero or positive numbers.

Table 2.3. The third Simlex table for problem Equation 2.6.

Basic set	x_1	x_2	s_1	s_2	s_3	b	(b/c), c>0
x_1	1	0	1/8	-1/16	0	1500/8	
x_2	0	1	-1/20	1/8	0	125	
s_3	0	0	-1/10	-1/4	1	150	
-f	0	0	0	10	0	20000	

In Table 2.3, the coefficient of non basic variable s_1 is zero. If s_1 is forced into the basis in place of x_1, then Table 2.2 will be reconstructed, the previously obtained solution will be recovered again, and this cycling will continue. That is, the solution will swing between the two alternate optimal solutions (x_1*, x_2*, f*) as $(0,\ 200,\ -20000)$ and $(187.5,\ 125,\ -20000)$, as seen in Fig. 2.3.

The standard LP software such as the Excel Solver, LINDO, GAMS, and MATLAB will provide only one optimal extreme point even if there are several alternate optimal

solutions. With the Recursive MILP (Mixed Integer Linear Programming) method invented by Prof. Grossmann's group (Lee *et al.*, 2000), which is provided in Section 3, it is possible to obtain all alternate optima and sub-optimal next K-best solutions in LP problems.

The alternate optima and the next K-best sub-optima can be described best with Figure 2.4 which depicts the feasible region of the LP problem given by Eq. 2.5. As demonstrated, there are two alternate optima corresponding to points ① and ② in Figure 2.4. Both of these points have the same optimal objective function values of $f_1^* = f_2^* = -20000$ but different (alternative) decision variables $(x_1, x_2)^1 = (187.5, 125)$ and $(x_1, x_2)^2 = (0, 200)$. Point ③ has the next best objective function value of $f_3 = -10000$ with $(x_1, x_2)^3 = (250, 0)$. Point ④ has the second next best objective function value of $f_4 = 0$ with $(x_1, x_2)^4 = (0, 0)$.

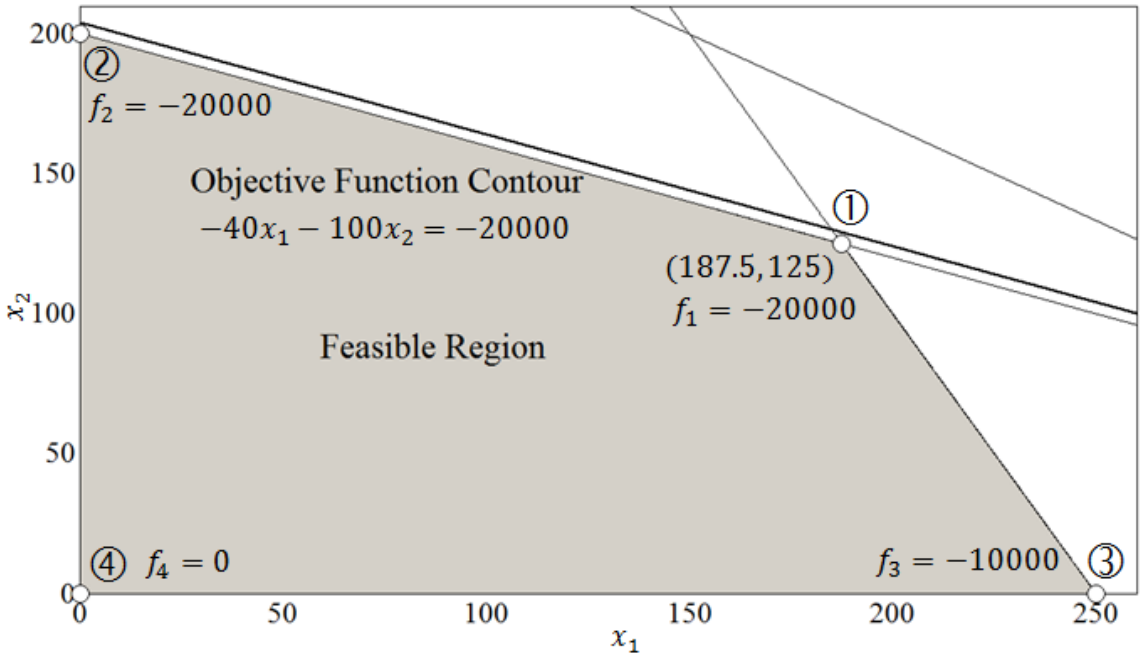

Figure 2.4. Feasible region depicting the alternate optima and the K-best solutions.

Therefore, K-best solution with K=1 refers to the points ① and ② simultaneously. The K-best solution with K=2 refers to the points ①, ② and ③. The K-best solution with K=3 refers to the points ①, ②, ③, and ④. The K-best solution with K=4 is an empty set since all solutions are covered by the K-best solution with K=3.

On the other hand, the Kth-best solution with K=1 refers to the points ① and ② simultaneously. The Kth-best solution with K=2 refers to the point ③ only. The Kth-best solution with K=3 refers to the ④ only. The Kth-best solution with K=4 is an empty set since all solutions are covered by the Kth-best solution with K=3.

From now on, the words “multiple”, “alternate”, and “alternative” will be used interchangeably through the thesis. Furthermore, depending on the context, the word “solution” will refer to the objective function value or to the values of the decision variables However, it should be kept in mind that the word “alternate minima” will refer to a single unique objective function value that can be obtained with alternative (two or more) sets of decision variable values. On the other hand, the word “K-best minima” will refer to set of objective function values with cardinality K (i.e., K different objective function values in increasing order), and the word “K^{th}-best minimum” will refer to a single K^{th} smallest objective function value. Similarly, the word “K-best maxima” will refer to set of objective function values with cardinality K (i.e., K different objective function values in decreasing order), and the word “K^{th}-best maximum” will refer to a single K^{th} largest objective function value.

3. RECURSIVE MILP METHOD FOR FINDING ALTERNATE AND NEXT BEST OPTIMA

Alternate optima are useful in practice since they allow the decision maker to choose from multiple solutions without experiencing any deterioration in the objective function (Tsai *et al.*, 2008). Furthermore, alternate solutions enable decision maker to make choice by considering other incremental factors (e.g., fuzzy knowledge) that are not (or cannot be) explicitly incorporated into the optimization model. The Recursive Mixed Integer Linear Programming (RMILP) Method has been invented by Prof. Grossmann's group at the Carnegie-Mellon University with the purpose of finding all alternate optimal solutions of LP problems originating from metabolic engineering examples (Lee *et al.*, 2000; Phalakornkule *et al.*, 2001). The method has been used in genome-scale metabolic models (Mahadevan and Schilling, 2003; Reed and Palsson, 2004; Herrgard *et al.*, 2006), biochemistry (Vo *et al.*, 2004), and in general integer linear programs (Tsai *et al.*, 2008).

Basically, a RMILP problem is constructed from Equation 3.1 (which is identical to Equation 2.3) with the addition of a set of constraints for changing the basis and obtaining a new extreme point corresponding to one of the alternate optimal solutions. When no other solution having the same objective function can be obtained in the RMILP, the algorithm stops.

$$\begin{aligned} \min \quad & Z = \boldsymbol{\alpha}^T \boldsymbol{z} \\ \text{s.t.} \quad & \boldsymbol{B}\boldsymbol{z} = \boldsymbol{q} \\ & \boldsymbol{z} \geq \boldsymbol{0} \end{aligned} \tag{3.1}$$

where, $\boldsymbol{z}$ and $\boldsymbol{\alpha}$ are $(2n+m)$-dimensional column vectors, $\boldsymbol{B}$is an $(k+m+n) \times (2n+m)$ coefficient matrix of the equality constraints, and $\boldsymbol{q}$ is an $(k+m+n)$-dimensional column vector composed of right hand side values of the equality constraints. Here, n is the number of decision variables and k and m are the number of equality and inequality constraints in Equation 2.1, which is identically provided below as Equation 3.2.

$$\begin{aligned} \min \quad & Z = \boldsymbol{c}^T \boldsymbol{x} \\ \text{s.t.} \quad & \boldsymbol{A}^1 \boldsymbol{x} = \boldsymbol{b}^1 \\ & \boldsymbol{A}^2 \boldsymbol{x} \leq \boldsymbol{b}^2 \\ & \boldsymbol{x}^L \leq \boldsymbol{x} \leq \boldsymbol{x}^U; \quad \boldsymbol{x} \in \boldsymbol{R}^n \end{aligned} \tag{3.2}$$

In Equation 3.1, non-basic variables are equal to zero because they are set to the lower bound of z. Normally, the basic variables are non-zero but they may be equal to zero when there is degeneracy in Equation 3.1. To perform the selection of a new basis, the $0-1$ variable y_i is defined for each variable z_i that is a non-zero basic variable, $i \in NZ^{K-1}$, (NZ^K being the set of non-zero basic variables at the Kth iteration) at the previous iteration, $K-1$. z_i is selected to become non-basic when y_i is set to 1, and when y_i is 0, z_i remains in the basis. For the selection of a new basis, at least one variable y_i must be selected and this is represented by the following logical constraint.

$$\sum_{i \in NZ^{K-1}} y_i \geq 1 \tag{3.3}$$

According to Equation 3.3, at least one of the non-zero basic variables in the $(K-1)^{th}$ soluton is selected to become non-basic at the current iteration, K. To ensure that all alternate bases are generated, the $0-1$ variable w_i is defined which has a value of 1 if variable z_i is non-zero at the current iteration, K. When variable z_i is zero at iteration K, w_i takes the value of 0. The two following constraints (as integer cuts) are valid for w_i.

$$\sum_{i \in NZ^K} w_i \leq |NZ^k| - 1, \quad k = 1, 2, \ldots, K-1 \tag{3.4}$$

$$0 \leq z_i \leq U w_i, \quad i \in I \tag{3.5}$$

With Equation 3.4, at least one of the non-zero variables of the basis found at the previous iterations, $k = 1, 2, \ldots, K-1$, is eliminated from consideration. In Equation 3.5, w_i is forced to take a value of 1 if z_i is non-zero, where U is a valid upper bound for all z_i.

The relationship between the variables y_i and w_i is such that if z_i is non-zero, y_i is 0 and w_i is 1. That is, if the ith non-zero variable is selected to be non-basic, then it must be

0, or it cannot be non-zero. A linear constraint presenting this logical relationship is given below.

$$y_i + w_i \leq 1, \;\; i \in NZ^{K-1} \tag{3.6}$$

The RMILP algorithm (Lee *et al.*, 2000) for finding all the alternate optima is as follows:

Step 1. Set iteration counter $K = 1$. Solve Equation 3.1. Define the set NZ^1 and the optimal objective $(Z^1)^*$.

Step K, for $K \geq 2$

(a) Solve the master problem

$$\begin{aligned} \min \quad & Z = \boldsymbol{\alpha}^T \boldsymbol{z} \\ \text{s.t.} \quad & \boldsymbol{B}\boldsymbol{z} = \boldsymbol{q} \\ & \textstyle\sum_{i \in NZ^{K-1}} y_i \geq 1 \\ & \textstyle\sum_{i \in NZ^K} w_i \leq |NZ^k| - 1, \;\; k = 1, 2, \ldots, K-1 \\ & 0 \leq z_i \leq U w_i, \;\; i \in I \\ & y_i + w_i \leq 1, \;\; i \in NZ^{K-1} \\ & \boldsymbol{z} \geq \boldsymbol{0} \end{aligned} \tag{3.7}$$

(b) Define set NZ^K and continue until $Z^K > (Z^1)^*$.

One advantage of the above algorithm of Prof. Grossmann's group (Lee *et al.*, 2000; Phalakornkule *et al.*, 2001) is that it can be implemented relatively easily in algebraic modeling languages such as of the GAMS or AMPL. The algorithm provided above was invented by Prof. Grossmann's group (Lee *et al.*, 2000; Phalakornkule *et al.*, 2001) but the code written in GAMS was not generic. It was problem specific coding (GAMS' scalar coding) and must be changed for each different problem. As a product of several ChE492 Project (B.Sc. Thesis) works, Prof. Akman generated a generic GAMS code (GAMS' array coding) that can be used even by a novice LP user without any need to modification of the

RMILP part by the user. Here in this thesis, the RMILP algorithm will be implemented in MATLAB. MATLAB has optimization tools which are useful to some extent for small-to-medium-scale linear models. While very powerful and robustly tested modeling languages, such as GAMS, have optimization tools very useful for very-large-scale linear models, they have extremely limited capabilities for data manipulation and visualization, for which MATLAB is much better (Ferris, 2005). Furthermore, data import/export to/from MATLAB is possible for various file formats including Excel worksheet.

While it is possible to find only alternate optima with the original code, with the code generated by Prof. Akman, it is possible to obtain the next K-best vertex solutions with the re-definition of an additional parameter, ε. The termination criterion then becomes: continue until $Z^K - (Z^1)^* > \varepsilon$. This way, all the vertex solutions giving objective function values greater than the initial optimum value, $(Z^1)^*$, (base solution obtained via a standard LP solver) by less than ε are found by the algorithm. With this property, the decision maker will see the difference between the optimal value and the next best objective function value, or next K best vertex objective values with a single run of the LP/RMILP code.

An example, which is a degenerate LP, is provided to show the use of the algorithm. A degenerate LP is the one in which at least one basic variable has a value of 0. The example also proves that the Prof. Grossmann's algorithm (Lee *et al.*, 2000; Phalakornkule *et al.*, 2001) can handle degenerate cases as well. The problem statement is as follows (Phalakornkule *et al.*, 2001):

$$\begin{aligned} \min \quad & Z = x_3 \\ \text{s.t.} \quad & 2x_1 + x_2 + x_3 \leq 8 \\ & x_1 - 2x_2 + x_3 \leq 4 \\ & x_1 - 2x_2 + x_3 \geq -6 \\ & x_1 + 2x_2 + x_3 \geq -2 \\ & x_1 + x_2 + x_3 \leq 6 \\ & -5 \leq x_1, x_2 \leq 5; 0 \leq x_3 \end{aligned} \tag{3.8}$$

where the optimal objective function value is $x_3 = 0$; $(Z^1)^*$=0 The problem has four extreme points on the hyperplane of $x_3 = 0$ (i.e., the plane of x_1 and x_2 once x_3 is fixed at zero, the optimal solution) and has degeneracy. In canonical form, the problem has 10 positive variables and 7 equality constraints. Hence the degrees of freedom is 3 and at each extreme point there are 7 basic variables.

The problem can be cast into the standard LP form that was given by Eq. 2.1, which is identical to Equation 3.9 (Lee *et al.*, 2000).

$$\begin{aligned} \min \quad & Z = \boldsymbol{c}^T \boldsymbol{x} \\ \text{s.t.} \quad & \boldsymbol{A}^1 \boldsymbol{x} = \boldsymbol{b}^1 \\ & \boldsymbol{A}^2 \boldsymbol{x} \leq \boldsymbol{b}^2 \\ & \boldsymbol{x}^L \leq \boldsymbol{x} \leq \boldsymbol{x}^U; \quad \boldsymbol{x} \in \boldsymbol{R}^n \end{aligned} \tag{3.9}$$

where, the vectors and matrices take the following values:

$$\boldsymbol{c}^T = [0\ 0\ 1],\ \boldsymbol{A}^2 = \begin{bmatrix} 2 & 1 & 1 \\ 1 & -2 & 1 \\ -1 & 2 & -1 \\ -1 & -2 & -1 \\ 1 & 1 & 1 \end{bmatrix},\ \boldsymbol{b}^2 = \begin{bmatrix} 8 \\ 4 \\ 6 \\ 2 \\ 6 \end{bmatrix},\ \boldsymbol{x}^L = \begin{bmatrix} -5 \\ -5 \\ 0 \end{bmatrix},\ \boldsymbol{x}^U = \begin{bmatrix} 5 \\ 5 \\ 500 \end{bmatrix}$$

where, 500 is an arbitrarily given upper bound for x_3., and $\boldsymbol{A}^1 = [\,]$, $\boldsymbol{b}^1 = [\,]$.

The canonical form of LP problem is given by Equation 2.3, which is identically provided below as Equation 3.10.

$$\begin{aligned} \min \quad & Z = \boldsymbol{\alpha}^T \boldsymbol{z} \\ \text{s.t.} \quad & \boldsymbol{B}\boldsymbol{z} = \boldsymbol{q} \\ & \boldsymbol{z} \geq \boldsymbol{0} \end{aligned} \tag{3.10}$$

where,

$$\boldsymbol{z} = \begin{bmatrix} \boldsymbol{s}^L_{3\times1} \\ \boldsymbol{s}^U_{3\times1} \\ \boldsymbol{s}_{5\times1} \end{bmatrix},\ \boldsymbol{\alpha} = \begin{bmatrix} \boldsymbol{c}_{3\times1} \\ \boldsymbol{0}_{3\times1} \\ \boldsymbol{0}_{5\times1} \end{bmatrix},\ \boldsymbol{B} = \begin{bmatrix} \boldsymbol{A}^2_{5\times3} & \boldsymbol{0}_{5\times3} & \boldsymbol{I}_{5\times5} \\ \boldsymbol{I}_{3\times3} & \boldsymbol{I}_{3\times3} & \boldsymbol{0}_{3\times5} \end{bmatrix},\ \boldsymbol{q} = \begin{bmatrix} (\boldsymbol{b}^2 - \boldsymbol{A}^2\boldsymbol{x}^L)_{5\times1} \\ (\boldsymbol{x}^U - \boldsymbol{x}^L)_{3\times1} \end{bmatrix}$$

where, $\boldsymbol{z}$ and $\boldsymbol{\alpha}$ are $(2n+m)$-dimensional column vectors, $\boldsymbol{B}$ is an $(k+m+n)\times(2n+m)$ coefficient matrix, and $\boldsymbol{q}$ is an $(k+m+n)$-dimensional column vector composed of right hand side values of the equality constraints. **I** is the identity matrix, n is the number of decision variables, k is the number of equality constraints, and m is the number of inequality constraints in Equation 3.9. For this problem, the vectors and matrices take the following values:

$$\boldsymbol{\alpha} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ - \\ 0 \\ 0 \\ 0 \\ - \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix},\ \boldsymbol{B} = \left[\begin{array}{ccc|ccc|ccccc} 2 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & -2 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ -1 & -2 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ - & - & - & - & - & - & - & - & - & - & - \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{array}\right],\ \boldsymbol{q} = \begin{bmatrix} 23 \\ -1 \\ 11 \\ -13 \\ 16 \\ - \\ 5 \\ 5 \\ 500 \end{bmatrix}$$

where, the number of decision variables, n, is 3, the number of inequality constraints, m, is 5, and there are no equality constraints in the LP problem.

The four multiple optimal solutions found by the algorithm are as follows (for all of them the objective function value is zero):

$$(x_1^*, x_2^*, x_3^*) = (2,4,0),\ (4,0,0),\ (1,-1.5,0),\ (-4,1,0)$$

Due to degeneracy, the point $(2,4,0)$ has a zero-valued basic variable but the algorithm finds all the extreme points and stops when the objective function value is greater than zero. Though the number of positive basic variables changes at each iteration,

no alternate extreme optimum is excluded and each optimal extreme point is visited once. The four extreme points on the hyperplane of $x_3 = 0$ are shown on the following plot:

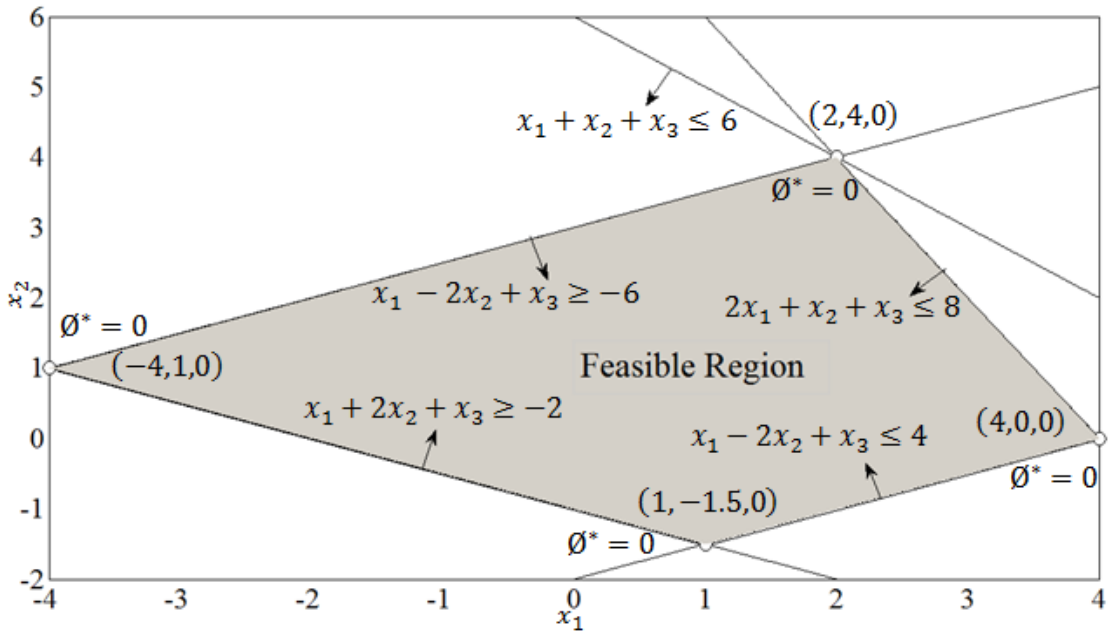

Figure 3.1. Hyperplane of $x_3 = 0$ in Equation 3.8.

The above alternate solutions can be obtained with $\epsilon = 0$ (since $(Z^1)^* = \mathrm{x}_3 = 0$).

For this minimization problem, the 6-best solutions (K-best solutions with K=6): (x_1^*, x_2^*, x_3^*) for the four alternate 1st best solutions and (x_1, x_2, x_3) for the remaining next 5-best solutions, can be obtained by setting $\epsilon = 11$, and these are as follows:

1st best solution:	$(2,\ 4,\ 0)$,	$f^* = 0$
1st best solution:	$(4,\ 0,\ 0)$,	$f^* = 0$
1st best solution:	$(-4,\ 1,\ 0)$,	$f^* = 0$
1st best solution:	$(1,\ -1.5,\ 0)$,	$f^* = 0$
2nd next best vertex solution:	$(-5,\ 1,\ 1)$,	$f = 1$
3rd next best vertex solution:	$(2,\ 0.67,\ 3.33)$,	$f = 3.33$
4th next best vertex solution:	$(-5,\ -1.5,\ 6)$,	$f = 6$
5th next best vertex solution:	$(-5,\ 4,\ 7)$,	$f = 7$
6th next best vertex solution:	$(-5,\ 0.67,\ 10.33)$,	$f = 10.33$

For the 9 set of solutions, the objective function values and values of decision variables for each solution are provided in Figures 3.2 and 3.3 below.

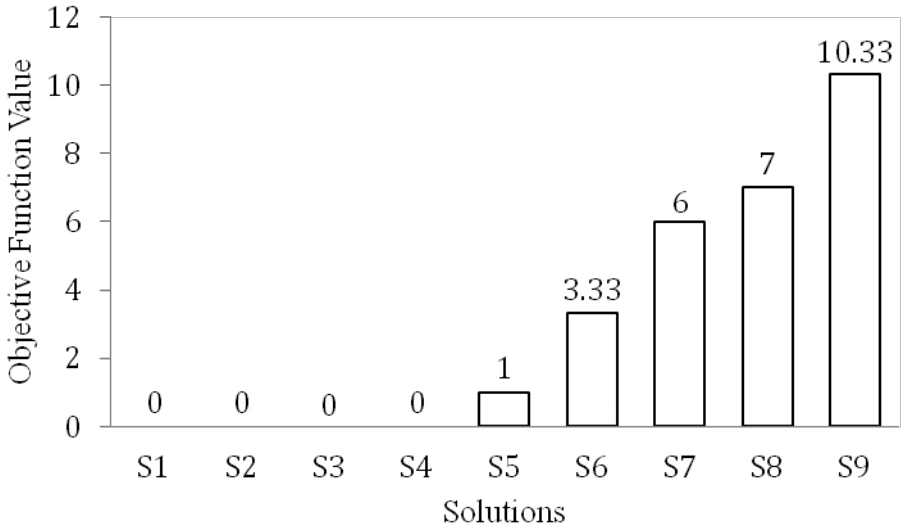

Figure 3.2. The four alternate and the next best objective function values of the LP problem in Equation 3.8.

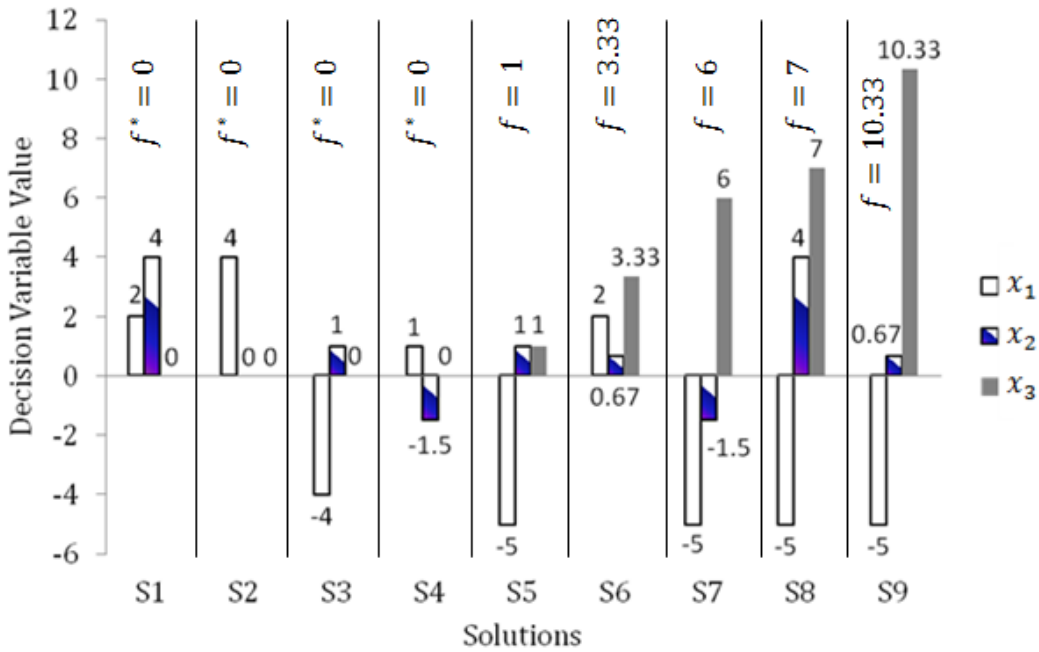

Figure 3.3. Decision variable values in the 9 solutions of the LP problem in Equation 3.8.

No any other solution is possible since $K = 6$ exhausts all possible vertices. The above solutions were obtained using LPSOLVE for MILP in MATLAB. The same

solutions were obtained using GLPK for MILP in MATLAB as well. In GAMS, when XA is used as solver for both LP and MILP, 3rd, 4th, 5th, and 6th best solutions are replicated two times. When XPRESS is used for LP and MILP in GAMS, the 1st best solution of $(x_1^*, x_2^*, x_3^*) = (2,4,0)$ is replicated six times, the 1st best solution of $(x_1^*, x_2^*, x_3^*) = (1, -1.5, 0)$ is replicated four times, the 2nd best solution is replicated three times, the 4th, 5th, and 6th best solutions are replicated two times. However, the unique set of 6 best solutions is obtained using any of the LP/MILP solvers in MATLAB and GAMS and replications depend on the solver used for LP and MILP in GAMS and for MILP in MATLAB. Such a replication is also observed in MATLAB when Equation 3.8 is solved with the addition of Equations 3.11 and 3.12 below using LPSOLVE for MILP. The unique set of results, provided in Figure 3.4, is the same as those obtained using GLPK as the MILP solver.

For example for the 4th next best solution the hyperplane of $x_3 = 6$ is as follows:

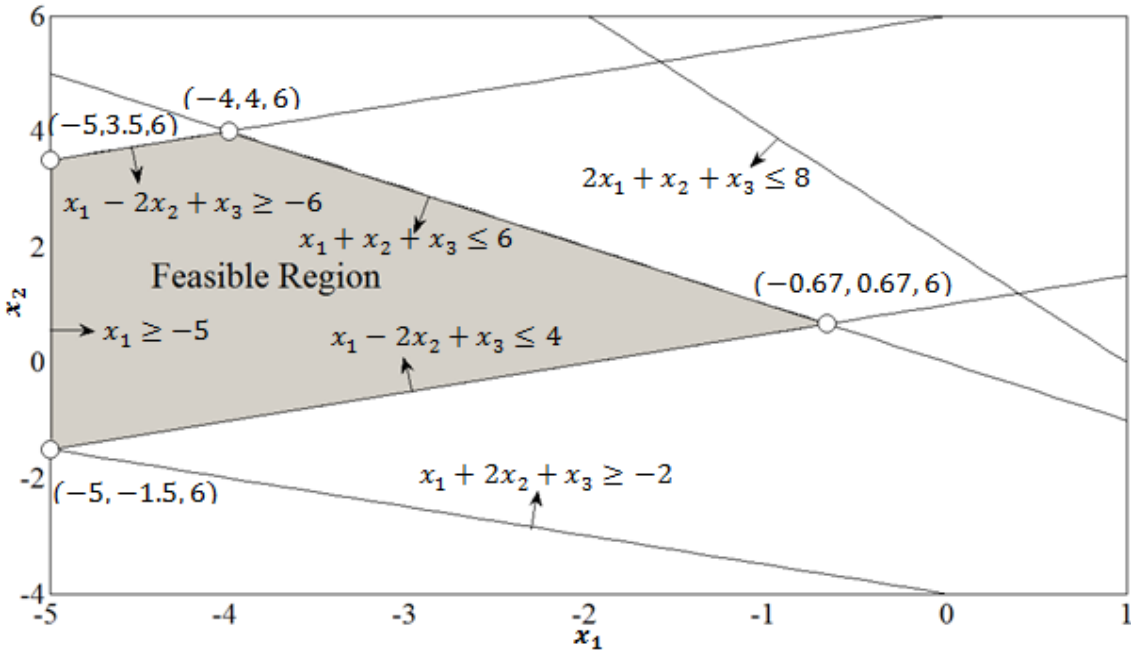

Figure 3.4. Hyperplane of $x_3 = 6$ for the 4th next best solution of the LP in Equation 3.8.

The alternate solutions in Figure 3.4 are obtained with the addition of the constraints below into the LP in Equation 3.8.

$$x_3 \geq 6 \tag{3.11}$$

$$x_1 \geq -5 \tag{3.12}$$

Equation 3.11 forces the objective function, $f = x_3$, to have a value of at least 6 and Equation 3.12 forces x_1 to be at least -5.

The algorithm given above will provide the two alternative optimal solutions of Problem 2.5 as presented in Tables 2.2 and 2.3, as well.

4. GAMS-MATLAB INTERFACE

GAMS has optimization tools that are very useful for very-large-scale linear models. However, it has extremely limited capabilities for data manipulation and visualization, for which MATLAB is much better. In this thesis, to utilize the powerful optimization tools of GAMS and data manipulation and visualization capabilities of MATLAB at the same time, an interface between GAMS and MATLAB (Ferris, 2005) is used such that the problem definition and data entry are made in MATLAB but the primal LP and recursive MILP problems are solved by calling GAMS within MATLAB, and the optimization results are obtained in MATLAB. This way, the results can be further analyzed or viewed using wide variety of tools of MATLAB.

Data import and export from GAMS to MATLAB is achieved using GDX files. GDX is GAMS Data Exchange File. It is a platform independent file which stores data efficiently in binary format. GDX files are machine independent. In GDX files, data is stored in sparse format and is consistent with no duplication, contradiction or syntax errors (Ferris *et al.*, 2011).

Three MATLAB routines of GAMS-MATLAB interface are used in this thesis. The first one is '`rgdx`', which is the MATLAB utility to import data from the GDX file. The second one is '`gams`', which is used to execute a GAMS model from MATLAB and get the results back into MATLAB. The third routine is '`wgdx`' and it is used to create the GDX file containing the MATLAB data. All of these routines get the data in almost the same form as stored in the GDX file (Ferris *et al.*, 2011). In addition to these MATLAB routines, '`$GDXIN`' and '`$LOAD`' directives are used in GAMS to read the data from the GDX file into GAMS during compilation of the GAMS model and '`execute_unload`' statement is used to write the data into the GDX file during execution of the model (GAMS Development Corporation, 2012).

A schematic representation of GAMS-MATLAB interface is provided below in Figure 4.1.

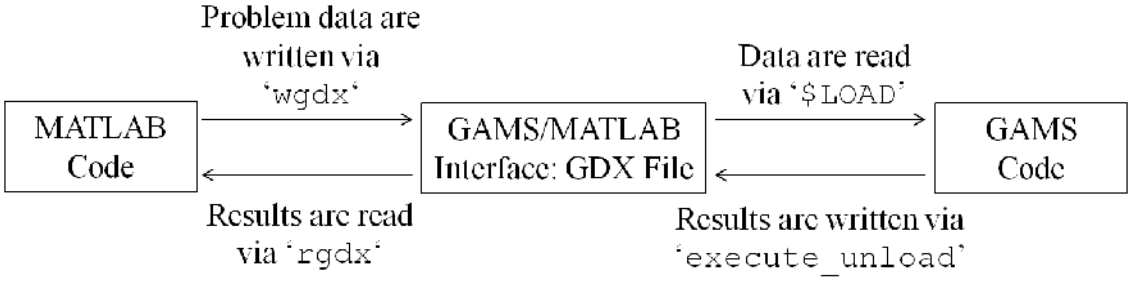

Figure 4.1. Flowchart of GAMS-MATLAB interface.

The MATLAB and GAMS codes for the LP problem in Equation 3.8 (Phalakornkule *et al.*, 2001), identically provided below as Equation 4.1, illustrate the use of the MATLAB- GAMS interface.

$$\begin{aligned} \min \quad & Z = x_3 \\ \text{s.t.} \quad & 2x_1 + x_2 + x_3 \leq 8 \\ & x_1 - 2x_2 + x_3 \leq 4 \\ & x_1 - 2x_2 + x_3 \geq -6 \\ & x_1 + 2x_2 + x_3 \geq -2 \\ & x_1 + x_2 + x_3 \leq 6 \\ & -5 \leq x_1, x_2 \leq 5; 0 \leq x_3 \end{aligned} \tag{4.1}$$

When both sides of the third and fourth constraints in Equation 4.1, $x_1 - 2x_2 + x_3 \geq -6$ and $x_1 + 2x_2 + x_3 \geq -2$, are multiplied by -1, the non canonical form is obtained as:

$$\begin{aligned} \min \quad & Z = x_3 \\ \text{s.t.} \quad & 2x_1 + x_2 + x_3 \leq 8 \\ & x_1 - 2x_2 + x_3 \leq 4 \\ & -x_1 + 2x_2 - x_3 \leq 6 \\ & -x_1 - 2x_2 - x_3 \leq 2 \\ & x_1 + x_2 + x_3 \leq 6 \\ & -5 \leq x_1, x_2 \leq 5; 0 \leq x_3 \end{aligned} \tag{4.2}$$

Equation 4.2 can be written in the form of Equation 2.1, provided below as Equation 4.3:

$$\begin{aligned} \min \quad & Z = \boldsymbol{c}^T \boldsymbol{x} \\ \text{s.t.} \quad & \boldsymbol{A}^1 \boldsymbol{x} = \boldsymbol{b}^1 \\ & \boldsymbol{A}^2 \boldsymbol{x} \leq \boldsymbol{b}^2 \\ & \boldsymbol{x}^L \leq \boldsymbol{x} \leq \boldsymbol{x}^U; \quad \boldsymbol{x} \in \boldsymbol{R}^n \end{aligned} \tag{4.3}$$

where,

$$\boldsymbol{c}^T = [0\ 0\ 1],\ \boldsymbol{A}^2 = \begin{bmatrix} 2 & 1 & 1 \\ 1 & -2 & 1 \\ -1 & 2 & -1 \\ -1 & -2 & -1 \\ 1 & 1 & 1 \end{bmatrix},\ \boldsymbol{b}^2 = \begin{bmatrix} 8 \\ 4 \\ 6 \\ 2 \\ 6 \end{bmatrix},\ \boldsymbol{x}^L = \begin{bmatrix} -5 \\ -5 \\ 0 \end{bmatrix}, \boldsymbol{x}^U = \begin{bmatrix} 5 \\ 5 \\ 500 \end{bmatrix}$$

where, 500 is an arbitrarily given upper bound for x_3.

The following is the MATLAB code for the problem in Equation 4.1. Parameters are defined as follows:

```
C  = [0 0 1];
b2 = [8  4  6  2  6 ];
A2 = [2  1  1
      1 -2  1
     -1  2 -1
     -1 -2 -1
      1  1  1];
XL = [-5 -5 0];
XU = [5 5 500];
EPSF = 0.1;
T = 15;
```

Figure 4.2. Parameter definition.

where, `EPSF` is the termination criterion (the RMILP algorithm stops when the last obtained objective function value is greater than the optimum value of the objective function plus `EPSF`). `T` is the maximum number of iterations.

The number of decision variables and the number of inequality constraints in Equation 4.3 are defined below as sets `N` and `M`, respectively.

```
N = size(c,2);
M = size(b2,2);
```

Figure 4.3. Definition of the sets.

Inputs of the ‘`wgdx`’ routine are provided below.

```
Ns.name = 'N';
Ns.val  = [linspace(1,N,N)]';
Ns.type = 'set';
Ms.name = 'M';
Ms.val  = [linspace(1,M,M)]';
Ms.type = 'set';
Ts.name = 'T';
Ts.val  = [linspace(1,T,T)]';
Ts.type = 'set';
cs.name = 'c';
cs.type = 'parameter';
cs.val  = c';
cs.form = 'full';
cs.dim  = 1;
b2s.name = 'b2';
b2s.type = 'parameter';
b2s.val  = b2';
b2s.form = 'full';
b2s.dim  = 1;
XLs.name = 'XL';
XLs.type = 'parameter';
XLs.val  = XL;
XLs.form = 'full';
XLs.dim  = 1;
XUs.name = 'XU';
XUs.type = 'parameter';
XUs.val  = XU;
XUs.form = 'full';
XUs.dim  = 1;
EPSFs.name = 'EPSF';
EPSFs.type = 'parameter';
EPSFs.val  = EPSF;
EPSFs.form = 'full';
EPSFs.dim  = 0;
A2s.name = 'A2';
A2s.type = 'parameter';
A2s.val  = A2;
A2s.form = 'full';
A2s.dim  = 2;
```

Figure 4.4. Inputs of the ‘`wgdx`’ routine.

where, ‘name’ is a string input representing the name of the symbol, ‘val’ represents the value matrix of the parameter or set, ‘type’ is a string input to specify the type of the symbol (set or parameter), ‘form’ is a string input representing the format in which the ‘val’ matrix has been entered (full or sparse), and ‘dim’ defines the dimension of the data (Ferris *et al.*, 2011). The ‘wgdx’ routine can write multiple symbols into a single GDX file in one call as given below.

```
wgdx('Problem_Data', Ns,Ms,Ts,cs,b2s,XLs,XUs,EPSFs,A2s);
```

Figure 4.5. The ‘wgdx’ routine.

By default, output of the ‘gams’ routine is in structure form. Here, in this example, only the data matrix, ‘val’ field, of the data is of interest. Therefore, the following command is employed to obtain only the value matrix as output.

```
gams_output = 'std'
```

Figure 4.6. Determining the output of the ‘gams’ routine.

At this point, the ‘gams’ routine initializes the GAMS model with MATLAB data, and then executes GAMS on that model and brings the results back into MATLAB. The name of the GAMS file to be called is “Example”.

```
gams('Example');
```

Figure 4.7. The ‘gams’ routine.

After GAMS solves the problem, outputs are written into the GDX file called “Solution_Data”. The ‘rgdx’ routine is used to read the data into MATLAB as below.

```
rs.name = 'P';
r = rgdx ('Solution_Data', rs);
obje = r.val;

rs.name = 'XSOL';
r = rgdx ('Solution_Data', rs);
xs = r.val;
```

Figure 4.8. The ‘rgdx’ routine.

The ‘rgdx’ routine can take up to two arguments. The first argument is a string input containing the GDX file name, which is “Solution_Data” in the above example code. The second argument is a structure input containing information regarding the desired symbol.

The two parts of the GAMS code for the problem in Equation 4.1 are provided below as an example of ‘$GDXIN’, ‘$LOAD’, and ‘execute_unload’ directives. The first part is taken from the beginning of the code, where sets and parameters are introduced.

```
sets
N  'N row dimension of x: (n) / 1*3 /'
M  'row dimension of A2: (m) / 1*5 /'
T  'maximum number of MILP iterations / 1*100 /';
parameters
c(N)      'vector c'
b2(M)     'vector b2'
A2(M,N)   'matrix A2'
XL(N)     'lower bounds of decision variables'
XU(N)     'upper bounds of decision variables'
EPSF      'alternative solution discrimination criterion';
```

Figure 4.9. Definition of the sets and parameters.

The sets and parameters above are read from the GDX file named “Problem_Data” into GAMS using the following directives.

```
$gdxin Problem_Data
$load N M c b2 XL XU EPSF A2 T
$gdxin
```

Figure 4.10. Reading the data into GAMS.

Here, ‘$gdxin’ closes the current GDX input file, if any. The filename that follows it specifies the GDX file from where data are read in. ‘$load’ statement lists all symbols in the GDX file to be read. Lastly, ‘$gdxin’ statement closes the current GDX file,

“Problem_Data”. When GAMS solves the problem, the results are written to the GDX file using the following statement.

```
execute_unload 'Solution_Data', P, XSOL;
```

Figure 4.11. Writing the results into the GDX file.

where, ‘`P`’ keeps the objective function values and ‘`XSOL`’ keeps the values of the decision variables obtained in all iterations. “Solution_Data” is the name of the GDX file in which the data, ‘`P`’ and ‘`XSOL`’, are written.

The results obtained using the interface between GAMS and MATLAB are the same as those obtained in GAMS. The interface enables the user to utilize the powerful optimization tools of GAMS and data manipulation and visualization capabilities of MATLAB at the same time.

5. GAMS TO MATLAB CONVERSION

The Recursive MILP (RMILP) algorithm provided in Section 3 was invented by Prof. Grossmann's group (Lee *et al.*, 2000; Phalakornkule *et al.*, 2001), however the code written in GAMS was not generic. It was problem specific and must be changed for each different problem. As several ChE 492 Project works, Prof. Akman generated a generic GAMS code that can be used even by a novice LP user without any need for modification to the RMILP part of the code by the user. In this thesis, the RMILP algorithm is implemented in MATLAB. In this chapter, first, the GAMS code is explained then, an explanation to the MATLAB code is provided.

An LP problem can be presented as Equation 2.1, which is identically provided below as Equation 5.1.

$$\begin{aligned} \min \quad & Z = \boldsymbol{c}^T \boldsymbol{x} \\ \text{s.t.} \quad & \boldsymbol{A}^1 \boldsymbol{x} = \boldsymbol{b}^1 \\ & \boldsymbol{A}^2 \boldsymbol{x} \leq \boldsymbol{b}^2 \\ & \boldsymbol{x}^L \leq \boldsymbol{x} \leq \boldsymbol{x}^U; \quad \boldsymbol{x} \in \boldsymbol{R}^n \end{aligned} \tag{5.1}$$

where, Z is an objective function, $\boldsymbol{c}$ is an n-dimensional column vector of objective function cost coefficients, $\boldsymbol{x}$ is an n-dimensional column vector of decision variables, $\mathbf{A^1}$ is a $k \times n$ matrix of equality constraint coefficients, $\boldsymbol{b}^1$ is a k-dimensional column vector of right hand side values of the equality constraints, $\boldsymbol{A}^2$ is an $m \times n$ matrix of inequality constraint coefficients, $\boldsymbol{b}^2$ is an m-dimensional column vector of right hand side values of the inequality constraints, $\boldsymbol{x}^L$ and $\boldsymbol{x}^U$ are n-dimensional column vectors of lower and upper bounds of the decision variables.

Equation 5.1, which is in non-canonical form, can be formulated in canonical form with the introduction of slack variables $\boldsymbol{s}$ of dimension $m \times 1$ to the inequalities and by the conversion of all variables into non-negative variables. To convert variables into non-negative ones, slack variables $\boldsymbol{s}^L$ and $\boldsymbol{s}^U$, n-dimensional column vectors, are introduced for the lower bound and upper bound of $\boldsymbol{x}$, respectively. At this point, it is assumed that $\boldsymbol{x}^L$ and

$\boldsymbol{x}^U$ are finite. Using $\boldsymbol{s}^L$ and $\boldsymbol{s}^U$, $\boldsymbol{x}$ is eliminated to obtain the following canonical form of LP.

$$\begin{aligned} \min \quad & \boldsymbol{c}^T\boldsymbol{s}^L + \boldsymbol{c}^T\boldsymbol{x}^L \\ \text{s.t} \quad & \boldsymbol{A}^1\boldsymbol{s}^L = \boldsymbol{b}^1 - \boldsymbol{A}^1\boldsymbol{x}^L \\ & \boldsymbol{A}^2\boldsymbol{s}^L + \boldsymbol{s} = \boldsymbol{b}^2 - \boldsymbol{A}^2\boldsymbol{x}^L \\ & \boldsymbol{s}^L + \boldsymbol{s}^U = \boldsymbol{x}^U - \boldsymbol{x}^L \\ & \boldsymbol{s}^L,\ \boldsymbol{s}^U, \boldsymbol{s} \geq \boldsymbol{0} \\ & \boldsymbol{s}^L, \boldsymbol{s}^U \in \boldsymbol{R}^n,\ \boldsymbol{s} \in \boldsymbol{R}^m \end{aligned} \tag{5.2}$$

where, n is the number of decision variables and m is the number of inequality constraints in Equation 5.1.

The last term $\boldsymbol{c}^T\boldsymbol{x}^L$ in the objective function is constant and can be dropped. However, it must be added to the value of the optimal objective function value at the end of optimization to recover the true value of the objective function. In Equation 5.2, all the constraints are in the form of equalities and all the variables are non-negative variables.

The LP obtained by writing Equation 5.2 in canonical matrix form is as follows:

$$\begin{aligned} \min \quad & Z = \boldsymbol{\alpha}^T\boldsymbol{z} \\ \text{s.t.} \quad & \boldsymbol{B}\boldsymbol{z} = \boldsymbol{q} \\ & \boldsymbol{z} \geq \boldsymbol{0} \end{aligned} \tag{5.3}$$

where, $\boldsymbol{z}$ and $\boldsymbol{\alpha}$ are $(2n+m)$-dimensional column vectors, $\boldsymbol{B}$ is an $(k+m+n) \times (2n+m)$ coefficient matrix, and $\boldsymbol{q}$ is an $(k+m+n)$-dimensional column vector composed of right hand side values of the equality constraints. n is the number of decision variables, k is the number of equality constraints, and m is the number of inequality constraints in the original LP problem provided in Equation 5.1.

The RMILP algorithm (Lee *et al.*, 2000) for finding all the alternate optima is given in Equation 3.7, which is identically provided below as Equation 5.4.

Step 1. Set iteration counter $K = 1$. Solve Equation 5.3. Define the set NZ^1 and the optimal objective $(Z^1)^*$.

Step K, for $K \geq 2$

(a) Solve the master problem

$$\begin{aligned} \min \quad & Z = \boldsymbol{\alpha}^T \boldsymbol{z} \\ \text{s.t.} \quad & \boldsymbol{B}\boldsymbol{z} = \boldsymbol{q} \\ & \textstyle\sum_{i \in NZ^{K-1}} y_i \geq 1 \\ & \textstyle\sum_{i \in NZ^K} w_i \leq |NZ^k| - 1, \;\; k = 1, 2, \ldots, K-1 \\ & 0 \leq z_i \leq U w_i, \;\; i \in I \\ & y_i + w_i \leq 1, \;\; i \in NZ^{K-1} \\ & \boldsymbol{z} \geq \boldsymbol{0} \end{aligned} \tag{5.4}$$

(b) Define set NZ^K and continue until $Z^K > (Z^1)^*$.

Parameters Z, $\boldsymbol{\alpha}^T$, $\boldsymbol{z}$, $\boldsymbol{B}$, and $\boldsymbol{q}$ are defined in Equation 5.3. z_i is the ith element of the vector of decision variables, $\boldsymbol{z}$. The $0-1$ variable y_i is defined, to perform the selection of a new basis, for each variable z_i that is a non-zero basic variable, $i \in NZ^{K-1}$, (NZ^K being the set of non-zero basic variables at the Kth iteration) at the previous iteration, $K-1$. z_i is selected to become non-basic when y_i is set to 1, and when y_i is 0, z_i remains in the basis. The $0-1$ variable w_i is defined, to ensure that all alternate bases are generated, which has a value of 1 if variable z_i is non-zero at the current iteration, K. When variable z_i is zero at iteration K, w_i takes the value of 0. U is a valid upper bound for all z_i. The selection of upper bound is important. If U is not big enough, then the results may be wrong. If U is too big, then numerical problems may arise due to machine precision. Generally, U should be sufficiently larger than the largest z_i value to make it sure that when a component of w_i becomes one U becomes the proper upper bound for all components of z_i.

The generic GAMS code developed by Prof. Akman is provided below. Problem specific data are not presented. First, the generic LP is formulated as canonical matrix form, Equation 5.3. In the first part of the code, this canonical LP is solved and the optimum solution is obtained with the global lowest objective function value that is unique (since LP is convex programming). This constitutes the first step of the algorithm provided in Equation 5.4. In the second part of the code, a MILP is constructed as in Equation 5.4 with the addition of constraints to the LP in canonical matrix form and it is solved recursively until the termination criterion is met.

In the beginning of the generic GAMS code, sets are defined. Set `K` is the row dimension of the coefficient matrix $\boldsymbol{B}$, $(k+m+n)$, giving the number of equality constraints in the canonical matrix form. Set `N` is the row dimension of the decision variable vector $\boldsymbol{z}$, $(2n+m)$, which gives the number of decision variables in Equation 5.3. Set `I` is the number of decision variables in the original LP, which is n. Set `T` is the maximum number of iterations to be performed. The corresponding GAMS code section is as follows:

```
SET K row dimension of B: (n+k+m) / /;
SET N row dimension of z: (2n+m) / /;
SET I(N) number of decision variables / /;
SET T maximum number of MILP iterations / /;
ALIAS(T,TC);
```

Figure 5.1. Definition of the sets.

where, set `TC` is the control set for set `T`.

A big number for U in Equation 5.4, `M`, and the value for ε in Equation 5.4, `EPSF`, the alternative solution discrimination criterion ($Z^K - Z^{K-1} \leq \varepsilon$), are defined to be used in the second (RMILP) part of the code. The coefficient matrices and vectors in canonical matrix form are also defined as follows:

```
SCALAR M big-M parameter / /;
PARAMETER a(N) vector a / /;
PARAMETER q(K) vector q / /;
TABLE B(K,N) matrix B;
PARAMETER XL(I) lower bounds of decision variables / /;
PARAMETER EPSF alternative solution discrimination
criterion / /;
```

Figure 5.2. Definition of the coefficient matrices and vectors in canonical matrix form.

where, `a(N)` represents the coefficient vector, $\boldsymbol{\alpha}^T$, in the objective function equation, `q(K)` is the vector of right hand side values, $\boldsymbol{q}$, in the equality constraints, and `B(K,N)` is the coefficient matrix, $\boldsymbol{B}$, in the equality constraints in Equation 5.3.Parameter `XL(I)` is the lower bounds, $\boldsymbol{x}^L$, for the decision variables in the original LP, given in Equation 5.1.

The objective function value, `F`, which is denoted as Z in canonical matrix form, Equation 5.3, and the vector of decision variables, `Z(N)`, denoted as $\boldsymbol{z}$ in the canonical matrix form are defined and the equality constraints are declared in the code section provided below.

```
VARIABLE F objective function value;
POSITIVE VARIABLE Z(N) decision variables (vector z);
EQUATIONS CANONICALEQ(K) canonical form equalities,
          OBJ objective function definition;
CANONICALEQ(K) .. SUM(N, (B(K,N)*Z(N))) =E= q(K);
OBJ .. SUM(N, (a(N)*Z(N))) + SUM(I, (a(I)*xL(I))) =E= F;
```

Figure 5.3. Definition of variables and declaration of the equality constraints.

where, `CANONICALEQ(K)` corresponds to the equality constraints and `OBJ` represents the objective function definition in Equation 5.3. In the objective function definition, the term $\boldsymbol{c}^T\boldsymbol{x}^L$ is included to obtain the actual objective function value of the original LP.

In the following part of the code, the initial canonical LP model is constituted and solved by minimizing the objective function value. The LP solver is selected as XA in this example code. The amount of resource units (in CPU seconds) to solve the model is recorded and displayed, as well as the objective function value and the decision variables.

```
MODEL CANONICALLP / CANONICALEQ, OBJ /;
OPTION LP = XA;
PARAMETER TOTAL_CPU total CPU used;
TOTAL_CPU = 0;
SOLVE CANONICALLP USING LP MINIMIZING F;
TOTAL_CPU = CANONICALLP.RESUSD;
DISPLAY TOTAL_CPU;
DISPLAY F.L, Z.L
```

Figure 5.4. Model construction for the initial canonical LP.

The optimal solution $\boldsymbol{x}^*$, given by `X(I)`, is recovered from $\boldsymbol{z}^*$, given by `Z.L(I)`, with the addition of lower bound, `XL(I)`, to `Z.L(I)` and displayed in the following code.

```
PARAMETER X(I) original decision variables;
X(I) = Z.L(I) + XL(I);
DISPLAY X;
```

Figure 5.5. Recovering the optimal solution.

At this point, the LP problem in canonical matrix form is solved, values of the objective function and decision variables are obtained, and Step 1 of the RMILP algorithm, provided in Equation 5.4, is completed. At the second step of the algorithm, binary variables `Y(N)` and `W(N)` keeping the values of y_i and w_i in Equation 5.4 are defined to force the selection of a new basis and to ensure that all alternate bases are generated, respectively.

```
BINARY VARIABLES Y(N) control variable in the MILP part,
                 W(N) control variable in the MILP part;
```

Figure 5.6. Definition of the binary variables.

New parameters are defined to keep the iteration results. Parameter `ZT(T,N)` is used to keep the decision variables of the canonical matrix form, `Z.L(N)`, in each iteration, `T. FOPT` is the current optimum objective function value, `F.L`, in the first iteration of which is the value obtained by solving the LP in the first part of the code. `FVALUE(T)` keeps the objective function value obtained in each iteration, `T. MSTAT(T)`

keeps the model status code in each iteration, `T`. `YT(T,N)` and `WT(T,N)` keep the values of `Y(N)` and `W(N)` in each iteration, `T`. `ITER` is the RMILP iteration number. `NZ(T)` is the number of non-zero basic variables in each iteration, `T`. `XSOL(T,I)` keeps, in each iteration, `T`, the values of the optimal decision variables $\boldsymbol{x}^*$, given by `X(I)`, recovered from $\boldsymbol{z}^*$, given by `Z.L(I)`, with the addition of lower bound, `XL(I)`, to `Z.L(I)`.

```
PARAMETER ZT(T,N),FOPT,FVALUE(T) obj fct values,MSTAT(T);
PARAMETER YT(T,N),WT(T,N);
PARAMETER ITER MILP Iteration;
PARAMETER NZ(T) set of the non-zero basic variables;
PARAMETER XSOL(T,I) recovered optimal solution x* from z*;
```

Figure 5.7. Definition of parameters to keep the iteration results.

The results obtained in the first part of the code, which is also the first iteration, by solving the initial LP are kept in the following parameters. Since `Y(N)` values in each iteration `T` are defined considering the values of the decision variables in the previous iteration `T-1`, they are set equal to zero in the first iteration.

```
ZT('1',N)    = Z.L(N);
YT('1',N)    = 0;
FOPT         = F.L;
FVALUE('1')  = F.L;
MSTAT('1')   = CANONICALLP.MODELSTAT;
```

Figure 5.8. Keeping the results of the first iteration.

Parameters `WT(T,N)` and `NZ(T)` are defined below for the first iteration. If decision variable `ZT('1',N)` is 0, `WT('1',N)` is also 0. If `ZT('1',N)` is nonzero, `WT('1',N)` becomes 1. `NZ('1')`, the number of nonzero basic variables equals the sum of `WT('1',N)` values for all `N`, in the first iteration.

```
WT('1',N)$(ZT('1',N) EQ 0.) = 0;
WT('1',N)$(ZT('1',N) NE 0.) = 1;
NZ('1') = SUM(N,WT('1',N));
```

Figure 5.9. Definition of parameters `WT(T,N)` and `NZ(T)`.

`XSOL(T,I)` is defined below, for the first iteration, as the result of the addition of lower bound, `XL(I)`, to `ZT('1',I)`.

```
XSOL('1',I) = ZT('1',I) + XL(I);
DISPLAY XSOL;
```

Figure 5.10. Definition of parameter `XSOL(T,I)`.

Equation that forces `W(N)` to be 1 if `Z(N)` is nonzero, which is given below as Equation 5.4, is named as `ACUT(T,N)`. This equation is valid for `N` decision variables in each iteration, `T`. Equation called as YSUM and given below as Equation 5.5 means that at least one of the nonzero basic variables in the $(K-1)^{th}$ solution is selected and is set to 0 (non-basic) in the current iteration, K. At each iteration, one `YSUM` equation is present.

$$0 \leq z_i \leq Uw_i, \;\; i \in I \tag{5.4}$$

$$\sum_{i \in NZ^{K-1}} y_i \geq 1 \tag{5.5}$$

These equations are declared in the following code section.

```
EQUATIONS ACUT(T,N), YSUM(T);
ACUT(T,N)$(ORD(T) EQ ITER) .. Z(N) =L= M*W(N);
YSUM(T)$(ORD(T) EQ ITER) .. SUM(N,(Y(N))$(WT(T,N) EQ 1)) =G= 1.0;
```

Figure 5.11. Declaration of Equations 5.4 and 5.5.

Equations named as `BASISC(T)` and `MNB(T,N)` in the GAMS code are given below as Equations 5.6 and 5.7. There are one `BASISC` equation and N `MNB` equations in each iteration, `T`. `BASISC(T)` equation eliminates from consideration at least one of the non-zero variables of the basis that were found in the previous iterations, $k = 1, 2, \ldots, K -$

1. `MNB(T,N)` equation states that the sum of the variables `Y(N)` and `W(N)` should be less than or equal to 1 in each iteration, `T`.

$$\sum_{i \in NZ^K} w_i \leq |NZ^k| - 1, \;\; k = 1, 2, \ldots, K-1 \tag{5.6}$$

$$y_i + w_i \leq 1, \;\; i \in NZ^{K-1} \tag{5.7}$$

These equations are declared in the following code section.

```
EQUATIONS BASISC(T), MNB(T,N);
BASISC(T)$(ORD(T) LE ITER) .. SUM(N,W(N)$(WT(T,N) EQ 1))=L=NZ(T)-1;
MNB(T,N)$((ORD(T) EQ ITER) AND (WT(T,N) EQ 1)) .. Y(N) +
W(N) =L= 1;
```

Figure 5.12. Declaration of Equations 5.6 and 5.7.

The RMILP model is declared including all of the equations presented up to this point. The solver is selected as XA in this example code. Options `OPTCA`, absolute termination criterion and `OPTCR`, relative termination criterion for MIP are defined. The corresponding GAMS code is as follows:

```
MODEL RECMILP / ALL /;
OPTION MIP = XA;
OPTION OPTCA = 0.0;
OPTION OPTCR = 0.0;
```

Figure 5.13. Declaration of the RMILP model.

Iteration loop begins, where the index used for the iteration number is `TC`. In each iteration, the RMILP is solved by minimizing the objective function value. The total amount of resource units (in CPU seconds) to solve the LP and RMILP models is calculated and displayed in the following code section:

```
LOOP(TC,
ITER = ORD(TC);
SOLVE RECMILP USING MIP MINIMIZING F;
TOTAL_CPU = TOTAL_CPU + RECMILP.RESUSD;
DISPLAY TOTAL_CPU;
```

Figure 5.14. Solving RMILP iteratively.

The results of the iterations are kept in the following parameters.

```
FVALUE(T)$(ORD(T) EQ (ITER+1)) = F.L;
ZT(T,N)  $(ORD(T) EQ (ITER+1)) = Z.L(N);
YT(T,N)  $(ORD(T) EQ (ITER+1)) = Y.L(N);
WT(T,N)  $(ORD(T) EQ (ITER+1)) = W.L(N);
NZ(T)    $(ORD(T) EQ (ITER+1)) = SUM(N,WT(T,N));
MSTAT(T) $(ORD(T) EQ (ITER+1)) = RECMILP.MODELSTAT;
```

Figure 5.15. Keeping the iteration results.

At this point, if the objective function value obtained in the current iteration is less than or equal to the previous objective function value, `FOPT`, plus `EPSF`, ($Z^K - Z^{K-1} \leq \varepsilon$), this means that another alternate or next best solution is obtained and the program will continue to search for yet another alternate or next best solution until the current objective function value fails to satisfy the termination criterion. Parameters `XSOL` and `FOPT` are updated and displayed, as well as the parameters below. Here, parameter `FOPT` is the previous objective function value. When it is updated, it becomes the current objective function value. However, it can also be defined as the minimum objective function value obtained up to this point such that `FOPT=MIN(F.L,FOPT)`. When this is the case, the termination criterion requires the current objective function value to be less than or equal to the minimum objective function value obtained up to current iteration plus `EPSF` ($Z^K - (Z^1)^* \leq \varepsilon$).

```
IF((F.L LE (FOPT+EPSF)),
    DISPLAY ITER, MSTAT;
    DISPLAY ZT, YT, WT, NZ;
    DISPLAY Y.L, W.L;
    DISPLAY Z.L, F.L;
    XSOL(T,I)$((ORD(T) EQ (ITER+1))) = ZT(T,I) + XL(I);
    DISPLAY XSOL;
    FOPT = F.L;
    DISPLAY FOPT;
    DISPLAY MSTAT, TOTAL_CPU;
    DISPLAY XSOL, FVALUE;
```

Figure 5.16. Determining whether the program will continue.

If the objective function value obtained in the current iteration is greater than the previous objective function value, FOPT, plus EPSF, ($Z^K - Z^{K-1} \leq \varepsilon$), this means that there is no more alternate (if EPSF is 0) or next best (if EPSF is greater than 0) solution and the iterations stop. This condition is given in the following code section:

```
ELSE
    ABORT " >>>>> NO MORE ALTERNATE OPTIMA";
);
```

Figure 5.17. Stopping the iterations.

If the model status for the RMILP is 10, this means that the problem is integer infeasible and the iterations stop and this condition is given in the following code section:

```
IF((RECMILP.MODELSTAT EQ 10),
    DISPLAY MSTAT, TOTAL_CPU;
    ABORT ">>>>> PROBLEM IS NOW INTEGER INFEASIBLE";
); );
```

Figure 5.18. Detecting integer infeasibility.

The last solution is generally not one of the optimum solutions since the loop stops when the last objective function value is greater than the previous objective function value by EPSF.

MATLAB implementation of the above generic GAMS code developed by Prof. Akman consists of two parts namely, a function where LP and RMILP problems are solved and a main part where the function is called. The user only enters the problem specific data and termination criteria in the main part. The same function part can be used for all problems without any modification.

In case of the MATLAB implementation, the problem can be in canonical matrix form given by Equation 5.3 or in non-canonical form provided in Equation 5.1.

The main part of the MATLAB code, where the problem specific data and termination criteria are entered, is given below, as well as the corresponding part of the GAMS code. Problem specific data are not presented. The GAMS code section is as follows:

```
SCALAR M big-M parameter / /;
PARAMETER a(N) vector a / /;
PARAMETER q(K) vector q / /;
TABLE B(K,N)   matrix B ;
PARAMETER XL(I) lower bounds of decision variables / /;
PARAMETER EPSF alternative solution discrimination criterion / /;
```

Figure 5.19. Entering the problem specific data.

The corresponding MATLAB code section is provided below. The coefficient matrices and vectors in canonical matrix form are defined where, `a` corresponds to parameter `a(N)` in the GAMS code, `B` corresponds to table `B(K,N)`, `q` corresponds to parameter `q(K)`, `XL` corresponds to parameter `XL(I)`, `EPSF` corresponds to parameter `EPSF`, and `BM` corresponds to parameter `M`. Here, `T` is the maximum number of iterations which is defined as a set in the GAMS code.

```
a = [.];
B = [.];
q = [.];
XL = [ ];
T = ;
EPSF = ;
BM = ;
```

Figure 5.20. Definition of the coefficient matrices and vectors in canonical matrix form.

In the following part of the MATLAB code section, parameter `InputType` is defined such that it is 1 if the LP is given in canonical matrix form and 0 if it is given in non-canonical form. In this code, the LP is in canonical matrix form and vectors and matrices are defined accordingly. If the LP is in non-canonical form, then parameters `c`, `A1`, `A2`, `b1`, `b2`, `XL`, `XU` are defined which correspond to $\boldsymbol{c}$, $\boldsymbol{A}^1$, $\boldsymbol{A}^2$, $\boldsymbol{b}^1$, $\boldsymbol{b}^2$, $\boldsymbol{x}^L$, and $\boldsymbol{x}^U$ in Equation 5.1.

```
InputType = 1;
if InputType == 0
    a = 0;
    B = 0;
    q = 0;
end
if InputType == 1
    A1 = 0;
    A2 = 0;
    b1 = 0;
    b2 = 0;
    c = 0;
    XU = 0;
end
[XSOL,FVALUE]=RecMILP(a,B,q,c,A1,A2,b1,b2,XL,XU,T,EPSF,B,InputType);
```

Figure 5.21. Definition of parameter `InputType`.

The function part of the code is written for canonical matrix form. Therefore, when the LP is in non-canonical form, it is automatically transformed into the canonical matrix form at the beginning of the function part. The function is named `RecMILP` in this example code. Its inputs are `a`, `B`, `q`, `c`, `A1`, `A2`, `b1`, `b2`, `XL`, `XU`, `T`, `EPSF`, `BM`,

`InputType`. As seen in the above code, if the LP is in canonical matrix form, parameters `c`, `A1`, `A2`, `b1`, `b2`, and `XU` are set equal to 0 and when it is in non-canonical form, parameters `a`, `B`, `q` are set equal to 0. The outputs of the function are `XSOL` and `FVALUE`. `XSOL` keeps the values of the decision variables obtained in the iterations and `FVALUE` keeps the corresponding objective function values.

In the function part of the code an '`if`' statement is used such that when the LP is in non-canonical form, given in Equation 5.1, which means that `InputType==0`, it is transformed into the canonical matrix form, given in Equation 5.3, where,

$$\boldsymbol{z} = \begin{bmatrix} \boldsymbol{s}^L_{n\times 1} \\ \boldsymbol{s}^U_{n\times 1} \\ \boldsymbol{s}_{m\times 1} \end{bmatrix},\ \boldsymbol{\alpha} = \begin{bmatrix} \boldsymbol{c}_{n\times 1} \\ \boldsymbol{0}_{n\times 1} \\ \boldsymbol{0}_{m\times 1} \end{bmatrix},\ \boldsymbol{B} = \begin{bmatrix} \boldsymbol{A}^{\boldsymbol{1}}_{k\times n} & \boldsymbol{0}_{k\times n} & \boldsymbol{0}_{k\times m} \\ \boldsymbol{A}^{\boldsymbol{2}}_{m\times n} & \boldsymbol{0}_{m\times n} & \boldsymbol{I}_{m\times m} \\ \boldsymbol{I}_{n\times n} & \boldsymbol{I}_{n\times n} & \boldsymbol{0}_{n\times m} \end{bmatrix},\ \boldsymbol{q} = \begin{bmatrix} (\boldsymbol{b}^1 - \boldsymbol{A}^1\boldsymbol{x}^L)_{k\times 1} \\ (\boldsymbol{b}^2 - \boldsymbol{A}^2\boldsymbol{x}^L)_{m\times 1} \\ (\boldsymbol{x}^U - \boldsymbol{x}^L)_{n\times 1} \end{bmatrix}$$

where, n is the number of decision variables, k is the number of equality constraints and m is the number of inequality constraints in Equation 5.1. If the LP is in canonical matrix form, then the '`if`' statement is skipped. Here, `R` is defined as the row dimension of the coefficient vector `c`, `P` is defined as the row dimension of the coefficient matrix `A1` of the equality constraints in Equation 5.1, and `S` is defined as the row dimension of the coefficient matrix `A2` of the inequality constraints in Equation 5.1. The corresponding MATLAB code section is provided below.

```
function [XSOL,FVALUE] =
         RecMILP(a,B,q,c,A1,A2,b1,b2,XL,XU,T,EPSF,BM,InputType)
if (InputType==0)
    [R,junk] = size(c);
    [P,junk] = size(A1);
    [S,junk] = size(A2);
    a = [c;zeros(R,1);zeros(S,1)];
    q = [(b1-A1*XL); (b2-A2*XL); (XU-XL)];
    B = [A1       zeros(P,R)    zeros(P,S)
         A2       zeros(S,R)    eye(S)
         eye(R)   eye(R)        zeros(R,S)];
end
```

Figure 5.22. Constructing the parameters of canonical matrix form.

There is no corresponding part of the above code section in the GAMS code.

Sets are defined in the GAMS code and the corresponding MATLAB code section is provided below. Set `K` is the row dimension of the coefficient matrix `B`, which is equal to the row dimension of the vector `q`. Set `N` defines the number of decision variables in canonical matrix form. Set `I` is the number of decision variables in the original LP. The GAMS code section is as follows:

```
SET K row dimension of B: (n+k+m) / /;
SET N row dimension of z: (2n+m) / /;
SET I(N) number of decision variables / /;
SET T maximum number of MILP iterations / /;
ALIAS(T,TC);
```

Figure 5.23. Definition of the sets in GAMS.

The corresponding MATLAB code section is as follows:

```
[N,junk] = size(a);
[K,junk] = size(q);
[I,junk] = size(XL);
```

Figure 5.24. Definition of the sets in MATLAB.

In the GAMS code section provided below, variable `F` is defined to be the objective function value and positive variable `Z(N)` is defined as the values of decision variables in canonical matrix form. Equations are declared, the model is constructed, the LP solver is selected, CPU time is recorded and displayed, as well as the objective function value and values of the decision variables.

```
VARIABLE F objective function value;
POSITIVE VARIABLE Z(N) decision variables (vector z);
EQUATIONS CANONICALEQ(K) canonical form equalities,
          OBJ objective function definition;
CANONICALEQ(K) .. SUM(N, (B(K,N)*Z(N))) =E= q(K);
OBJ .. SUM(N, (a(N)*Z(N))) + SUM(I, (a(I)*xL(I))) =E= F;
MODEL CANONICALLP / CANONICALEQ, OBJ /;
OPTION LP = XPRESS;
PARAMETER TOTAL_CPU total CPU used;
TOTAL_CPU = 0;
SOLVE CANONICALLP USING LP MINIMIZING F;
TOTAL_CPU = CANONICALLP.RESUSD;
DISPLAY TOTAL_CPU;
DISPLAY F.L, Z.L;
```

Figure 5.25. Constructing and solving the model in GAMS.

The corresponding MATLAB code section is provided below:

```
ZL = zeros(N,1);
options = optimset('LargeScale', 'off', 'Simplex', 'on');
[Z, F1, exitflag, output, lambda] =
linprog(a, [], [], B, q, ZL, [], [], options);
for i = 1:1:I
C(i) = a(i);
end
F = F1 + C*XL;
```

Figure 5.26. Constructing and solving the model in MATLAB.

Here, parameter `ZL` is defined to be an `N`-dimensional column vector of zeros, which consists of the lower bounds of the decision variables in canonical matrix form. Simplex method option is selected to solve the LP and the LP is solved using '`linprog`' command by minimizing the objective function value, `F1`. `Z` is an `N`-dimensional decision variable vector corresponding to `Z(N)` in the GAMS code. `F1` is the objective function value of the LP in canonical matrix form given by Equation 5.3. Here, `F1` corresponds to $\boldsymbol{c}^T\boldsymbol{s}^L$ and to obtain the objective function value of the original LP, the term $\boldsymbol{c}^T\boldsymbol{x}^L$ is added to `F1`, resulting in parameter `F`. Parameter `F` corresponds to the variable `F` in the GAMS code.

In the following GAMS code section, original decision variables `X(I)` are recovered from `Z.L(I)` by the addition of `XL(I)` and displayed. This part of the code has no corresponding part in the MATLAB code. Later, parameter `XSOL` will be defined in GAMS and MATLAB codes which keeeps the original decision variables obtained in the iterations. Here, parameter `X(I)` corresponds to `XSOL` value in the first iteration.

```
PARAMETER X(I) original decision variables;
X(I) = Z.L(I) + XL(I);
DISPLAY X;
```

Figure 5.27. Recovering original decision variables in GAMS.

In the following GAMS code section, the binary variables and parameters are defined. The values of the parameters in the first iteration are declared.

```
BINARY VARIABLES Y(N) control variable in the MILP part,
                 W(N) control variable in the MILP part;
PARAMETER ZT(T,N),FOPT,FVALUE(T) obj fct values,MSTAT(T);
PARAMETER YT(T,N),WT(T,N);
PARAMETER ITER MILP Iteration;
PARAMETER NZ(T) set of the non-zero basic variables;
PARAMETER XSOL(T,I) recovered optimal solution x* from z*;
ZT('1',N)   = Z.L(N);
YT('1',N)   = 0;
FOPT        = F.L;
FVALUE('1') = F.L;
MSTAT('1')  = CANONICALLP.MODELSTAT;
```

Figure 5.28. Defining binary variables and parameters in GAMS.

The corresponding MATLAB code section is provided below.

```
i = 1:1:N;
ZT(1,i) = Z(i);
YT(1,i) = 0;
FOPT = F;
FVALUE(1) = F;
```

Figure 5.29. Defining binary variables and parameters in MATLAB.

Parameters ZT, Z, YT, FOPT, F, and FVALUE have the same names as the corresponding parameters in the GAMS code.

Parameter WT, which is a matrix of 0-1 binary variables displaying the variables in the basic set, is defined in the following code sections. An element of the matrix WT(T,N) is 1 if the corresponding element of the matrix ZT(T,N) is non-zero. Parameter NZ is defined as the number of the non-zero basic variables.

```
WT('1',N)$(ZT('1',N) EQ 0.) = 0;
WT('1',N)$(ZT('1',N) NE 0.) = 1;
NZ('1') = SUM(N,WT('1',N));
```

Figure 5.30. Defining matrices WT and ZT in GAMS.

The corresponding MATLAB code section is provided below.

```
WT(1,1:N) = ZT(1,1:N) ~= 0;
NZ(1,1) = sum(WT,2);
```

Figure 5.31. Defining matrices WT and ZT in MATLAB.

The original decision variables given in XSOL are recovered from the parameter ZT with the addition of the lower bounds, XL, and displayed. The GAMS code section is as follows:

```
XSOL('1',I) = ZT('1',I) + XL(I);
DISPLAY XSOL;
```

Figure 5.32. Recovering original decision variables in GAMS.

The corresponding MATLAB code section is given below where, original decision variables given by XSOL are recovered and displayed in table format as well as the objective function value.

```
for k =1:1:I
    XSOL(1,k) = ZT(1,k) + XL(k);
end
fprintf('\nBASE CASE:');
fprintf('\nObjective Function value: %10.4f\n', F);
fprintf('====================================');
fprintf('\nVariable        Value\n');
for i = 1:1:I
fprintf('  X(%0.0f)   %12.4f\n',i,XSOL(i));
end
fprintf('=====================================\n');
```

Figure 5.33. Recovering original decision variables in MATLAB.

Equations `ACUT(T,N)` and `YSUM(T)` are declared in the following GAMS code.

```
EQUATIONS ACUT(T,N), YSUM(T);
ACUT(T,N)$(ORD(T) EQ ITER) .. Z(N) =L= M*W(N) ;
YSUM(T)$(ORD(T) EQ ITER) .. SUM(N,(Y(N))$(WT(T,N) EQ 1)) =G= 1.0;
```

Figure 5.34. Declaring Equations `ACUT(T,N)` and `YSUM(T)` in GAMS.

The corresponding MATLAB code section is given below where, a '`for`' loop is constructed as the iteration counter being `t` and the maximum umber of iterations allowed being `T`. At this point, binary variables `Y(N)` and `W(N)` given in the above code are also decision variables.

```
tic;
for t = 1:1:T
ACUTz = eye(N);
ACUTy = zeros(N);
ACUTw = -BM*eye(N);
ACUTB = [ACUTz ACUTy ACUTw];
ACUTq = zeros(N,1);
YSUMy(1:N) = WT(t,1:N) == 1;
YSUMz = zeros(1,N);
YSUMw = zeros(1,N);
YSUMB = [YSUMz YSUMy YSUMw];
YSUMq = 1;
```

Figure 5.35. Declaring Equations `ACUT(T,N)` and `YSUM(T)` in MATLAB.

where, ’tic‘ and ‘toc‘ statemets are used to measure the time it takes MATLAB to execute the RMILP iterations. In each iteration, there are N ACUT equations. ACUTz is the coefficient matrix of the decision variables given by vector ***z***, ACUTy is the coefficient matrix of binary variables ***y***, ACUTw is the coefficient matrix of binary variables ***w***, ACUTB is the combination of these three coefficient matrices, giving the overall coefficient matrix, and ACUTq is the vector of right hand side values of the ACUT equations. In each iteration there is one YSUM equation. The elements of YSUMy(1:N), the coefficient vector of binary variables ***y***, are defined as 1 if the corresponding elements of matrix WT(t,1:N) are 1 and 0 otherwise. YSUMz is the coefficient vector of the decision variables given by vector ***z***, YSUMw is the coefficient vector of binary variables ***w***, YSUMB is the combination of these three coefficient vectors, and YSUMq is th eright hand side of value of the YSUM equation.

Equations BASISC(T) and MNB(T,N) are declared in the following GAMS code section.

```
EQUATIONS BASISC(T), MNB(T,N);
BASISC(T)$(ORD(T) LE ITER) .. SUM(N,W(N)$(WT(T,N) EQ 1))=L=NZ(T)-1;
MNB(T,N)$((ORD(T) EQ ITER) AND (WT(T,N) EQ 1)) .. Y(N) + W(N) =L= 1;
```

Figure 5.36. Declaring Equations BASISC(T) and MNB(T,N) in GAMS.

The corresponding MATLAB code section is given below.

```
BASISCq = zeros(t,1);
BASISCz = zeros (t,N);
BASISCy = zeros (t,N);
BASISCw(1:t,1:N)=WT(1:t,1:N)==1;
BASISCq(1:t) = NZ(1:t) - 1;
BASISCB = [BASISCz BASISCy BASISCw];
MNBy = zeros(N); MNBw = zeros(N);
        for n = 1:1:N
                MNBy(n,n) = WT(t,n) == 1;
                MNBw(n,n) = WT(t,n) == 1;
        end
MNBz = zeros(N);
MNBB = [MNBz MNBy MNBw];
MNBq = ones(N,1);
```

Figure 5.37. Declaring Equations BASISC(T) and MNB(T,N) in MATLAB.

where, there are `t BASISC` equations in each iteration. `t` is the curent iteration number. `BASISCq` is the vector of right hand side values of `BASISC` equations. `BASISCz` is the coefficient matrix of the decision variables given by vector $\boldsymbol{z}$, `BASISCy` is the coefficient matrix of binary variables $\boldsymbol{y}$, `BASISCw` is the coefficient matrix of binary variables $\boldsymbol{w}$, the elements of which are equal to 1 if the corresponding elements of the matrix `WT(1:t,1:N)` are 1 and 0 otherwise. `BASISCB` is the combination of these three coefficient matrices, giving the overall coefficient matrix of `BASISC` equations. There are `N MNB` equations in each iteration. `MNBy` is the coefficient matrix of binary variables $\boldsymbol{y}$, `MNBw` is the coefficient matrix of binary variables $\boldsymbol{w}$ where, the elements `MNBy(n,n)` and `MNBw(n,n)` are 1 if the corresponding elements of the matrix `WT(t,n)` are 1 and 0 otherwise. `MNBz` is the coefficient matrix of the decision variables given by vector $\boldsymbol{z}$, `MNBB` is the combination of the matrices `MNBz`, `MNBy`, and `MNBw`, constituting the overall coefficient matrix of the `MNB` equations. `MNBq` is the vector of right hand side values of `MNB` equations.

In the GAMS code section provided below, the model is constructed including all of the equations declared up to this point and the MIP solver is selected as XA. Options `OPTCA`, absolute termination criterion and `OPTCR`, relative termination criterion for MIP are defined. The RMILP is then solved by minimizing the objective function value and the total CPU time is displayed.

```
MODEL RECMILP / ALL /;
OPTION MIP = XPRESS;
OPTION OPTCA = 0.0;
OPTION OPTCR = 0.0;
LOOP(TC,
ITER = ORD(TC);
SOLVE RECMILP USING MIP MINIMIZING F;
TOTAL_CPU = TOTAL_CPU + RECMILP.RESUSD;
DISPLAY TOTAL_CPU;
```

Figure 5.38. Constructing and solving RMILP in GAMS.

The corresponding MATLAB code section is given below.

```
ad = zeros(2*N,1);
newa = [a;ad];
newaa = -1*newa;
newZL = [ZL; zeros(2*N,1)];
Bd = zeros(K,2*N);
newB = [B Bd];
```

Figure 5.39. Updating the matrices and vectors.

where, `a` is the objective function coefficient vector in canonical matrix form. At this point, N-dimensional vectors of binary variables $\boldsymbol{y}$ and $\boldsymbol{w}$ are also decision variables and the coefficient matrices and vectors in canonoical matrix form are redefined including these new decision variables. `ad`, a $2 \times N$-dimensional vector of zeros, is the coefficient vector of variables $\boldsymbol{y}$ and $\boldsymbol{w}$ of the objective function in canonical matrix form. `newa` is the combination of vectors `a` and `ad`, which is the coefficient matrix of objective function in the RMILP problem constructed. `ZL` is the vector of lower bounds of decision variables in canonicalmatrix form. a $2 \times N$-dimensional vector of zeros, being the vector of lower bounds of newly added decision variables, is combined with `ZL` to construct the new vector of lower bounds. Matrix `Bd`, the coefficient matrix for newly added decision variables in the equality constraints in canonical matrix form, is combined with matrix `B` to obtain the overall coefficient matrix as `newB`.

```
OverallB = [newB;ACUTB;YSUMB;BASISCB;MNBB];
Overallq = [q;ACUTq;YSUMq;BASISCq;MNBq];
```

Figure 5.40. Constructing the overall coefficient matrix and vector of right hand sides.

`OverallB`, the overall coefficient matrix, is obtained by combining the coefficient matrices of all equations declared up to now. `Overallq`, the overall vector of right hand side values, is obtained by combining the vectors of right hand sidevalues of all equations declared up to now.

```
b = zeros(K,1);
ac = ones(N,1);
ac = -1*ac;
y = 1;
m = ones(N,1);
m = -1*m;
be = -1*ones(t,1);
e = [b;ac;y;be;m];
```

Figure 5.41. Determining equation types.

In the above MATLAB code section, equation types are determined in order to present them to ‘`lp_solve`’, which is a third party MILP solver employed in MATLAB codes. Value of `-1` represents the equations with “less than or equal to” sign, `0` represents the ones with “equal” sign, and `1` represents the ones with “greater than or equal to” sign. The vector `b` represents the of equalities in the canonical matrix form, `ac` represents the “less than or equal to” signs in `ACUT` equations, `y` represents the “greater than or equal to” signs in `YSUM` equations, `m` represents the “less than or equal to” signs in `MNB` equations, `be` represents the “less than or equal to” signs in `BASISC` equations, and `e` is the combination of them.

In the MATLAB code section provided below, the binary integer variables are identified with the definition of the parameter `xint`.

```
xint = N+1:3*N;
```

Figure 5.42. Defrining parameter `xint`.

In the MATLAB code section given below, ‘`lp_solve`’command is used to solve the RMILP maximizing the objective function value. However, the problem is contstructed as a minimization problem. Therefore, the vector `newaa`, the objective function of the maximization problem, is obtained as the multiplication of `newa`, the one of the minimization problem, with -1.

```
[F2,Z,duals,stat] =
lp_solve(newaa,OverallB,Overallq,e,newZL,[],xint);
F = F2 + C*XL;
F = -F;
```

Figure 5.43. Solving the RMILP.

where, the inputs to '`lp_solve`' are `newaa`, `OverallB`, `Overallq`, `e`, `newZL`, and `xint`. The empty vector, '`[]`', represents the upper bounds for the decision variables in the RMILP which are not given. The outputs of the function are `F2`, `Z`, `duals`, and `stat` where, `F2` is the objective function value, `Z` is the vector of decision variables, `duals` represents the dual variables, and `stat` represents the model status. After the problem is solved, the term, `C*XL`, is added to the objective function value, `F2` to obtain the actual objective function value, `F` and this value is multiplied by -1 to turn the result of the maximization problem into the minimization problem. Here, `C` is the coefficient vector of the objective function in the original LP model and `XL` is the vector of lower bounds.

If '`glpk`', which is another third party MILP solver employed in MATLAB codes, is to be used as the MIP solver, the following MATLAB code section will be used in place of the '`lp_solve`' related sections.

```
for i = 1:1:1+t+N+N+K;
    if e(i) == 1
        ctype(i,1) = 'L';
    end
    if e(i) == 0
        ctype(i,1) = 'S';
    end
    if e(i) == -1
        ctype(i,1) = 'U';
    end
end
j = 1:1:N;
vartype(j,1)  = 'C';
k = N+1:3*N;
vartype(k,1)  = 'B';
sense = 1;
newa1 = -1*newaa;
```

Figure 5.44. Defining parameters.

where, ctype determines the equation type as the vector e in the case of 'lp_solve'. It is an array of characters containing the sense of each constraint in the constraint matrix. For the constraints with "greater than or equal to" sign ctype becomes 'L', for the ones with "less than or equal to" sign it becomes 'U', and for the equalities it is 'S'. vartype is a column array containing the types of the variables. It is 'C' for continuous variables and 'B' for binary ones. sense determines the type of the problem to be solved. If the problem is a minimization, sense becomes 1 and if it is a maximization, sense becomes -1. Here, newaa, the coefficient vector in the objective function of RMILP in the case of maximization is multiplied by -1 to obtain the coefficient vector for minimization, newa1.

In the following MATLAB code section, the RMILP problem is solved using 'glpk' minimizing the objective function value.

```
[Z, F3, status, extra] = glpk (newa1, OverallB, Overallq,
newZL, [], ctype, vartype, sense);
F = F3 + C*XL;
```

Figure 5.45. Solving RMILP using 'glpk'.

where, the inputs to 'glpk' are newa1, OverallB, Overallq, newZL, ctype, vartype, and sense. The empty vector, '[]', represents the upper bounds for the decision variables in the RMILP which are not provided. The outputs of the function are F3, Z, status, and extra where, F3 is the objective function value, Z is the vector of decision variables, status represents status of the optimization, and extra is a data structure containing the dual variables, reduced costs, time in seconds used for solving the problem, and memory in kilobytes used for solving the problem.

If the model status for the RMILP is 10, this means that the problem is integer infeasible and the iterations stop and this condition is given in the following GAMS code section:

```
IF((RECMILP.MODELSTAT EQ 10),
    DISPLAY MSTAT, TOTAL_CPU;
    ABORT ">>>>> PROBLEM IS NOW INTEGER INFEASIBLE";
); );
```

Figure 5.46. Checking integer feasibility.

The problem feasibility is checked in the following MATLAB code section in the case of ‘`glpk`’. If `status` is 3, which means that the problem is infeasible, if it is 4, meaning that no feasible solution exists, if it is 110, meaning that there is no primal feasible solution iterations stop.

```
if (status == 3)
    fprintf('\n>>>>> PROBLEM IS NOW INFEASIBLE\n');
    break;
elseif (status == 4)
    fprintf('\n>>>>> NO FEASIBLE SOLUTION EXISTS \n');
    break;
elseif (status == 110)
    fprintf('\n>>>>> PROBLEM IS NOW PRIMAL INFEASIBLE\n');
     break;
end
```

Figure 5.47. Checking the problem feasibility in case of ‘`glpk`’.

The problem feasibility is checked in the following MATLAB code section in the case of ‘`lp_solve`’. If `stat` is 2, which means that the problem is infeasible iterations stop.

```
if (stat == 2)
    fprintf('\n>>>>> PROBLEM IS NOW INFEASIBLE\n');
    break;
 end
```

Figure 5.48. Checking the problem feasibility in case of ‘`lp_solve`’.

In case of the feasible problem, the results of the iterations are kept in the following parameters in the GAMS code.

```
FVALUE(T)$(ORD(T) EQ (ITER+1)) = F.L;
ZT(T,N)  $(ORD(T) EQ (ITER+1)) = Z.L(N);
YT(T,N)  $(ORD(T) EQ (ITER+1)) = Y.L(N);
WT(T,N)  $(ORD(T) EQ (ITER+1)) = W.L(N);
NZ(T)    $(ORD(T) EQ (ITER+1)) = SUM(N,WT(T,N));
MSTAT(T) $(ORD(T) EQ (ITER+1)) = RECMILP.MODELSTAT;
```

Figure 5.49. Keeping the results of the iterations in GAMS.

The corresponding MATLAB code section is provided below.

```
n = 1:1:N;
Y(n) = Z(n+N);
W(n) = Z(n+2*N);
n = 1:1:N;
ZT(t+1,n) = Z(n);
YT(t+1,n) = Y(n);
WT(t+1,n) = W(n);
Dummy2 = sum(WT,2);
NZ(t+1) = Dummy2(t+1);
```

Figure 5.50. Keeping the results of the iterations in MATLAB.

If the last objective function value which is obtained in the current iteration is less than or equal to the sum of `FOPT` and `EPSF`, ($Z^K - Z^{K-1} \leq \varepsilon$), the parameters are displayed, `FOPT` is updated as the current objective function value, `XSOL` is updated as the vector of decision variables of the original problem obtained in the current iteration, and iterations continue. If the last objective function value which is obtained in the current iteration is greater than the given sum that means there is no more alternate optimum (if `EPSF` is 0) or next best solution (if `EPSF` is greater than 0) and iterations stop. These cases are presented in the GAMS code section provided below.

```
IF((F.L LE (FOPT+EPSF)),
    DISPLAY ITER, MSTAT;
    DISPLAY ZT, YT, WT, NZ;
    DISPLAY Y.L, W.L;
    DISPLAY Z.L, F.L;
    XSOL(T,I)$((ORD(T) EQ (ITER+1))) = ZT(T,I) + XL(I);
    DISPLAY XSOL;
    FOPT = F.L;
    DISPLAY FOPT;
    DISPLAY MSTAT, TOTAL_CPU;
    DISPLAY XSOL, FVALUE;
ELSE
    ABORT " >>>>> NO MORE ALTERNATE OPTIMA";
);
```

Figure 5.51. Termination criterion in GAMS.

The corresponding MATLAB code section is provided below.

```
    if (F <= FOPT + EPSF)
        for i = 1:1:I
            XSOL(t+1,i) = ZT(t+1,i) + XL(i);
        end
        FVALUE(t+1) = F;
        FOPT = F;
    elseif (F >= FOPT + EPSF)
        fprintf('\n>>>>> NO MORE ALTERNATE OPTIMA\n');
        break;
    end
end
toc;
```

Figure 5.52. Termination criterion in MATLAB.

where, the objective function value and decision variables obtained in each iteration are displayed using the MATLAB code section given below.

```
fprintf('\nRECURSIVE MILP:\n');
for i = 1:1:t
    fprintf('==============');
end
fprintf('\n');
fprintf('Variable      ');
i = 1:1:t;
fprintf('Sol-%0.0f        ',i);
fprintf('\n');
for i = 1:1:t
    fprintf('==============');
end
fprintf('\n');
fprintf('ObjFun ');
i = 1:1:t;
fprintf('%13.4f',FVALUE(i));
fprintf('\n');
for i = 1:1:t
    fprintf('--------------');
end
fprintf('\n');
for j = 1:1:I
    fprintf(' X(%0.0f)  ',j);
    for k = 1:1:t
        fprintf('%13.4f', XSOL(k,j));
    end
    fprintf('\n');
end
for i = 1:1:t
    fprintf('==============');
end
fprintf('\n');
end
```

Figure 5.53. Displaying the results in MATLAB.

6. EXAMPLES

In this chapter, the tools developed as explained in previous chapters are applied to various representative LP problems taken from the literature. The problems are solved by GAMS and MATLAB, alternative and next best solutions are presented. The problem specifications, namely the number of decision variables, equality constraints, inequality constraints, alternate solutions, and K-best solutions obtained are provided for each LP problem in Table 6.1 below.

Table 6.1. Problem specifications for each LP.

Problem Name	# of decision variables	# of equality constraints	# of inequality constraints	# of alternate solutions obtained	# of K-best solutions obtained
Simple LP	2	-	3	2	4
Degenerate LP	3	-	5	4	6
Refinery Blending	6	4	3	2	4
Thermal Cracker	7	3	3	1	6
Bio FBA	33	29	1	9	5

6.1. A Simple LP

The simple two variable (x, y) LP problem, which was also considered in Chapter 2, is as follows:

$$\begin{aligned} \max_{x,y} \quad & \phi = 10x + 20y \\ \text{s.t.} \quad & x \leq 60 \\ & y \leq 50 \\ & x + 2y \leq 120 \\ & x, y \geq 0 \end{aligned} \tag{6.1}$$

The problem can be cast into the standard LP form that was given by Equation 2.1, which is identically provided below as Equation 6.2 (Lee *et al.*, 2000).

$$\begin{aligned} \min_{\boldsymbol{x}} \quad & \mathrm{Z} = \boldsymbol{c}^T \boldsymbol{x} \\ \text{s.t.} \quad & \boldsymbol{A}^1 \boldsymbol{x} = \boldsymbol{b}^1 \\ & \boldsymbol{A}^2 \boldsymbol{x} \leq \boldsymbol{b}^2 \\ & \boldsymbol{x}^L \leq \boldsymbol{x} \leq \boldsymbol{x}^U; \quad \boldsymbol{x} \in \boldsymbol{R}^n \end{aligned} \tag{6.2}$$

where, the vectors and matrices take the following values:

$$\boldsymbol{c}^T = [-10 \ -20], \ \boldsymbol{A}^2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 2 \end{bmatrix}, \ \boldsymbol{b}^2 = \begin{bmatrix} 60 \\ 50 \\ 120 \end{bmatrix}, \ \boldsymbol{x}^L = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \ \boldsymbol{x}^U = \begin{bmatrix} 500 \\ 500 \end{bmatrix} \tag{6.3}$$

where, $\boldsymbol{c}^T$ is obtained by multiplying the objective function coefficients in Equation 6.1 by -1, to turn the maximization problem into minimization, 500 is an arbitrarily given upper bound for x and y, and $\boldsymbol{A}^1 = [\,]$, $\boldsymbol{b}^1 = [\,]$.

The canonical form of LP problem is given by Equation 2.3, which is identically provided below as Equation 6.4.

$$\begin{aligned} \min_{\mathbf{z}} \quad & \mathrm{Z} = \boldsymbol{\alpha}^T \boldsymbol{z} \\ \text{s.t.} \quad & \boldsymbol{B}\boldsymbol{z} = \boldsymbol{q} \\ & \boldsymbol{z} \geq \mathbf{0} \end{aligned} \tag{6.4}$$

where, $\boldsymbol{z} = \begin{bmatrix} \boldsymbol{s}^L_{2\times1} \\ \boldsymbol{s}^U_{2\times1} \\ \boldsymbol{s}_{3\times1} \end{bmatrix}$, $\boldsymbol{\alpha} = \begin{bmatrix} \boldsymbol{c}_{2\times1} \\ \mathbf{0}_{2\times1} \\ \mathbf{0}_{3\times1} \end{bmatrix}$, $\boldsymbol{B} = \begin{bmatrix} \boldsymbol{A}^2_{3\times2} & \mathbf{0}_{3\times2} & \boldsymbol{I}_{3\times3} \\ \boldsymbol{I}_{2\times2} & \boldsymbol{I}_{2\times2} & \mathbf{0}_{2\times3} \end{bmatrix}$, $\boldsymbol{q} = \begin{bmatrix} (\boldsymbol{b}^2 - \boldsymbol{A}^2\boldsymbol{x}^L)_{3\times1} \\ (\boldsymbol{x}^U - \boldsymbol{x}^L)_{2\times1} \end{bmatrix}$

For this problem, the vectors and matrices take the following values:

$$\boldsymbol{\alpha} = \begin{bmatrix} -10 \\ -20 \\ - \\ 0 \\ 0 \\ - \\ 0 \\ 0 \\ 0 \end{bmatrix}, \ \boldsymbol{B} = \begin{bmatrix} 1 & 0 & | & 0 & 0 & | & 1 & 0 & 0 \\ 0 & 1 & | & 0 & 0 & | & 0 & 1 & 0 \\ 1 & 2 & | & 0 & 0 & | & 0 & 0 & 1 \\ - & - & - & - & - & - & - & - & - \\ 1 & 0 & | & 1 & 0 & | & 0 & 0 & 0 \\ 0 & 1 & | & 0 & 1 & | & 0 & 0 & 0 \end{bmatrix}, \ \boldsymbol{q} = \begin{bmatrix} 60 \\ 50 \\ 120 \\ - \\ 500 \\ 500 \end{bmatrix}$$

Graphical representation of the LP problem given in Equation 6.1 is provided as Figure 6.1 below. The contours of the objective function, as shown by the bold line in Figure 6.1, are parallel to the inequality constraint, $x + 2y \leq 120$. In this case, there are two alternate solutions at $x = 20$, $y = 50$ and at $x = 60$, $y = 30$. These two extreme vertex points and the infinite number of points connecting them have the same optimal objective function value of $\phi^* = 1200$.

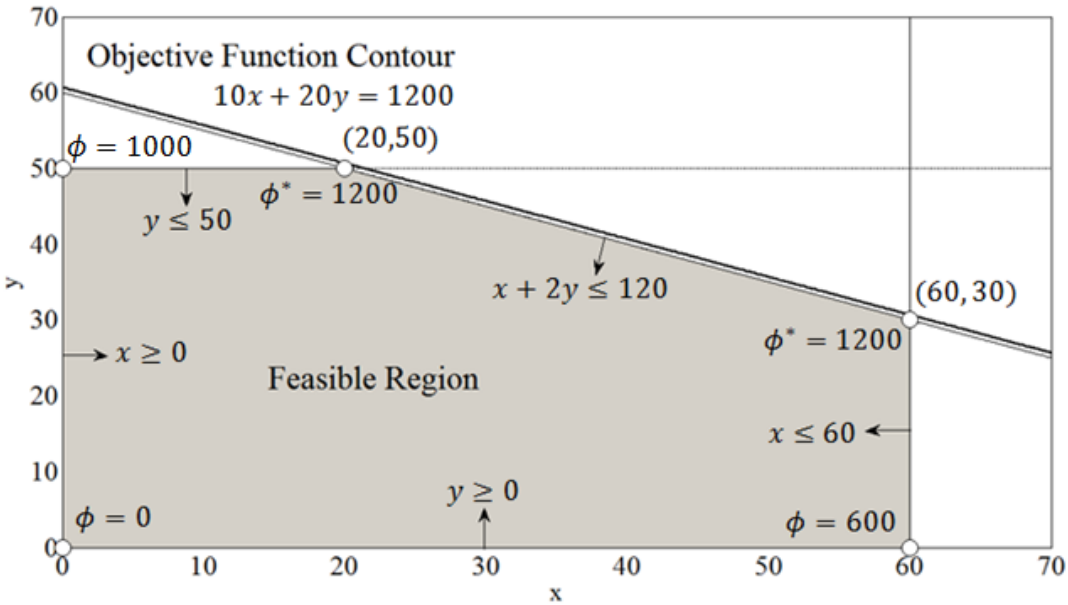

Figure 6.1. Graphical representation of the LP problem in Equation 6.1.

The problem was solved with GAMS and MATLAB, both versions gave the same results. In GAMS, XA was used as the LP and MILP solver. In MATLAB, `linprog` was used to obtain the 1st basic solution, and LPSOLVE is used as the LP and MILP solver for the RMILP section of the algorithm. In both versions the big-M parameter was U = 1000 and the termination criterion was ε = 1200.

For this maximization problem, the 4-best solutions (K-best solutions with K=4): (x^*, y^*) for the two alternate 1st best solutions and (x, y) for the remaining next 3-best solutions are as follows:

1st best solution:	$(60,\ 30)$,	$f^* = 1200$
1st best solution:	$(20,\ 50)$,	$f^* = 1200$
2nd next best solution:	$(0,\ 50)$,	$f = 1000$
3rd next best solution:	$(60,\ 0)$,	$f = 600$
4th next best solution:	$(0,\ 0)$,	$f = 0$

For the 5 set of solutions, the objective function values and values of decision variables for each solution are provided in Figures 6.2 and 6.3 below.

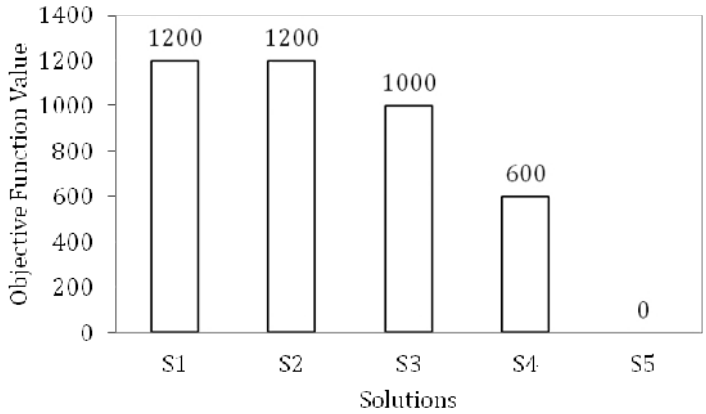

Figure 6.2. The two alternate and the next best objective function values of the LP problem in Equation 6.1.

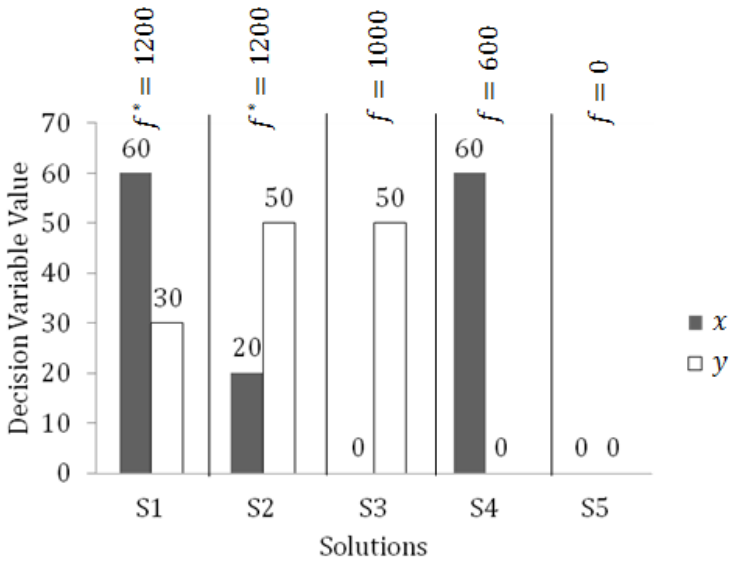

Figure 6.3. Decision variables for the two alternate and the next best solutions of the LP problem in Equation 6.1.

The GAMS and MATLAB codes for LP problem in Equation 6.1 for non-canonical form are provided in Appendix A1.

As can be seen by examining Figure 6.1, there cannot be any other solution since the solutions obtained by the GAMS and MATLAB versions of the algorithm exhaust all possible vertices in the 2-dimensional xy plane.

The solutions S1 and S2 are identical, these are alternatives of each other with the unique / common objective function value $f^* = 1200$. The third solution S3, under external knowledge that is not modeled into LP, may be a good next best selection since the objective function differs by 16.67 %. The fourth solution S4 may not be selected since it has an objective function value which is half of the optimal one. It can also be said that the second decision variable y plays an important role as it persists to be present with approximately the same value in solutions S1, S2, and S3. Whereas, the first decision variable x is not present (has a value of 0) in S3.

6.2. A Degenerate LP

A degenerate LP is the one in which at least one basic variable has a value of zero. A degenerate LP, which is also considered in Chapter 3, is provided below. The example also proves that Prof. Grossmann's algorithm (Lee *et al.*, 2000; Phalakornkule *et al.*, 2001) can handle degenerate cases as well. The problem statement is as follows (Phalakornkule *et al.*, 2001):

$$
\begin{aligned}
\min_{x_1,x_2,x_3} \quad & Z = x_3 \\
\text{s.t.} \quad & 2x_1 + x_2 + x_3 \leq 8 \\
& x_1 - 2x_2 + x_3 \leq 4 \\
& x_1 - 2x_2 + x_3 \geq -6 \\
& x_1 + 2x_2 + x_3 \geq -2 \\
& x_1 + x_2 + x_3 \leq 6 \\
& -5 \leq x_1 \leq 5 \\
& -5 \leq x_2 \leq 5 \\
& 0 \leq x_3
\end{aligned}
\tag{6.5}
$$

where, the optimal objective function value is $x_3^* = 0$. The problem has four extreme points on the hyperplane of $x_3 = 0$ (i.e., the plane of x_1 and x_2 once x_3 is fixed at zero, the optimal solution) and has degeneracy. In canonical form, the problem has 10 positive variables and 7 equality constraints. Hence the degrees of freedom is 3 and at each extreme point there are 7 basic variables.

The problem can be cast into the standard LP form given by Equation 6.2, where the vectors and matrices take the following values:

$$\boldsymbol{c}^T = [0\ 0\ 1],\ \ \boldsymbol{A}^2 = \begin{bmatrix} 2 & 1 & 1 \\ 1 & -2 & 1 \\ -1 & 2 & -1 \\ -1 & -2 & -1 \\ 1 & 1 & 1 \end{bmatrix},\ \ \boldsymbol{b}^2 = \begin{bmatrix} 8 \\ 4 \\ 6 \\ 2 \\ 6 \end{bmatrix},\ \ \boldsymbol{x}^L = \begin{bmatrix} -5 \\ -5 \\ 0 \end{bmatrix},\ \ \boldsymbol{x}^U = \begin{bmatrix} 5 \\ 5 \\ 500 \end{bmatrix}$$

where, 500 is an arbitrarily given upper bound for x_3., and $\boldsymbol{A}^1 = [\ ]$, $\boldsymbol{b}^1 = [\ ]$.

The canonical form of LP problem is given by Equation 6.4, where the vectors and matrices take the following values:

$$\boldsymbol{z} = \begin{bmatrix} \boldsymbol{s}^L_{3\times1} \\ \boldsymbol{s}^U_{3\times1} \\ \boldsymbol{s}_{5\times1} \end{bmatrix},\ \ \boldsymbol{\alpha} = \begin{bmatrix} \boldsymbol{c}_{3\times1} \\ \boldsymbol{0}_{3\times1} \\ \boldsymbol{0}_{5\times1} \end{bmatrix},\ \ \boldsymbol{B} = \begin{bmatrix} \boldsymbol{A}^2_{5\times3} & \boldsymbol{0}_{5\times3} & \boldsymbol{I}_{5\times5} \\ \boldsymbol{I}_{3\times3} & \boldsymbol{I}_{3\times3} & \boldsymbol{0}_{3\times5} \end{bmatrix},\ \ \boldsymbol{q} = \begin{bmatrix} (\boldsymbol{b}^2 - \boldsymbol{A}^2\boldsymbol{x}^L)_{5\times1} \\ (\boldsymbol{x}^U - \boldsymbol{x}^L)_{3\times1} \end{bmatrix}$$

where, $\boldsymbol{z}$ and $\boldsymbol{\alpha}$ are $(2n + m)$-dimensional column vectors, $\boldsymbol{B}$ is a $(k + m + n) \times (2n + m)$ coefficient matrix, and $\boldsymbol{q}$ is a $(k + m + n)$-dimensional column vector composed of right hand side values of the equality constraints. $\boldsymbol{I}$ is the identity matrix, n is the number of decision variables, k is the number of equality constraints, and m is the number of inequality constraints in Equation 6.4.

For this problem, the vectors and matrices are provided below where, the number of decision variables, n, is 3, the number of inequality constraints, m, is 5, and there are no equality constraints in the LP problem.

$$\boldsymbol{\alpha} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ - \\ 0 \\ 0 \\ 0 \\ - \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \ \boldsymbol{B} = \left[\begin{array}{ccc|ccc|ccccc} 2 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & -2 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ -1 & -2 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ - & - & - & - & - & - & - & - & - & - & - \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{array}\right], \ \boldsymbol{q} = \begin{bmatrix} 23 \\ -1 \\ 11 \\ -13 \\ 16 \\ - \\ 5 \\ 5 \\ 500 \end{bmatrix}$$

The four alternate optimal solutions found by the algorithm are as follows (for all of them the objective function value is zero):

$$(x_1^*,\ x_2^*,\ x_3^*) = (2,4,0),\ (4,0,0),\ (1,\ -1.5,0),\ (-4,1,0)$$

Due to degeneracy, the point $(2,4,0)$ has a zero-valued basic variable but the algorithm finds all the extreme points and stops when the objective function value is greater than zero. Though the number of positive basic variables changes at each iteration, no alternate extreme optimum is excluded and each optimal extreme point is visited once. The four extreme points on the hyperplane of $x_3 = 0$ are shown on the following plot:

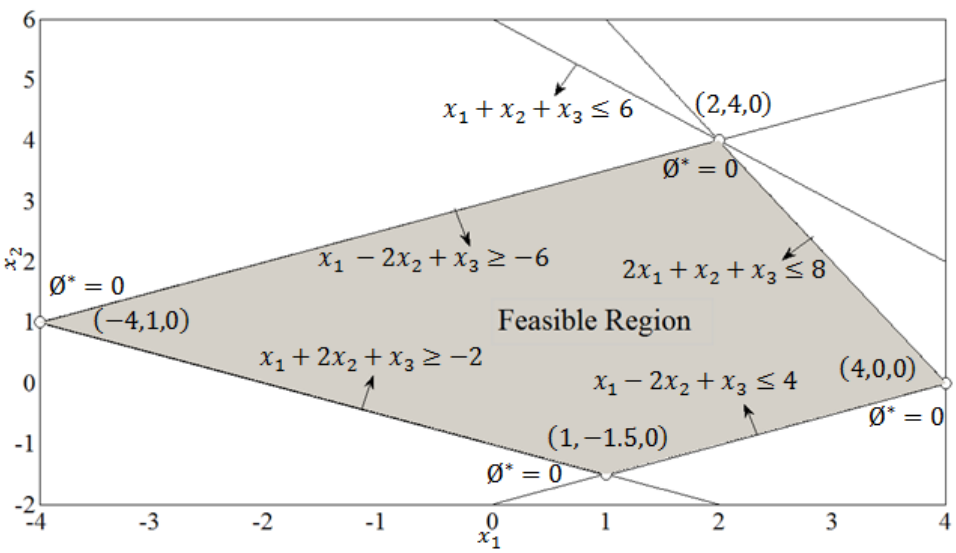

Figure 6.4. Hyperplane of $x_3 = 0$ in Equation 6.5.

The above alternate solutions can be obtained with $\varepsilon = 0$ (since $x_3^* = 0$). In Figure 6.4, arrows show the directions in which the constraints are valid.

The problem was solved with GAMS and MATLAB, both versions gave the same results. In GAMS, XA was used as the LP and MILP solver. In MATLAB, `linprog` was used to obtain the 1st basic solution, and LPSOLVE is used as the LP and MILP solver for the RMILP section of the algorithm. In both versions the big-M parameter was U = 1000 and the termination criterion was ε = 11.

For this minimization problem, the 6-best solutions (K-best solutions with K=6): (x_1^*, x_2^*, x_3^*) for the four alternate 1st best solutions and (x_1, x_2, x_3) for the remaining next 5-best solutions obtained are as follows:

1st best solution:	$(2,\ 4,\ 0)$,	$f^* = 0$
1st best solution:	$(4,\ 0,\ 0)$,	$f^* = 0$
1st best solution:	$(-4,\ 1,\ 0)$,	$f^* = 0$
1st best solution:	$(1,\ -1.5,\ 0)$,	$f^* = 0$
2nd next best solution:	$(-5,\ 1,\ 1)$,	$f = 1$
3rd next best solution:	$(2,\ 0.67,\ 3.33)$,	$f = 3.33$
4th next best solution:	$(-5,\ -1.5,\ 6)$,	$f = 6$
5th next best solution:	$(-5,\ 4,\ 7)$,	$f = 7$
6th next best solution:	$(-5,\ 0.67,\ 10.33)$,	$f = 10.33$

For the 9 set of solutions, the objective function values and values of decision variables for each solution are provided in Figures 6.5 and 6.6 below.

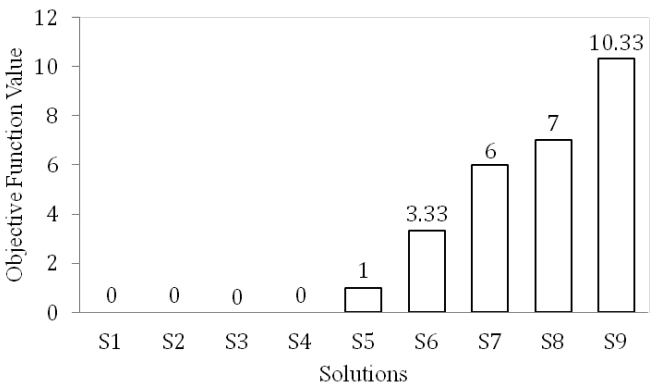

Figure 6.5. The four alternate and the next best objective function values of the LP problem in Equation 6.5.

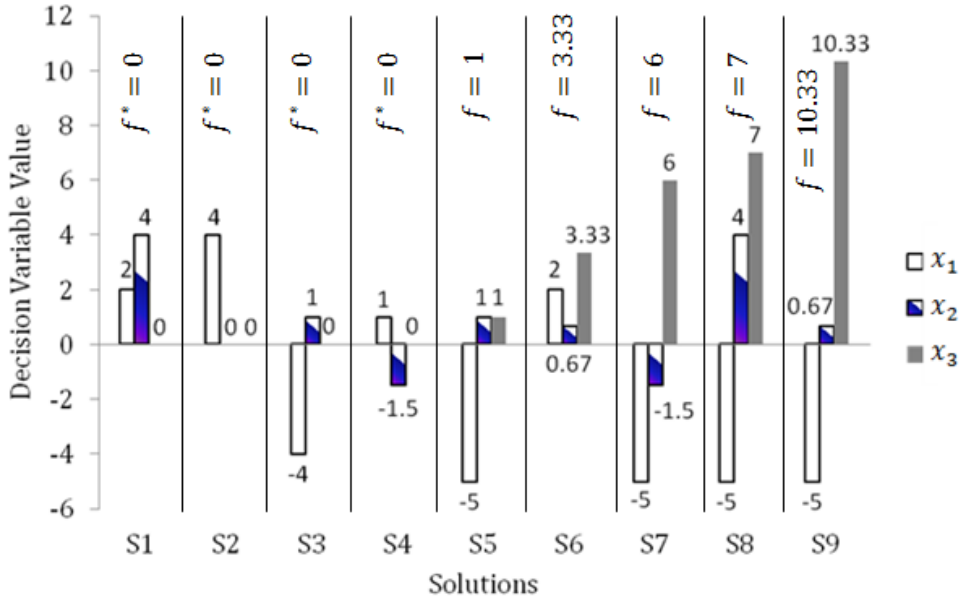

Figure 6.6. Decision variable values in the 9 solutions of the LP problem in Equation 6.5.

The GAMS and MATLAB codes for LP problem in Equation 6.5 for non-canonical form are provided in Appendix A2.

No any other solution is possible since $K = 6$ exhausts all possible vertices. The above solutions were obtained using LPSOLVE for MILP in MATLAB. The same solutions were obtained using GLPK for MILP in MATLAB as well. In GAMS, when XA is used as solver for both LP and MILP, 3rd, 4th, 5th, and 6th best solutions are replicated two times. When XPRESS is used for LP and MILP in GAMS, the 1st best solution of $(x_1^*, x_2^*, x_3^*) = (2, 4, 0)$ is replicated six times, the 1st best solution of $(x_1^*, x_2^*, x_3^*) = (1, -1.5, 0)$ is replicated four times, the 2nd best solution is replicated three times, the 4th, 5th, and 6th best solutions are replicated two times. However, the unique set of 6 best solutions is obtained using any of the LP/MILP solvers in MATLAB and GAMS and replications depend on the solver used for LP and MILP in GAMS and for MILP in MATLAB.

Here, the user can select one of the four alternative optimal solutions, namely S1, S2, S3, and S4, without having any deterioration in the objective function, which is 0. The next best solution S5 has an objective function value of 1 and may be selected under external knowledge that is not modeled into LP since the objective function differs only by an

amount of 1. The 3rd best solution, S6, may not be selected since it has an objective function value larger than three times that of the 2nd best solution.

6.3. Refinery Blending Problem

In this section a Chemical Engineering related Refinery Blending problem is provided. In this problem, there is a refinery that can process two different crude oils to produce and sell four products namely, gasoline, kerosene, fuel oil, and residual. The crude oil costs and product prices are provided in Figure 6.7 below.

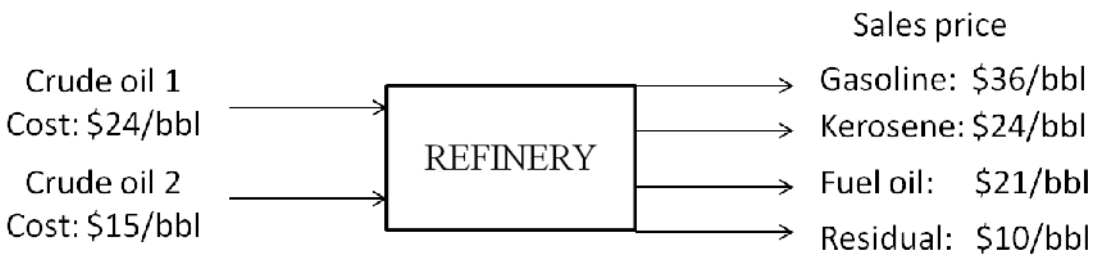

Figure 6.7. Crude oil costs and product prices for the Refinery Blending problem.

where, bbl stands for barrel. % yields of the two crude oils in terms of volume and maximum allowable production amounts for the products are provided in Table 6.2 below.

Table 6.2. % yield and maximum allowable production data for the Refinery Blending problem.

	Volume % yield		Maximum allowable
	Crude 1	Crude 2	production (bbl/day)
Gasoline	80	44	24000
Kerosene	8.1	10.8	2500
Fuel oil	10	36	6000
Residual	1.9	9.2	-
Processing cost ($/bbl)	0.934	1.112	

where, residual has no production limit.

The aim is to maximize profit where the decision variables are defined as: x_1: amount of crude oil 1 processed in bbl/day, x_2: amount of crude oil 2 processed in bbl/day,

x_3: amount of gasoline produced in bbl/day, x_4: amount of kerosene produced in bbl/day, x_5: amount of fuel oil produced in bbl/day, x_6: amount of residual produced in bbl/day.

Profit is defined in the following equation.

$$\text{Profit} = \text{Product price} - \text{Raw material cost} - \text{Processing cost} \tag{6.6}$$

where, product price, raw material cost, and processing cost for this problem are provided below.

$$\text{Product price} = 36x_3 + 24x_4 + 21x_5 + 10x_6 \tag{6.7}$$

$$\text{Raw material cost} = 24x_1 + 15x_2 \tag{6.8}$$

$$\text{Processing cost} = 0.934x_1 + 1.112x_2 \tag{6.9}$$

Profit, the objective function to be maximized, is obtained by substituting these in Equation 6.6 as given below.

$$\text{Profit}\ (\tfrac{\$}{\text{day}}) = 36x_3 + 24x_4 + 21x_5 + 10x_6 - 24.934x_1 - 16.112x_2 \tag{6.10}$$

The problem statement is as follows:

$$\begin{aligned}
\max_{x_1, x_2, x_3, x_4, x_5, x_6} \quad & Z = 36x_3 + 24x_4 + 21x_5 + 10x_6 - 24.934x_1 - 16.112x_2 \\
\text{s.t.} \quad & 0.80x_1 + 0.44x_2 = x_3 \\
& 0.081x_1 + 0.108x_2 = x_4 \\
& 0.10x_1 + 0.36x_2 = x_5 \\
& 0.019 + 0.092x_2 = x_6 \\
& x_3 \leq 24000 \\
& x_4 \leq 2500 \\
& x_5 \leq 6000 \\
& x_i \geq 0,\ i = 1, \dots, 6
\end{aligned} \tag{6.11}$$

The problem can be cast into the standard LP form given by Equation 6.2, where, when converted into minimization form, the vectors and matrices take the following values:

$$\boldsymbol{c} = \begin{bmatrix} 24.934 \\ 16.112 \\ -36 \\ -24 \\ -21 \\ -10 \end{bmatrix}, \quad \boldsymbol{A^1} = \begin{bmatrix} 0.80 & 0.44 & -1 & 0 & 0 & 0 \\ 0.081 & 0.108 & 0 & -1 & 0 & 0 \\ 0.10 & 0.36 & 0 & 0 & -1 & 0 \\ 0.019 & 0.092 & 0 & 0 & 0 & -1 \end{bmatrix}, \quad \boldsymbol{b^1} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

$$\boldsymbol{A}^2 = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}, \quad \boldsymbol{b}^2 = \begin{bmatrix} 24000 \\ 2500 \\ 6000 \end{bmatrix}, \quad \boldsymbol{x}^L = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \quad \boldsymbol{x}^U = \begin{bmatrix} 50000 \\ 50000 \\ 50000 \\ 50000 \\ 50000 \\ 50000 \end{bmatrix}$$

where, 50000 is an arbitrarily given upper bound for x_i where $i = 1, 2, \ldots, 6$. Objective function coefficients are multiplied by -1 to transform maximization into minimization.

The canonical form of LP problem is given by Equation 6.4, where the vectors and matrices take the following values:

$$\boldsymbol{z} = \begin{bmatrix} \boldsymbol{s}^L_{6\times1} \\ \boldsymbol{s}^U_{6\times1} \\ \boldsymbol{s}_{3\times1} \end{bmatrix}, \quad \boldsymbol{\alpha} = \begin{bmatrix} \boldsymbol{c}_{6\times1} \\ \boldsymbol{0}_{6\times1} \\ \boldsymbol{0}_{3\times1} \end{bmatrix}, \quad \boldsymbol{B} = \begin{bmatrix} \boldsymbol{A}^{\boldsymbol{1}}_{4\times6} & \boldsymbol{0}_{4\times6} & \boldsymbol{0}_{4\times3} \\ \boldsymbol{A}^{\boldsymbol{2}}_{3\times6} & \boldsymbol{0}_{3\times6} & \boldsymbol{I}_{3\times3} \\ \boldsymbol{I}_{6\times6} & \boldsymbol{I}_{6\times6} & \boldsymbol{0}_{6\times3} \end{bmatrix}, \quad \boldsymbol{q} = \begin{bmatrix} (\boldsymbol{b}^1 - \boldsymbol{A}^1\boldsymbol{x}^L)_{4\times1} \\ (\boldsymbol{b}^2 - \boldsymbol{A}^2\boldsymbol{x}^L)_{3\times1} \\ (\boldsymbol{x}^U - \boldsymbol{x}^L)_{6\times1} \end{bmatrix}$$

where, $\boldsymbol{z}$ and $\boldsymbol{\alpha}$ are $(2n + m)$-dimensional column vectors, $\boldsymbol{B}$ is a $(k + m + n) \times (2n + m)$ coefficient matrix, and $\boldsymbol{q}$ is a $(k + m + n)$-dimensional column vector composed of right hand side values of the equality constraints. $\boldsymbol{I}$ is the identity matrix, n is the number of decision variables, k is the number of equality constraints, and m is the number of inequality constraints in Equation 6.4. For this problem, the vectors and matrices take the following values:

$$\boldsymbol{B} = \left[\begin{array}{cccccc|cccccc|ccc}
0.80 & 0.44 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0.081 & 0.108 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0.10 & 0.36 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0.019 & 0.092 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\hline
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
\hline
1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0
\end{array}\right]$$

$$\boldsymbol{\alpha} = \left[\begin{array}{c}
24.934 \\ 16.112 \\ -36 \\ -24 \\ -21 \\ -10 \\ \hline
0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \hline
0 \\ 0 \\ 0
\end{array}\right], \quad \boldsymbol{q} = \left[\begin{array}{c}
0 \\ 0 \\ 0 \\ 0 \\ \hline
24000 \\ 2500 \\ 6000 \\ \hline
50000 \\ 50000 \\ 50000 \\ 50000 \\ 50000 \\ 50000
\end{array}\right]$$

where, the number of decision variables, n, is 6, number of equality constraints, k, is 4, and the number of inequality constraints, m, is 3.

The problem was solved with GAMS and MATLAB, both versions gave the same results. In GAMS, XA was used as the LP and MILP solver. In MATLAB, `linprog` was used to obtain the 1st basic solution, and LPSOLVE is used as the LP and MILP solver for the RMILP section of the algorithm. In both versions the big-M parameter was U = 1000000 and the termination criterion was ε = 250001.

For this maximization problem, the 4-best solutions (K-best solutions with K=4): $(x_1^*, x_2^*, x_3^*, x_4^*, x_5^*, x_6^*)$ for the two alternate 1st best solutions and $(x_1, x_2, x_3, x_4, x_5, x_6)$ for the remaining next 3-best solutions obtained are provided in the following table:

Table 6.3. The 4-best solutions of the LP in Equation 6.11.

	x_1	x_2	x_3	x_4	x_5	x_6	f
1st best solution	29393.22	1103.23	24000	2500	3336.49	659.97	250000
1st best solution	13725.49	12854.03	16636.17	2500	6000	1443.36	250000
2nd best solution	30000	0	24000	2430	3000	570	243000
3rd best solution	0	16666.67	7333.33	1800	6000	1533.33	180000
4th best solution	0	0	0	0	0	0	0

For the 5 set of solutions, the objective function values and values of decision variables for each solution are provided in Figures 6.8 and 6.9 below.

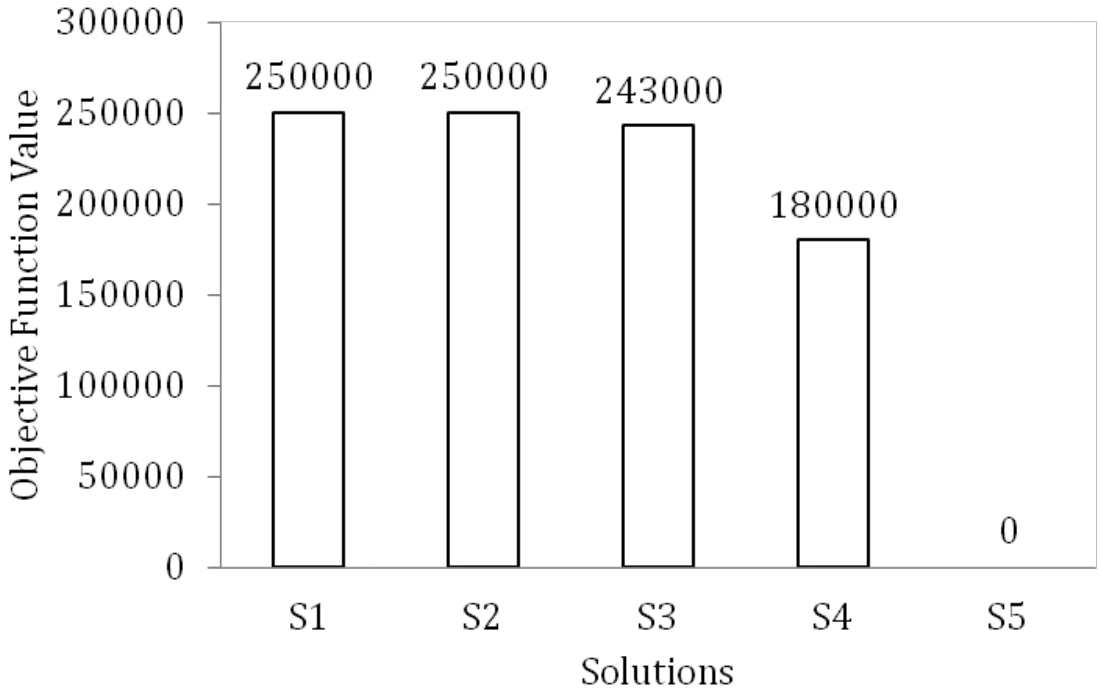

Figure 6.8. The two alternate and the next best objective function values of the LP problem in Equation 6.11.

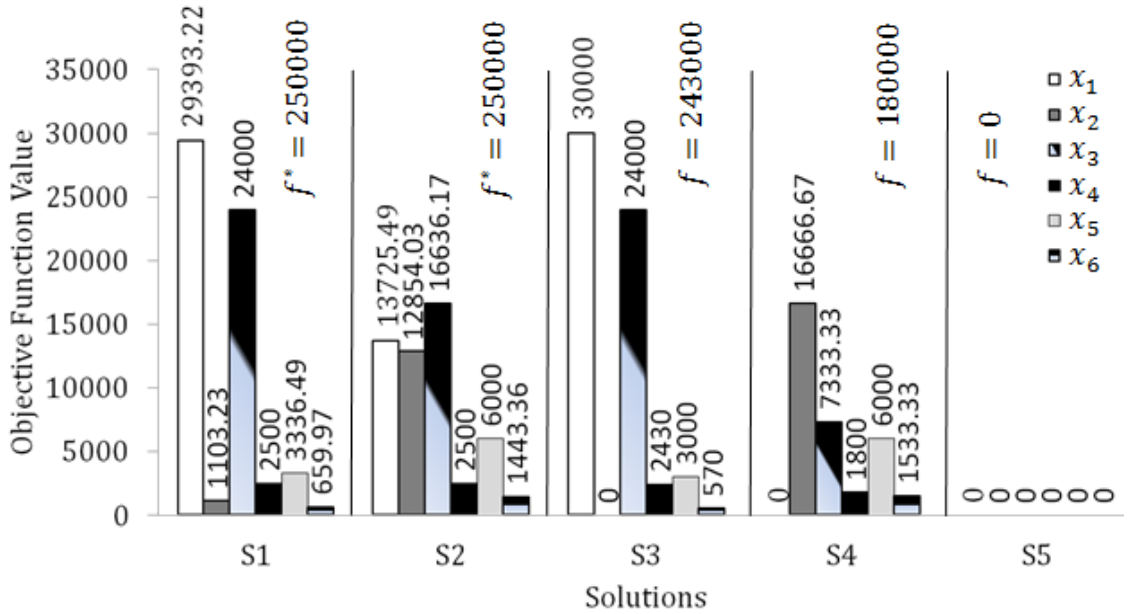

Figure 6.9. Decision variable values in the 5 solutions of the LP problem in Equation 6.11.

The GAMS and MATLAB codes for LP problem in Equation 6.11 for non-canonical form are provided in Appendix A3. No any other solution is possible since $K = 4$ exhausts all possible vertices.

Here, the user can select one of the two alternative optimal solutions namely, S1 and S2 without having any deterioration in the objective function, which is 250000 \$/day. The next best solution S3 has an objective function value of 243000 \$/day and may be selected under external knowledge that is not modeled into LP since the objective function differs only by 2.8 %. The 3rd best solution, S4, may not be selected since it has an objective function value that differs by 28 % when compared to that of the 1st best solution.

In order to prove that the solutions obtained via the RMILP algorithm are correct, the dimensionality of the LP in Equation 6.11 can be reduced by eliminating variables x_3 , x_4, x_5, and x_6 via equality constraints, leaving two independent decision variables namely, x_1 and x_2., to obtain the graphical solution. The problem then becomes:

$$\max_{x_1, x_2} \quad Z = 8.1x_1 + 10.8x_2$$

$$\text{s.t.} \quad 0.8x_1 + 0.44x_2 \leq 24000$$

$$0.081x_1 + 0.108x_2 \leq 2500$$

$$0.1x_1 + 0.36x_2 \leq 6000$$
$$x_1 \geq 0$$
$$x_2 \geq 0 \qquad (6.12)$$

Graphical representation of the LP in Equation 6.12 is provided below:

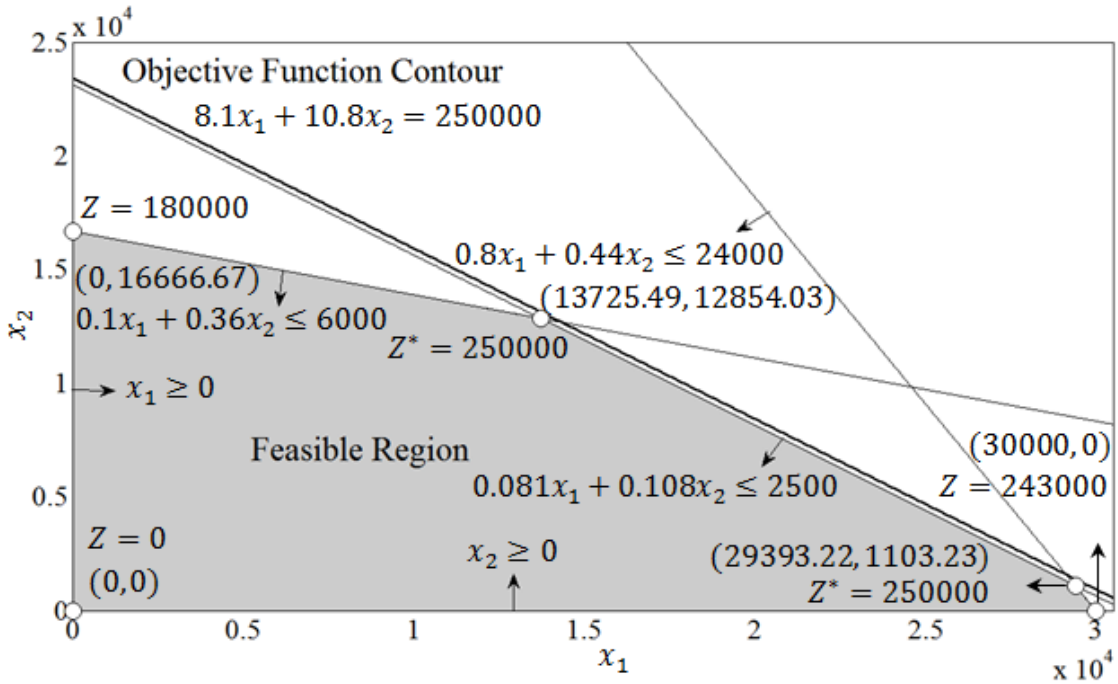

Figure 6.9. Graphical representation of the LP in Equation 6.12.

where, the arrows show the direction in which constraints are valid and bold line represents the objective function contour. Figure 6.9 represents all possible solutions of the LP in Equation 6.12, which are identical to those obtained with RMILP algorithm using GAMS or MATLAB.

6.4. Thermal Cracker Problem

In this section, a Chemical Engineeing related Thermal Cracker problem is provided (Edgar *et al.*, 2001). In this problem, operation of a thermal cracker, sketched below, is optimized.

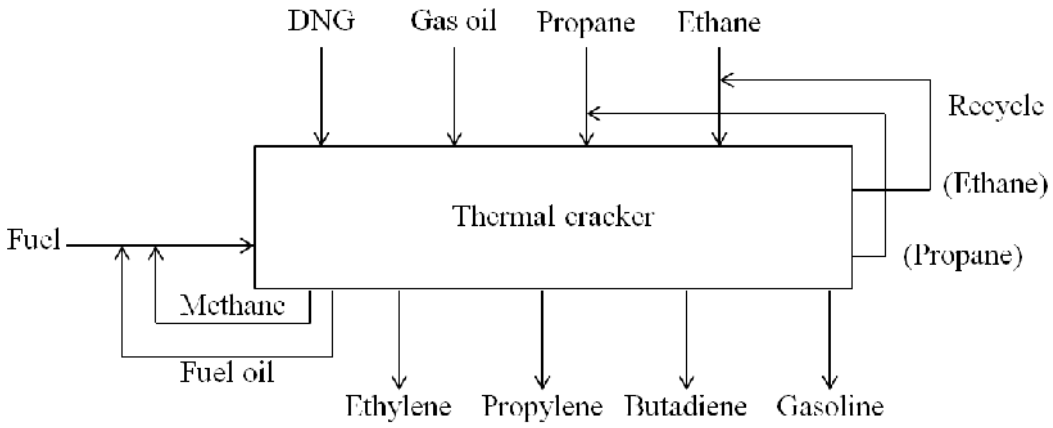

Figure 6.10. Flow diagram of thermal cracker.

Various feeds and corresponding product distribution for a thermal cracker that produces olefins are shown in Table 6.4. The possible feeds include ethane, propane, debutanized natural gasoline (DNG), and gas oil, some of which may be fed simultaneously. Based on plant data, eight products are produced in varying proportions according to Table 6.4. The capacity to run gas feeds through the cracker is 200000 lb/stream hour (total flow based on an average mixture) where, ethane uses the equivalent of 1.1 lb of capacity per pound of ethane; propane 0.9 lb; gas oil 0.9 lb/lb; and DNG 1.0.

Table 6.4. Yield structure in terms of weight fractions.

	Feed			
Product	Ethane	Propane	Gas oil	DNG
Methane	0.07	0.25	0.10	0.15
Ethane	0.40	0.06	0.04	0.05
Ethylene	0.50	0.35	0.20	0.25
Propane	-	0.10	0.01	0.01
Propylene	0.01	0.15	0.15	0.18
Butadiene	0.01	0.02	0.04	0.05
Gasoline	0.01	0.07	0.25	0.30
Fuel oil	-	-	0.21	0.01

Downstream processing limits are 50000 lb/stream hour for ethylene and 20000 lb/stream hour for propylene. The fuel requirements to run the cracking system for each feedstock type are provided in Table 6.5.

Table 6.5. Fuel requirements to run the cracking system.

Feedstock type	Feed requirement (Btu/lb)
Ethane	8364
Propane	5016
Gas oil	3900
DNG	4553

Methane and fuel oil produced by the cracker are recycled as fuel. All the ethane and propane produced are recycled as feed. Heating values are provided in Table 6.6.

Table 6.6. Heating values of the recycled products.

Recycled feed	Heat produced (Btu/lb)
Natural gas	21520
Methane	21520
Fuel oil	18000

Because of heat losses and the energy requirements for pyrolysis, the fixed fuel requirement is 20.0×10^6 Btu/stream hour. The price structure on the feeds and products and fuel costs are provided in Table 6.7.

Table 6.7. Price structure on the feeds and products and fuel costs.

Feeds	Price (¢/lb)
Ethane	6.55
Propane	9.73
Gas oil	12.50
DNG	10.14
Products	Price (¢/lb)
Methane	5.38 (fuel value)
Ethylene	17.75
Propylene	13.79
Butadiene	26.64
Gasoline	9.93
Fuel oil	4.50 (fuel value)

where, an energy (fuel) cost of $\$2.50/10^6$ Btu is assumed.

There are 7 decision variables, the flow rates to and from the furnace (in lb/h), which are defined as: x_1: fresh ethane feed, x_2: fresh propane feed, x_3: gas oil feed, x_4: DNG feed, x_5: ethane recycle, x_6: propane recycle, x_7: fuel added.

Two assumptions are used in formulating the objective function and constraints. The first one is that there is a fixed fuel requirement (methane) of 20×10^6 Btu/h to compensate for the heat loss. The second one is that all propane and ethane are recycled with the feed, and all methane and fuel oil are recycled as fuel. Additionally, a basis of 1 hour is used, and all costs are calculated in cents per hour.

Profit is defined as:

$$\text{Profit} = \text{Product value} - \text{Feed cost} - \text{Enegy cost} \tag{6.13}$$

where, the value for each product (in cents per pound) is provided below.

Ethylene: $$17.75(0.50x_1 + 0.50x_5 + 0.35x_2 + 0.35x_6 + 0.20x_3 + 0.25x_4) \tag{6.14}$$

Propylene: $$13.79(0.01x_1 + 0.01x_5 + 0.15x_2 + 0.15x_6 + 0.15x_3 + 0.18x_4) \tag{6.15}$$

Butadiene: $$26.64(0.01x_1 + 0.01x_5 + 0.02x_2 + 0.02x_6 + 0.04x_3 + 0.05x_4) \tag{6.16}$$

Gasoline: $$9.93(0.01x_1 + 0.01x_5 + 0.07x_2 + 0.07x_6 + 0.25x_3 + 0.30x_4) \tag{6.17}$$

where, the total product sales, sum of the above expressions, is:

$$9.38x_1 + 9.38x_5 + 9.51x_2 + 9.51x_6 + 9.17x_3 + 11.23x_4 \tag{6.18}$$

Feed cost (¢/h) is provided as:

$$6.55x_1 + 9.73x_2 + 12.50x_3 + 10.14x_4 \tag{6.19}$$

The fixed heat loss of 20×10^6 Btu/h can be expressed in terms of methane cost (5.38 ¢/h) using a heating value of 21520 Btu/lb for methane. The fixed heat loss

represents a constant cost that is independent of the decision variables, hence it can be ignored in the formulation of the LP problem, but in evaluating the final costs it must be taken into account. The value for x_7 depends on the amount of fuel oil and methane produced in the cracker (x_7 provides for any deficit in products recycled as fuel). The energy cost then becomes:

$$\text{Energy cost} = \text{Fuel cost} + \text{Fixed energy cost} \tag{6.20}$$

where, fuel cost and fixed energy cost are provided below.

$$\text{Fuel cost (¢/h)} = \frac{2.50\times100\times(8364\times(x_1+x_5)+5016\times(x_2+x_6)+3900x_3+4553x_4)}{10^6} \tag{6.21}$$

$$\text{Fixed energy cost (¢/h)} = \frac{20\times10^6\times5.38}{21520} \tag{6.22}$$

where, 100 is the factor to convert dollars into cents in Equation 6.21. The objective function is provided below.

$$f = 0.7376 - 1.4751x_2 - 4.3084x_3 - 0.0476x_4 + 7.2876x_5 + 8.2549x_6 - 5000 \tag{6.23}$$

where, 5000 represents the fixed energy cost, which may be eliminated in the formulation of the LP problem.

In this problem, there are six constraints, three of which are inequality constraints. There is a cracker capacity of 200000 lb/h, which is expressed in the inequality constraint provided below.

$$1.1x_1 + 0.9x_2 + 0.9x_3 + 1.0x_4 + 1.1x_5 + 0.9x_6 \leq 200000 \tag{6.24}$$

Ethylene processing limitation of 50000 lb/h is expressed in the inequality constraint given below.

$$0.5x_1 + 0.35x_2 + 0.2x_3 + 0.25x_4 + 0.5x_5 + 0.35x_6 \leq 50000 \tag{6.25}$$

Propylene processing limitation of 20000 lb/h is expressed in the following inequality constraint:

$$0.01x_1 + 0.15x_2 + 0.15x_3 + 0.18x_4 + 0.01x_5 + 0.15x_6 \leq 20000 \quad (6.26)$$

Ethane recycle is expressed in the following equality constraint:

$$0.4x_1 + 0.06x_2 + 0.04x_3 + 0.05x_4 - 0.6x_5 + 0.06x_6 = 0 \quad (6.27)$$

Propane recycle is expressed in the following equality constraint:

$$0.1x_2 + 0.01x_3 + 0.01x_4 - 0.9x_6 = 0 \quad (6.28)$$

Heat constraint is provided below as an equality.

$$\text{Required fuel} + \text{Fixed fuel requirement} = \text{THV} \quad (6.29)$$

where, THV stands for the total fuel heating value and all of the terms in Equation 6.29 have the units of Btu/h. The required fuel for cracking, fixed fuel requirement, and total fuel heating value are provided below.

$$\text{Required fuel} = 8364(x_1 + x_5) + 5016(x_2 + x_6) + 3900x_3 + 4553x_4 \quad (6.30)$$

$$\text{Fixed fuel requirement} = 20 \times 10^6 \quad (6.31)$$

$$\text{THV} = 21520x_7 + 21520(0.07x_1 + 0.25x_2 + 0.1x_3 + 0.15x_4 + 0.07x_5 + 0.25x_6) + 18000(0.21x_3 + 0.01x_4) \quad (6.32)$$

The heat constraint then becomes:

$$-6857.6x_1 + 364x_2 + 2032x_3 - 1145x_4 - 6857.6x_5 + 364x_6 + 21520x_7 = 20000000 \quad (6.33)$$

The LP problem can thus be given as follows:

$$\max_{x_1,x_2,x_3,x_4,x_5,x_6,x_7} \quad Z = 0.7376x_1 - 1.4751x_2 - 4.3084x_3 - 0.0476x_4 + 7.2876x_5 + 8.2549x_6 - 5000$$

$$\begin{aligned}
\text{s.t.} \quad & 1.1x_1 + 0.9x_2 + 0.9x_3 + 1.0x_4 + 1.1x_5 + 0.9x_6 \leq 200000 \\
& 0.5x_1 + 0.35x_2 + 0.2x_3 + 0.25x_4 + 0.5x_5 + 0.35x_6 \leq 50000 \\
& 0.01x_1 + 0.15x_2 + 0.15x_3 + 0.18x_4 + 0.01x_5 + 0.15x_6 \leq 20000 \\
& 0.4x_1 + 0.06x_2 + 0.04x_3 + 0.05x_4 - 0.6x_5 + 0.06x_6 = 0 \\
& 0.1x_2 + 0.01x_3 + 0.01x_4 - 0.9x_6 = 0 \\
& -6857.6x_1 + 364x_2 + 2032x_3 - 1145x_4 - 6857.6x_5 + 364x_6 + 21520x_7 = 20 \times 10^6 \\
& x_i \geq 0, \; i = 1, 2, \dots, 7
\end{aligned} \tag{6.34}$$

The problem can be cast into the standard LP form given by Equation 6.2, where, when converted into minimization form, the vectors and matrices take the following values:

$$\boldsymbol{c} = \begin{bmatrix} -0.7376 \\ 1.4751 \\ 4.3084 \\ 0.0476 \\ -7.2876 \\ -8.2549 \\ 0 \end{bmatrix}, \quad \boldsymbol{A^1} = \begin{bmatrix} 0.4 & 0.06 & 0.04 & 0.05 & -0.6 & 0.06 & 0 \\ 0 & 0.1 & 0.01 & 0.01 & 0 & -0.9 & 0 \\ -6857.6 & 364 & 2032 & -1145 & -6857.6 & 364 & 21520 \end{bmatrix}$$

$$\boldsymbol{b^1} = \begin{bmatrix} 0 \\ 0 \\ 20 \times 10^6 \end{bmatrix}, \quad \boldsymbol{A}^2 = \begin{bmatrix} 1.1 & 0.9 & 0.9 & 1 & 1.1 & 0.9 & 0 \\ 0.5 & 0.35 & 0.2 & 0.25 & 0.5 & 0.35 & 0 \\ 0.01 & 0.15 & 0.15 & 0.18 & 0.01 & 0.15 & 0 \end{bmatrix}$$

$$\boldsymbol{b}^2 = \begin{bmatrix} 200000 \\ 50000 \\ 20000 \end{bmatrix}, \quad \boldsymbol{x}^L = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \quad \boldsymbol{x}^U = \begin{bmatrix} 100000 \\ 100000 \\ 100000 \\ 100000 \\ 100000 \\ 100000 \\ 100000 \end{bmatrix}$$

where, 100000 is an arbitrarily given upper bound for x_i where $i = 1, 2, \dots, 7$. Objective function coefficients are multiplied by -1 to transform maximization into minimization.

The canonical form of LP problem is given by Equation 6.4, where the vectors and matrices take the following values:

$$\boldsymbol{z} = \begin{bmatrix} \boldsymbol{s}^L_{7\times1} \\ \boldsymbol{s}^U_{7\times1} \\ \boldsymbol{s}_{3\times1} \end{bmatrix},\ \boldsymbol{\alpha} = \begin{bmatrix} \boldsymbol{c}_{7\times1} \\ \boldsymbol{0}_{7\times1} \\ \boldsymbol{0}_{3\times1} \end{bmatrix},\ \boldsymbol{B} = \begin{bmatrix} \boldsymbol{A}^1_{3\times7} & \boldsymbol{0}_{3\times7} & \boldsymbol{0}_{3\times3} \\ \boldsymbol{A}^2_{3\times7} & \boldsymbol{0}_{3\times7} & \boldsymbol{I}_{3\times3} \\ \boldsymbol{I}_{7\times7} & \boldsymbol{I}_{7\times7} & \boldsymbol{0}_{7\times3} \end{bmatrix},\ \boldsymbol{q} = \begin{bmatrix} (\boldsymbol{b}^1 - \boldsymbol{A}^1\boldsymbol{x}^L)_{3\times1} \\ (\boldsymbol{b}^2 - \boldsymbol{A}^2\boldsymbol{x}^L)_{3\times1} \\ (\boldsymbol{x}^U - \boldsymbol{x}^L)_{7\times1} \end{bmatrix}$$

where, $\boldsymbol{z}$ and $\boldsymbol{\alpha}$ are $(2n + m)$-dimensional column vectors, $\boldsymbol{B}$ is a $(k + m + n) \times (2n + m)$ coefficient matrix, and $\boldsymbol{q}$ is a $(k + m + n)$-dimensional column vector composed of right hand side values of the equality constraints. $\boldsymbol{I}$ is the identity matrix, n is the number of decision variables, k is the number of equality constraints, and m is the number of inequality constraints in Equation 6.4. For this problem, the vectors and matrices take the following values:

$$\boldsymbol{B} = \left[\begin{array}{ccccccc|ccccccc|ccc}
0.4 & 0.06 & 0.04 & 0.05 & -0.6 & 0.06 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0.1 & 0.01 & 0.01 & 0 & -0.9 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-6857.6 & 364 & 2032 & -1145 & -6857.6 & 364 & 21520 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\hline
1.1 & 0.9 & 0.9 & 1 & 1.1 & 0.9 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0.5 & 0.35 & 0.2 & 0.25 & 0.5 & 0.35 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0.01 & 0.15 & 0.15 & 0.18 & 0.01 & 0.15 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
\hline
1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0
\end{array}\right]$$

$$\boldsymbol{\alpha} = \begin{bmatrix} -0.7376 \\ 1.4751 \\ 4.3084 \\ 0.0476 \\ -7.2876 \\ -8.2549 \\ 0 \\ \hline 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \hline 0 \\ 0 \\ 0 \end{bmatrix},\ \boldsymbol{q} = \begin{bmatrix} 0 \\ 0 \\ 20 \times 10^6 \\ \hline 200000 \\ 50000 \\ 20000 \\ \hline 100000 \\ 100000 \\ 100000 \\ 100000 \\ 100000 \\ 100000 \\ 100000 \end{bmatrix}$$

where, the number of decision variables, n, is 7, number of equality constraints, k, is 3, and the number of inequality constraints, m, is 3.

The problem was solved with GAMS and MATLAB, both versions gave the same results. In GAMS, XA was used as the LP and MILP solver. In MATLAB, `linprog` was used to obtain the 1st basic solution, and LPSOLVE is used as the LP and MILP solver for the RMILP section of the algorithm. In both versions the big-M parameter was U = 1000000 and the termination criterion was ε = 335760.

For this maximization problem, the 6-best solutions (K-best solutions with K=6): $(x_1^*, x_2^*, x_3^*, x_4^*, x_5^*, x_6^*)$ for the 1st best solution and $(x_1, x_2, x_3, x_4, x_5, x_6)$ for the remaining next 5-best solutions obtained are provided in the following table:

Table 6.8. The 6-best solutions of the LP in Equation 6.34.

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	f
1st best solution	60000	0	0	0	40000	0	32795.54	330760
2nd best solution	21768.07	0	0	107594.54	23597.81	1195.49	21090.23	187774.72
3rd best solution	0	0	0	109582	9253.62	1217.58	9688.02	67271.59
4th best solution	0	101902.80	0	15905.74	12665.68	11499.26	3893.58	31153.79
5th best solution	0	112500	0	0	12500	12500	2798.33	23332.50
6th best solution	0	0	0	0	0	0	929.37	−5000

For the 6 set of solutions, the objective function values and values of decision variables for each solution are provided in Figures 6.11 and 6.12 below.

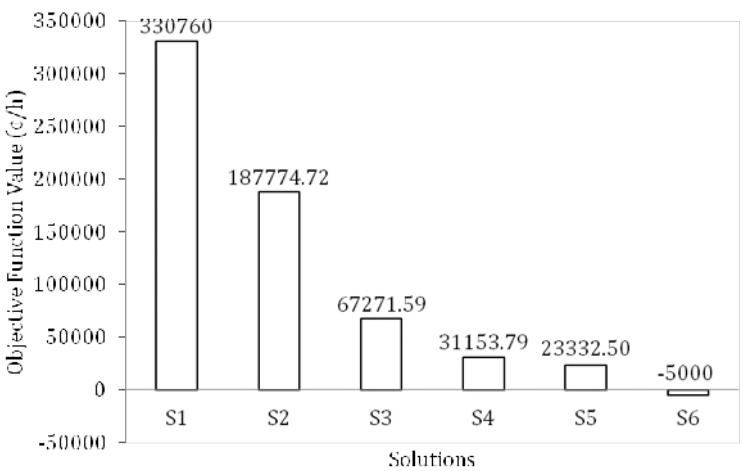

Figure 6.11. The 6 best objective function values of the LP problem in Equation 6.34.

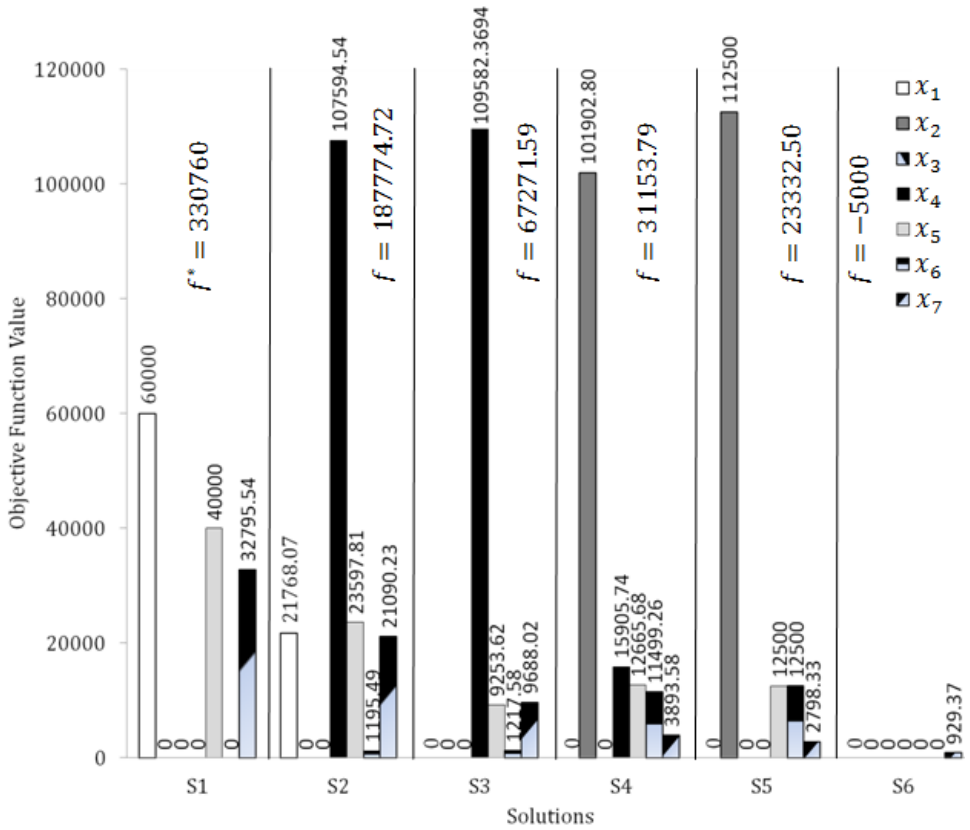

Figure 6.12. Decision variable values in the 6 solutions of the LP problem in Equation 6.34.

The GAMS and MATLAB codes for LP problem in Equation 6.34 for non-canonical form are provided in Appendix A4. No any other solution is possible since $K = 6$ exhausts all possible vertices.

Product flow amounts are also calculated for the 6-best solutions and provided below in terms of lb/h.

Table 6.9. Product flow values for the Thermal Cracker problem.

	S1	S2	S3	S4	S5	S6
Ethylene	50000	50000	32448.56	50000	50000	0
Propylene	1000	20000	20000	20000	18875	0
Butadiene	1000	5857.30	5596.01	3189.99	2625	0
Gasoline	1000	32815.71	33052.48	12836.52	8875	0
Methane	7000	19613.67	17389.50	31622.97	32125	0
Total	60000	128286.67	108487	117649.48	112500	0

Here, there is only one optimal solution namely, S1, having an objective function value of 330760 ¢/h. In this solution, maximum ethylene production is obtained but propylene production is far below the upper limit of 20000 lb/h. The next best solution, S2, has an objective function value of 187774.72 which differs by 43.2 % compared to the optimal one. Therefore, it may not be selected. In solution S2, maximum possible ethylene production is obtained as in S1, and maximum possible propylene production is attained, and amounts of all products are increased when compared to S1. However, this increase led to a decrease in the objective function (profit). This decrease stems from the large amount of DNG feed.

6.5. A Metabolic Engineering Example

In this section, a metabolic engineering example (Lee *et al.*, 2000) having 33 decision variables is provided. In this problem, the aim is to obtain all of the carbon trafficking alternatives of an *Escherichia coli* (*E. coli*) mutant lacking pyruvate kinase.

Metabolic networks consist of an array of numerous, connected reactions catalyzed by enzymes. Through the activity of these networks, raw materials can be converted to a variety of products, such as amino acids and vitamins, with economic value. One can alter the network structure and/or control mechanisms such as feedback loops to increase product yield. Such manipulation aimed at directing energetic and material resources toward target products and/or away from undesirable waste products is known as metabolic engineering (Lee *et al.*, 2000).

Flux balance analysis (FBA) is a widely used approach for studying biochemical networks, in particular the genome-scale metabolic network reconstructions, which contain all of the known metabolic reactions in an organism and the genes that encode each enzyme. With the use of FBA, the flow of metabolites through the metabolic network is calculated, making it possible to predict the growth rate of an organism or the rate of production of a biotechnologically important metabolite (Orth *et al.*, 2010).

With the GAMS and MATLAB codes explained in the fifth chapter, it is possible to enumerate all alternative ways that the fluxes are distributed in a metabolic network while

still satisfying the same constraints and objective function. The multiple solutions can be used to generate alternative flux scenarios that can account for limited experimental observations, forecast the potential responses to mutation, and design ^{13}C NMR experiments such that different potential flux patterns in a mutant can be distinguished (Phalakornkule *et al.*, 2001).

Metabolic reactions are represented as a stoichiometric matrix ($\boldsymbol{S}$) of size $p \times q$ where, there are p compounds each of which is represented by a row and q reactions each of which is represented by a column. The entries in each column are the stoichiometric coefficients of the metabolites participating in a reaction. There is a negative coefficient for every metabolite consumed and a positive coefficient for every metabolite produced. For metabolites that do not participate in a particular reaction, a stoichiometric coefficient of zero is used. $\boldsymbol{S}$ is a sparse matrix since only a few different metabolites are involved in most biochemical reactions. The vector $\boldsymbol{v}$ of size q represents the flux through all of the reactions in a network. At steady state, the flux through each reaction is given by the following equation.

$$\boldsymbol{Sv} = \boldsymbol{0} \tag{6.35}$$

There are more reactions than there are compounds ($q > p$) in any realistic large-scale metabolic model, meaning that there are more unknown variables than equations, and there is no unique solution to this system of equations. FBA aims at maximizing or minimizing an objective function $Z = \boldsymbol{c}^T\boldsymbol{v}$, which can be any linear combination of fluxes, where $\boldsymbol{c}$ is a vector of weights indicating how much each flux contributes to the objective function. Since optimization of such a system is accomplished by Linear Programming (LP), FBA can be defined as the use of LP to solve Equation 6.35, given a set of upper and lower bounds on $\boldsymbol{v}$ and a linear combination of fluxes, $\boldsymbol{c}$, as the coefficients of objective function (Orth *et al.*, 2010).

The metabolic network of *E. coli* and candidate reversible reactions are provided in Figure 6.13, where glycolysis, the HMP pathway, the TCA cycle, malic enzyme, and the anaplerotic reaction that links phosphoenolpyruvate to oxaloacetate are shown.

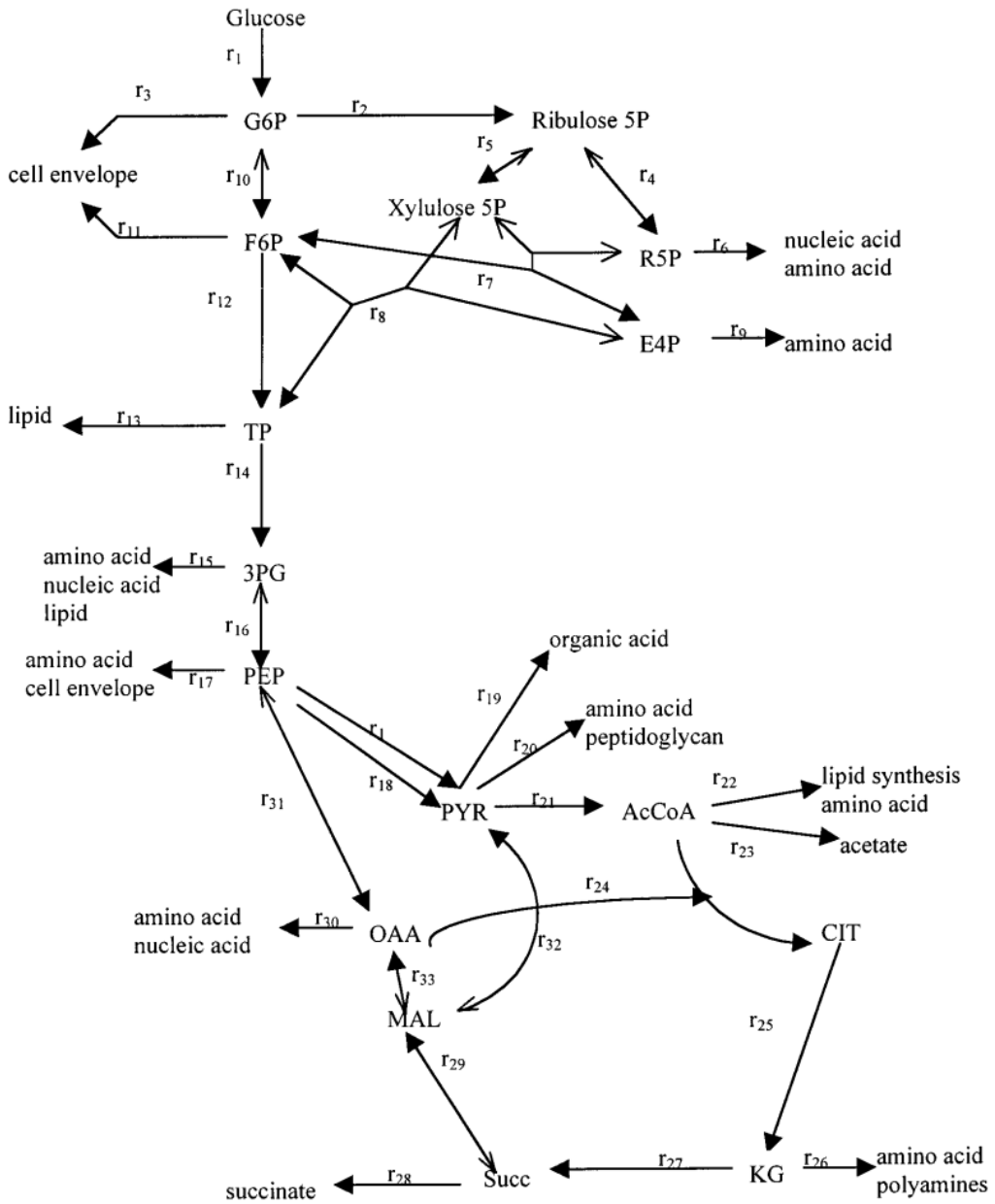

Figure 6.13. The metabolic network of *E. coli* (Phalakornkule *et al.*, 2001).

In Figure 6.13, the double-ended arrows represent potentially reversible fluxes and the solid-head arrows depict the net directions.

The 30 molar balances and constraints for 33 fluxes ($r_i, i = 1, 2, \ldots, 33$) for the network are provided below. First, the 16 molar balances are provided below (Phalakornkule *et al.*, 2001).

$$r_1 - r_2 - r_3 - r_{10} = 0 \tag{6.36}$$

$$r_{10} - r_{12} - r_{11} + r_7 + r_8 = 0 \quad (6.37)$$

$$r_{14} - r_{15} - r_{16} = 0 \quad (6.38)$$

$$-r_1 + r_{16} - r_{18} - r_{17} - r_{31} = 0 \quad (6.39)$$

$$r_1 + r_{18} - r_{21} - r_{19} - r_{20} + r_{32} = 0 \quad (6.40)$$

$$-r_{31} + r_{24} + r_{30} - r_{33} = 0 \quad (6.41)$$

$$r_{21} - r_{24} - r_{23} - r_{22} = 0 \quad (6.42)$$

$$r_{24} - r_{25} = 0 \quad (6.43)$$

$$r_{25} - r_{27} - r_{26} = 0 \quad (6.44)$$

$$-r_{29} + r_{27} - r_{28} = 0 \quad (6.45)$$

$$r_2 - r_4 - r_5 = 0 \quad (6.46)$$

$$-r_7 + r_4 - r_6 = 0 \quad (6.47)$$

$$-r_7 - r_8 + r_5 = 0 \quad (6.48)$$

$$r_7 - r_8 - r_9 = 0 \quad (6.49)$$

$$2r_{12} + r_8 - r_{14} - r_{13} = 0 \quad (6.50)$$

$$r_{29} - r_{33} - r_{32} = 0 \quad (6.51)$$

The 11 biosynthetic requirements that must be fulfilled by the individual reactions are represented by the following equations which were derived from bio-synthetic loads (mmol g^{-1} cell h^{-1}) based on cellular mass composition (Mandelstam *et al.*, 1982; Ingraham *et al.*, 1983; Goel *et al.*, 1996) and a specific growth rate equal to 0.4 h^{-1} (Phalakornkule *et al.*, 2001).

$$2.5r_3 = 0.205 \quad (6.52)$$

$$2.5r_{11} = 0.0709 \quad (6.53)$$

$$2.5r_{13} = 0.129 \quad (6.54)$$

$$2.5r_{15} = 1.493 \quad (6.55)$$

$$2.5r_{17} = 0.7191 \quad (6.56)$$

$$2.5r_6 = 0.897 \quad (6.57)$$

$$2.5r_9 = 0.361 \quad (6.58)$$

$$2.5r_{20} = 2.833 \quad (6.59)$$

$$2.5r_{22} = 2.928 \quad (6.60)$$

$$2.5r_{26} = 1.078 \quad (6.61)$$

$$2.5r_{30} = 1.786 \tag{6.62}$$

The aggregate activity of some reactions must also fulfill some metabolic requirements, which were derived from NADPH and minimum ATP requirements (Goel *et al.*, 1996) and are based on a specific growth rate equal to 0.4 h^{-1} (Phalakornkule *et al.*, 2001).

$$2r_2 + r_{25} + r_{32} = 7.2 \tag{6.63}$$

$$r_{\mathrm{ATP}} + r_{12} - 3r_{14} - r_{18} - r_{23} - 3r_{27} - 2r_{21} - 2r_{33} - r_{29} - r_{31} + 2r_{19} = 0 \tag{6.64}$$

$$r_{\mathrm{ATP}} \geq 13.3 \tag{6.65}$$

where, r_{ATP} denotes the rate of ATP production and Equation 6.64 is based on the assumption that NADH and FADH yield two and one mol of ATP per mole oxidized, respectively (Phalakornkule *et al.*, 2001). Equations 6.64 and 6.65 can be combined to give Equation 6.66 provided below.

$$-r_{12} + 3r_{14} + r_{18} + r_{23} + 3r_{27} + 2r_{21} + 2r_{33} + r_{29} + r_{31} - 2r_{19} \geq 13.3 \tag{6.66}$$

In this problem, the objective is to minimize r_{18} (Phalakornkule *et al.*, 2001) and the objective function is as follows:

$$f = r_{18} \tag{6.67}$$

All fluxes are bounded between 0 and 20 mmol g^{-1} cell h^{-1}, except r_{10} and r_{33} which range between -20 and 20 mmol g^{-1} cell h^{-1} (Phalakornkule *et al.*, 2001).

Carbon yield Y_c, and ATP yield Y_{ATP} are calculated using the following equations (Phalakornkule *et al.*, 2001):

$$Y_c = 500\mu/(72r_1) \tag{6.68}$$

$$Y_{ATP} = (\frac{1000}{r_{ATP}})\mu \tag{6.69}$$

In Equation 6.68, the cells are assumed to consist of 50 % (dry weight basis) carbon and μ denotes the specific growth rate assigned as 0.4 h^{-1} (Phalakornkule *et al.*, 2001).

The problem can be cast into the standard LP form given by Equation 6.2, where, when converted into minimization form, the vectors and matrices take the following values:

$$\boldsymbol{c} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \quad \boldsymbol{b}^1 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0.205 \\ 0.0709 \\ 0.129 \\ 1.493 \\ 0.7191 \\ 0.897 \\ 0.361 \\ 2.833 \\ 2.928 \\ 1.078 \\ 1.786 \\ 7.2 \end{bmatrix}, \quad \boldsymbol{b}^2 = [-13.3], \quad \boldsymbol{x}^L = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ -20 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ -20 \end{bmatrix}, \quad \boldsymbol{x}^U = \begin{bmatrix} 20 \\ 20 \end{bmatrix}$$

$$
\boldsymbol{A}^1 = \begin{bmatrix}
1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & -1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 2 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & -1 & -1 & 0 \\
0 & 0 & 2.5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2.5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2.5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2.5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2.5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2.5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 2.5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2.5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2.5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2.5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2.5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2.5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2.5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0
\end{bmatrix}
$$

$$
\boldsymbol{A}^2 = [0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 1\ \ 0\ \ -3\ \ 0\ \ 0\ \ 0\ \ -1\ \ 2\ \ 0\ \ -2\ \ 0\ \ -1\ \ 0\ \ 0\ \ 0\ \ -3\ \ 0\ \ -1\ \ 0\ \ 1\ \ 0\ \ -2]
$$

The canonical form of LP problem is given by Equation 6.4, where the vectors and matrices take the following values:

$$
\boldsymbol{z} = \begin{bmatrix} \boldsymbol{s}^L_{33\times1} \\ \boldsymbol{s}^U_{33\times1} \\ \boldsymbol{s}_{1\times1} \end{bmatrix},\ \boldsymbol{\alpha} = \begin{bmatrix} \boldsymbol{c}_{33\times1} \\ \boldsymbol{0}_{33\times1} \\ \boldsymbol{0}_{1\times1} \end{bmatrix},\ \boldsymbol{B} = \begin{bmatrix} \boldsymbol{A}^{\boldsymbol{1}}_{28\times33} & \boldsymbol{0}_{28\times33} & \boldsymbol{0}_{28\times1} \\ \boldsymbol{A}^{\boldsymbol{2}}_{1\times33} & \boldsymbol{0}_{1\times33} & \boldsymbol{I}_{1\times1} \\ \boldsymbol{I}_{33\times33} & \boldsymbol{I}_{33\times33} & \boldsymbol{0}_{33\times1} \end{bmatrix},\ \boldsymbol{q} = \begin{bmatrix} (\boldsymbol{b}^1 - \boldsymbol{A}^1\boldsymbol{x}^L)_{28\times1} \\ (\boldsymbol{b}^2 - \boldsymbol{A}^2\boldsymbol{x}^L)_{1\times1} \\ (\boldsymbol{x}^U - \boldsymbol{x}^L)_{33\times1} \end{bmatrix}
$$

where, $\boldsymbol{z}$ and $\boldsymbol{\alpha}$ are $(2n+m)$-dimensional column vectors, $\boldsymbol{B}$ is a $(k+m+n)\times(2n+m)$ coefficient matrix, and $\boldsymbol{q}$ is a $(k+m+n)$-dimensional column vector composed of right hand side values of the equality constraints. $\boldsymbol{I}$ is the identity matrix, n is the number of decision variables, k is the number of equality constraints, and m is the number of inequality constraints in Equation 6.4. For this problem, the number of decision variables, n, is 33, number of equality constraints, k, is 28, and the number of inequality constraints, m, is 1. The coefficient vectors $\boldsymbol{\alpha}$ and $\boldsymbol{q}$, and the coefficient matrix $\boldsymbol{B}$ will not be given explicitly due to their large sizes.

The problem was solved with GAMS and MATLAB, both versions gave the same results. In GAMS, XA was used as the LP and MILP solver. In MATLAB, `linprog` was used to obtain the 1st basic solution, and LPSOLVE is used as the LP and MILP solver for the RMILP section of the algorithm. In both versions the big-M parameter was U = 90 and the termination criterion was ε = 8.

Nine alternative optimal solutions were found, as well as the next 4-best solutions. The results obtained by Prof. Grossmann's group (Phalakornkule *et al.*, 2001) were reproduced. Selected fluxes (mmol g^{-1} cell h^{-1}) from the 5-best solutions (K-best solutions with K=5): $(x_i^*, i = 1, 2, \ldots 33)$ for the nine 1st best solution and $(x_i, i = 1, 2, \ldots 33)$ for the remaining next 5-best solutions obtained, carbon yield, Y_c, and ATP yield, Y_{ATP}, which are calculated from Equations 6.68 and 6.69 are provided in the following table that is formatted as compatible with the table presented by Phalakornkule *et al.* (2001).

Table 6.10. The 5-best solutions of the metabolic engineering example.

Solution		r_1	r_2	r_{10}	r_{18}	r_{19}	r_{23}	r_{27}	r_{32}	f	Y_c	Y_{ATP}
1	1st best	4.01	2.75	1.18	0.00	0.00	0.00	1.27	0.00	0.00	0.69	13.35
2	1st best	4.94	0.65	4.21	0.00	0.00	0.00	3.84	1.63	0.00	0.56	7.28
3	1st best	8.78	0.65	8.05	0.00	0.57	0.00	5.47	0.00	0.00	0.32	5.21
4	1st best	4.22	3.38	0.76	0.00	1.49	0.00	0.00	0.00	0.00	0.66	22.94
5	1st best	4.22	3.38	0.76	0.00	0.00	1.49	0.00	0.00	0.00	0.66	16.08
6	1st best	8.78	0.65	8.05	0.00	0.00	11.52	0.00	5.47	0.00	0.32	5.40
7	1st best	8.78	0.65	8.05	0.00	11.52	0.00	0.00	5.47	0.00	0.32	24.20
8	1st best	8.78	0.65	8.05	0.00	0.00	0.57	5.47	0.00	0.00	0.32	5.02
9	1st best	8.21	0.65	7.48	0.00	0.00	0.00	5.47	0.00	0.00	0.34	5.25
10	2nd best	10.64	0.65	9.91	1.86	4.29	0.00	5.47	0.00	1.86	0.26	4.97
11	2nd best	10.64	0.65	9.91	1.86	0.00	4.29	5.47	0.00	1.86	0.26	3.93
12	2nd best	10.64	0.65	9.91	1.86	15.23	0.00	0.00	5.47	1.86	0.26	19.76
13	2nd best	10.64	0.65	9.91	1.86	0.00	15.23	0.00	5.47	1.86	0.26	4.15
14	3rd best	5.76	0.65	5.03	2.45	0.00	0.00	5.47	0.00	2.45	0.48	5.25
15	4th best	11.10	3.38	7.63	6.87	0.81	14.42	0.00	0.00	6.87	0.25	3.87
16	4th best	11.10	3.38	7.63	6.87	15.23	0.00	0.00	0.00	6.87	0.25	12.83
17	5th best	10.64	0.65	9.91	7.33	6.28	3.48	5.47	0.00	7.33	0.26	3.87
18	5th best	10.64	0.65	9.91	7.33	9.76	0.00	5.47	0.00	7.33	0.26	4.66

where, r_1 is glucose uptake rate, r_2 is glucose-6-phosphate dehydrogenase flux, r_{10} is phosphoglucose isomerase flux, r_{18} is pyruvate kinase flux, r_{19} is lactate dehydrogenase flux, r_{23} is phosphotransacetylase + acetate kinase fluxes, r_{27} is α-ketoglutarate dehydrogenase flux, and r_{32} is malic enzyme flux. The complete set of fluxes (full version of Table 6.10) is given in Appendix B.

For the eighteen set of solutions, the objective function values are provided in Figures 6.11 below.

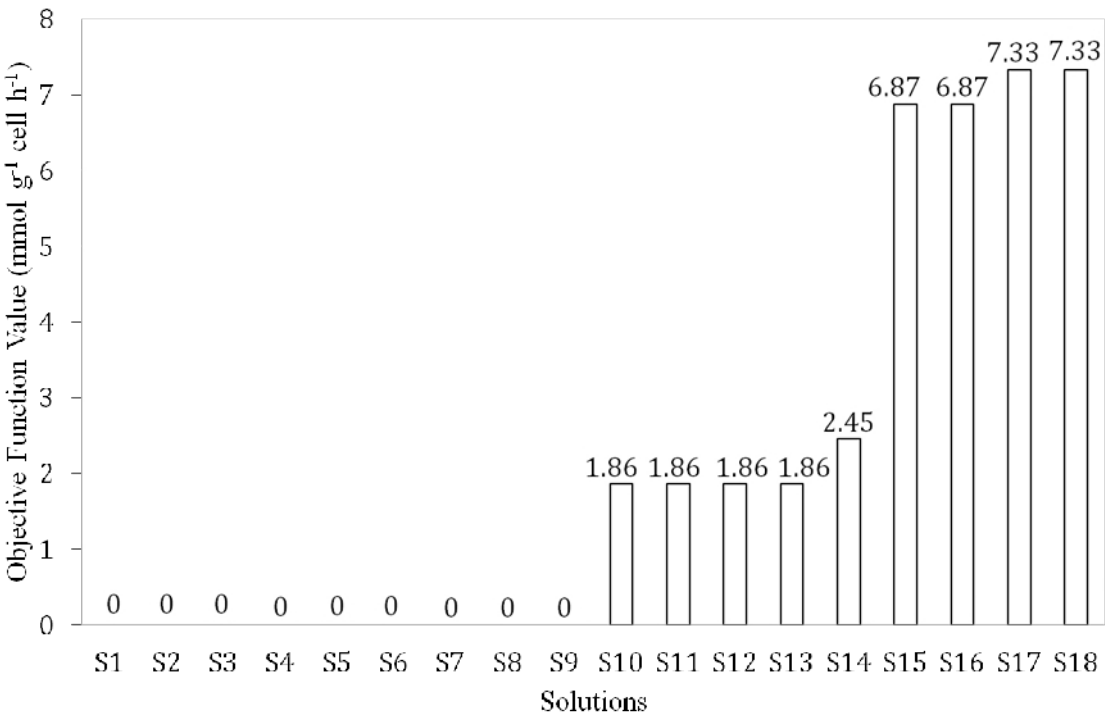

Figure 6.14. The 5 best objective function values of the metabolic engineering example.

The GAMS and MATLAB codes for LP problem given by Equations 6.36-6.67 for non-canonical form are provided in Appendix A5. No any other solution is possible since $K = 5$ exhausts all possible vertices.

In GAMS and MATLAB, depending on the LP/MILP solvers used, some of the solutions are replicated several times. This can easily be seen in small problems having several decision variables. However, in metabolic engineering example, having 33 decision variables, it is not easy to see which solutions provided by GAMS and MATLAB are unique and which ones are replicated. Although it cannot be recognized by glancing, there

is a technique, called Hierarchical Clustering Analysis, to group similar solutions together so that the solutions that are replicated and the ones that are unique are easily recognized. The following section is a brief introduction to hierarchical clustering analysis.

6.5.1. Hierarchical Clustering (HC) Analysis

The process of organizing objects into groups whose members are similar in some way is called clustering, the aim of which is to determine the intrinsic grouping in a set of unlabeled data. A cluster is a collection of objects that are similar to each other and are dissimilar to the objects belonging to other clusters. In distance-based clustering, the similarity criterion is distance: two or more objects belong to the same cluster if they are close according to a given distance.

In one of the most widely used clustering algorithms namely, HC, given a set of N data items to be clustered, and an N×N distance (or, similarity, proximity) matrix, first, each data point is assigned to a cluster, so that for N data, there are N clusters, each containing only one data. At this point, the distances between the clusters are the distances between the data items they contain. The closest (most similar) pair of clusters is found and merged into a single cluster, decreasing the number of clusters by one. Then, the distances between the new cluster and each of the old clusters are computed and these steps are repeated until all items are clustered into a single cluster of size N.

HC results can be presented graphically in the form of a cluster tree called as dendrogram. The clusters and the corresponding dendrogram for a hypothetical data with five objects (data points) labeled from one to five are illustrated in Figure 6.15 below.

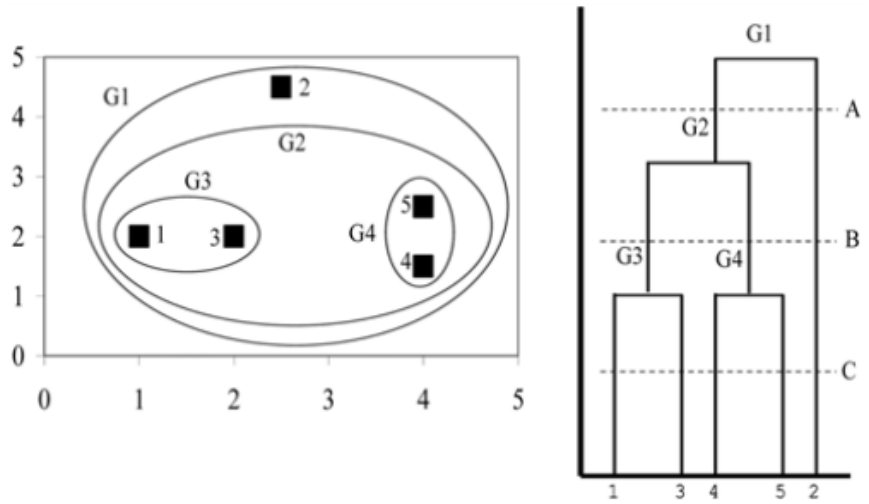

Figure 6.15. An example hierarchical clustering and the corresponding dendrogram (Akman *et al.*, 2008).

In Figure 6.15, the bottom of the dendrogram, along the horizontal axis, shows the leaf nodes that are individual data points namely, 1, 3, 4, 5, and 2. ∏-shaped lines represent links between objects where, the height of the lines indicates the distance between the objects. The clusters labeled G3 and G4, containing only two objects, are formed by points (1, 3) and (4, 5), respectively. Clusters G3 and G4 are then hierarchically grouped into cluster G2. Object 2 and group G2 finally form the top cluster G1 which encloses all the objects. Clusters can be obtained from a dendrogram plot by drawing a horizontal line across the dendrogram bisecting the ∏-shaped branches. For example, in Figure 6.15, the dashed line labeled as C bisects the dendrogram branches five times (at the leaf nodes), indicating five individual objects (five clusters each with one object). The dashed line labeled as B bisects the dendrogram branches three times, indicating three clusters labeled as G3, G4 and the object two. Finally, the dashed line labeled as A bisects the dendrogram branches two times, indicating two clusters labeled as G2 and object two. (Akman *et al.*, 2008).

As an example, when the LP given by Equations 6.36-6.67 is solved using MATLAB such that `linprog` was used to obtain the 1st basic solution, and LPSOLVE is used as the LP and MILP solver for the RMILP section of the algorithm, 24 solutions with an objective function value of zero are obtained, some of which are exactly the same. HC analysis is used to obtain the unique set of solutions that are partially given in Table 6.10 for the selected fluxes and fully given in Appendix B. The dendrogram obtained as the

result of HC analysis for the nine unique 1st best solutions of the metabolic engineering example is provided in Figure 6.16 below.

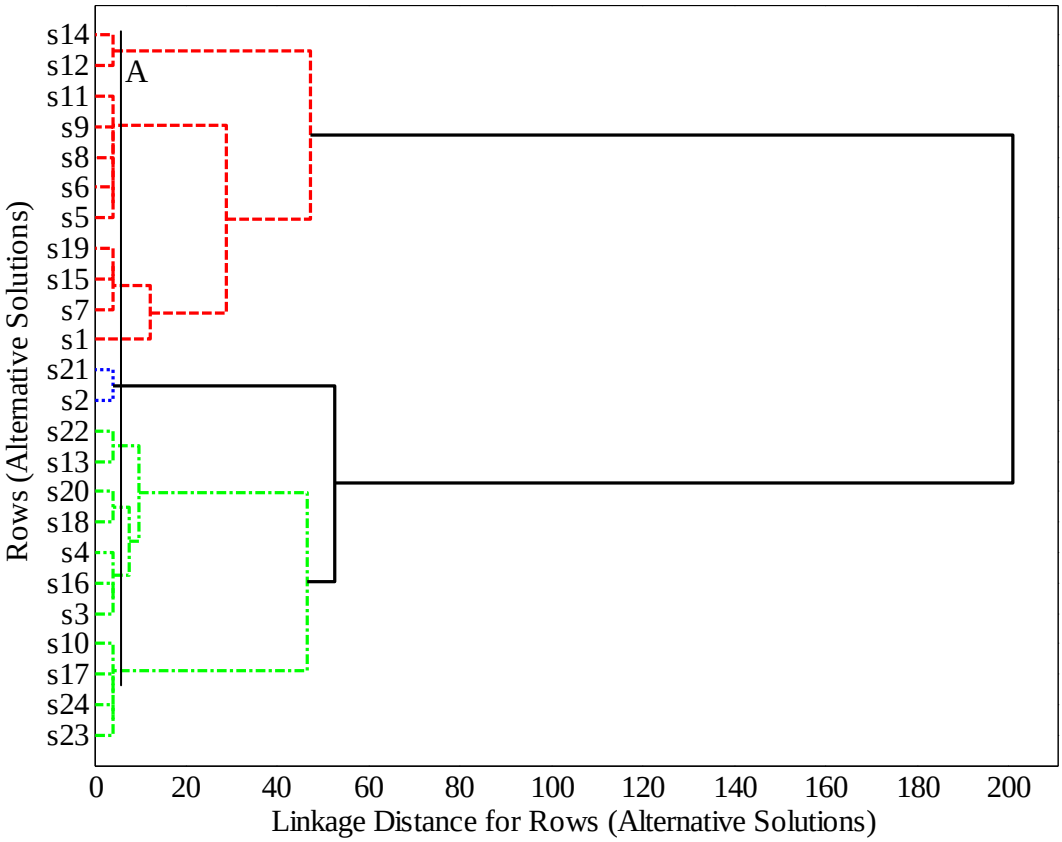

Figure 6.16. Hierarchical clustering dendrogram of alternative solutions (nine unique 1st best solutions) of the metabolic engineering example.

In Figure 6.16, the left of the dendrogram, along the vertical axis, shows the leaf nodes that are individual data points namely, s1, s2, ..., s24, which are the solutions obtained. When seen from the left, the ∏-shaped lines represent links between objects where, the height of the lines indicates the distance between the objects. The vertical line labeled as A bisects the dendrogram branches 9 times indicating the 9 unique alternative optimal solutions. As can be seen in Figure 6.16, solution given by s14 is exactly the same as the one given by s12, the solution given by s1 is unique, and so on.

In Figure 6.17, the left of the dendrogram, along the vertical axis, shows the leaf nodes that are individual data points namely, s1, s2, ..., s54, which are complete set of solutions that are partially given in Table 6.10 for the selected fluxes and fully given in

Appendix B, which were obtained via MATLAB. The vertical line labeled as A bisects the dendrogram branches 18 times indicating the 18 unique solutions provided in Table 6.10 for the selected fluxes.

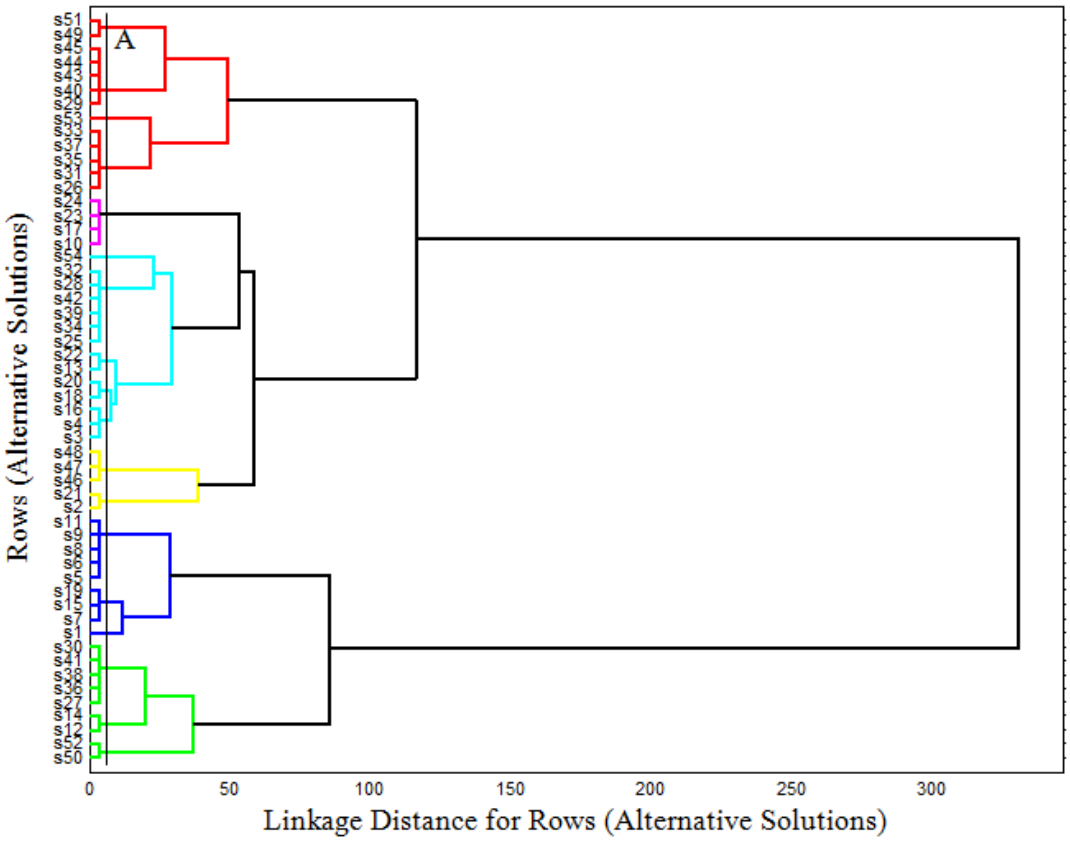

Figure 6.17. Hierarchical clustering dendrogram of all solutions of the metabolic engineering example.

The user may also want to see the similarities and differences between the decision variables (fluxes). The dendrogram representing the HC results of the decision variables (fluxes) of the metabolic engineering example including also r_{ATP}, the rate of ATP production, Y_c, carbon yield, and Y_{ATP}, ATP yield is provided in Figure 6.18 below.

As can be seen in Figure 6.18, the fluxes r_{24} and r_{25} have the same values, r_{20} and r_{22} have similar values, and so on. r_{ATP}, the rate of ATP production has a value that differs significantly from all other variables.

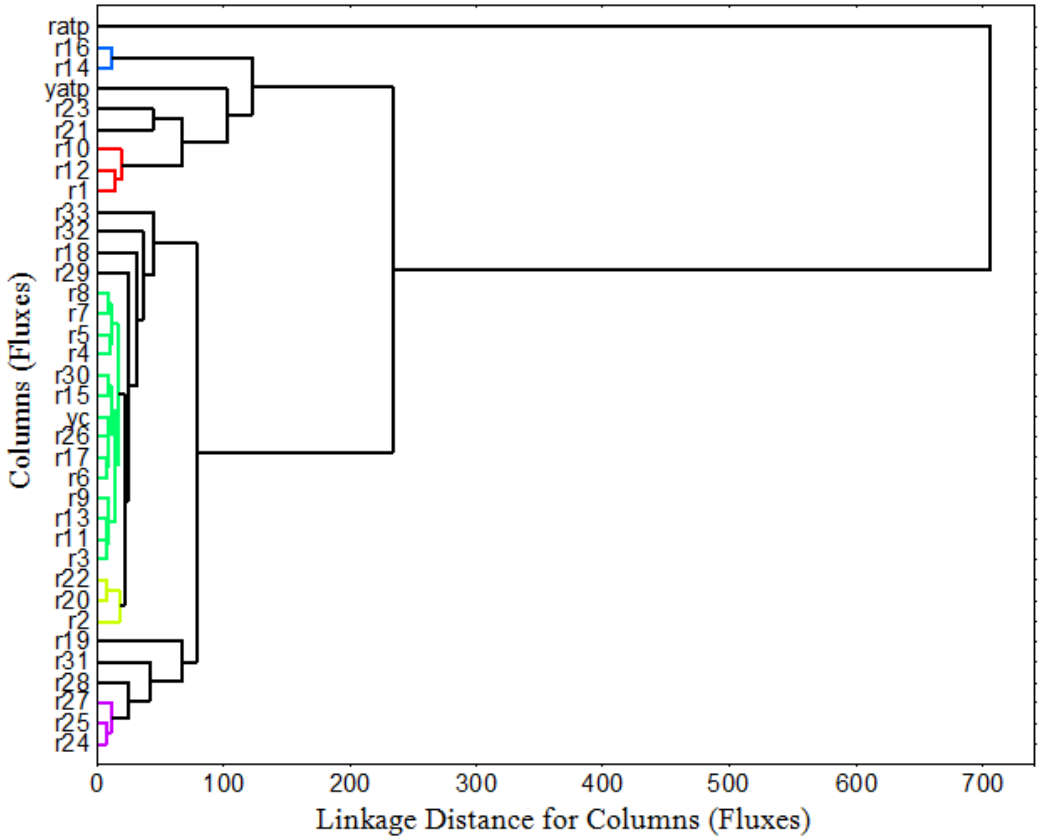

Figure 6.18. Hierarchical clustering dendrogram of fluxes in the metabolic engineering example.

All nine scenarios have the same objective function value that is, all have r_{18} equal to zero, but reveal different ways for carbon to traffic through an *E. coli* network lacking pyruvate kinase. For example, the 1st alternative provided in Table 6.10 has nil acid production and the minimum glucose uptake rate and consequently the maximum carbon yield. Alternatives four and five suggest that organic acid and acetate formation are possible with a higher glucose consumption rate and thus a smaller carbon yield. In alternative two, some malate is transformed to pyruvate via malic enzyme. In alternatives six and seven, net fluxes from oxaloacetate to malate (r_{33}) and from malate to pyruvate (r_{32}) are obtained with zero TCA fluxes (r_{27} and r_{29}). These alternatives thus suggest that the TCA cycle only provides the glutamate precursor,α-ketoglutarate (Phalakornkule *et al.*, 2001).

The GAMS and MATLAB codes also provides the user with the next best solutions, as well as all of the alternate optima. This way, the user knows the difference between the

1st best and the next best solutions. In this problem, the 2nd best solution has an objective function (r_{18}: pyruvate kinase flux) of 1.86 mmol g^{-1} cell h^{-1} and therefore may not be selected.

7. CONCLUSIONS & RECOMMENDATIONS

In this chapter, general experiences gathered during the development of the MATLAB code and implementation on example LP problems are summarized and some suggestions for future work are given.

7.1. Conclusions

Alternate optima are useful in practice in allowing the decision maker to choose from multiple solutions without experiencing any deterioration in the objective function. Furthermore, alternate solutions enable decision maker to make choice by considering other incremental factors (e.g., fuzzy knowledge) that are not (or cannot be) explicitly incorporated into the optimization model. The MATLAB code developed in this work provides the user with all the alternate optimal solutions, as well as next K best solutions of an LP problem with a single run of the LP/RMILP code.

The MATLAB code developed consists of two parts, a function where LP and RMILP problems are solved and a main part where the function is called. The user only enters the problem specific data and termination criterion in the main part. The same function part can be used for all problems without any modification. Therefore, the code can be used even by a novice LP user.

Several LP problems were solved using GAMS and MATLAB and the results were compared. The same results were obtained using the MATLAB code as the GAMS code developed by Prof. Akman.

In both GAMS and MATLAB, some solutions were replicated depending on the LP/MILP solvers used. The presence of replicate solutions can easily be detected in small problems having several decision variables. However, in larger problems, it is not easy to see which solutions provided by GAMS and MATLAB are unique and which ones are replicated. Hierarchical Clustering (HC) technique successfully grouped similar solutions together such that the solutions that were replicated and the ones that were unique were

easily recognized. A dendrogram representing the hierarchical clusters was functional in visualizing the proximity of the alternate solutions as well as the presence of replicate solutions.

7.2. Recommendations for Future Work

In using the GAMS code developed by Prof. Akman, the same solvers were used for both LP and MILP. Different solvers may be used to see if there are any differences. Only LPSOLVE and GLPK were used in MATLAB as MILP solvers. Different MILP solvers may also be tested in MATLAB.

Some solutions were replicated in both GAMS and MATLAB depending on the LP/MILP solvers used. There is a need to determine the unique set of solutions. In this work, dendrograms representing the results of HC analysis were used to be able to detect visually the unique set of solutions. However, the GAMS and MATLAB codes can be modified such that each solution is given once.

Data import/export to/from MATLAB is possible for numerous file formats such as Excel worksheet. MATLAB-Excel interface can be used / developed to export problem specific data written in Excel worksheet into MATLAB and import the results of the MATLAB code into Excel.

In using the GAMS and MATLAB codes, mostly the default values of the solver options such as `mipgap` option in GLPK solver were used. As an example, `mipgap`, the relative MILP gap tolerance, is equal to zero by default. The solver terminates the search if the relative MILP gap for currently known best integer feasible solution falls below this tolerance, allowing the user to obtain suboptimal integer feasible solutions if solving the problem to optimality takes too long. The effects of such options in GAMS and MATLAB may be studied in a future work.

Since there may be uncertainties in the data, generation of the reduced-cost and/or shadow-price information may be quite useful and can be implemented in GAMS and/or MATLAB platforms in a future work.

The termination criterion, ε should be given a relatively large number if the user wants to obtain all possible solutions to an LP problem. An automated selection of this parameter may be implemented in a future work.

There is a standard format for sharing systems biology models among various modeling and simulation software tools which is called Systems Biology Markup Language (SBML). It is a machine-readable format for representing models. SBML enables the use of multiple tools without rewriting models for each tool, sharing and publishing models in a form other researchers can use even in different software environment. In a future work, the MATLAB code developed in this thesis may be modified to import the data required by the LP model directly to MATLAB from SBML file by employing MATLAB's SBML import/export utility functions. In this way, the usability of the MATLAB code developed in this thesis for systems biology applications will be greatly improved.

APPENDIX A: GAMS & MATLAB CODES FOR THE EXAMPLE PROBLEMS

In this section, the main sections of the GAMS and MATLAB codes for the LP problems presented in Chapter 6 are provided.

A.1. A Simple LP

A.1.1. GAMS Code

```
$OFFSYMLIST OFFSYMXREF OFFUELLIST OFFUELXREF
OPTION DECIMALS=8, LIMROW=0, LIMCOL=0, SYSOUT=OFF, SOLPRINT=OFF;
SET N row dimension of x: (n) / 1*2 /;
SET K row dimension of A1: (k) / 1*1 /;
SET M row dimension of A2: (m) / 1*3 /;
SET T maximum number of MILP iterations / 1*20 /;
ALIAS(T,TC);
SCALAR BM big-M parameter / 1000 /;
PARAMETER c(N) vector c /1 -10, 2 -20/;
PARAMETER b1(K) vector b1 /1 0/ ;
PARAMETER b2(M) vector b2 /1 60, 2 50, 3 120/;
TABLE A1(K,N) matrix A1
   1     2
1  0     0;
TABLE A2(M,N) matrix A2
   1     2
1  1     0
2  0     1
3  1     2;
PARAMETER XL(N) lower bounds of decision variables / 1 0, 2 0 /;
PARAMETER XU(N) upper bounds of decision variables / 1 500, 2 500 /;
PARAMETER EPSF alternative solution discrimination criterion / 1200 /;
VARIABLE F     objective function value;
VARIABLE X(N)  decision variables (vector x);
POSITIVE VARIABLES sL(N) lower slcaks
                   sU(N) upper slcaks
```

```
                    s(M)  inequality slcaks;
EQUATIONS
          EQCON(K)   equalities,
          INEQCON(M) inequalities,
          SLSUCON(N) sL-sU=xU-xL equation
          XTOSEQ(N)  x=sL+xL equation
          OBJ objective function definition;
EQCON(K) .. SUM(N, (A1(K,N)*sL(N))) =E= b1(K) - SUM(N, (A1(K,N)*xL(N)));
INEQCON(M) .. SUM(N, (A2(M,N)*sL(N))) + s(M) =E= b2(M) -
              SUM(N, (A2(M,N)*xL(N)));
SLSUCON(N) .. sL(N) + sU(N) =E= xU(N) - xL(N);
XTOSEQ(N)  .. X(N) =E= sL(N) + xL(N);
OBJ        .. SUM(N, (c(N)*sL(N))) + SUM(N, (c(N)*xL(N))) =E= F ;
MODEL NONCANONICALLP / EQCON, INEQCON, SLSUCON, XTOSEQ, OBJ /;
OPTION LP = XA;
PARAMETER TOTAL_CPU total CPU used;
TOTAL_CPU = 0;
SOLVE NONCANONICALLP USING LP MINIMIZING F;
TOTAL_CPU = NONCANONICALLP.RESUSD;
DISPLAY TOTAL_CPU;
DISPLAY F.L, X.L;
DISPLAY sL.L, sU.L, s.L;
```

A.1.2. MATLAB Code

```
clear all; clc;
InputType = 0;
c  = [-10; -20];
b1 = 0;
b2 = [60; 50; 120];
A1 = [0 0];
A2 = [1  0
      0  1
      1  2];
XL = [0; 0];
XU = [500; 500];
T = 20;
EPSF = 1200;
BM = 1000;
if InputType == 0
```

```
    a = 0;
    B = 0;
    q = 0;
end
if InputType == 1
    A1 = 0;
    A2 = 0;
    b1 = 0;
    b2 = 0;
     c = 0;
end
[XSOL,FVALUE] = RecMILP(a,B,q,c,A1,A2,b1,b2,XL,XU,T,EPSF,BM,InputType);
```

A.2. A Degenerate LP

A.2.1. GAMS Code

```
$OFFSYMLIST OFFSYMXREF OFFUELLIST OFFUELXREF
OPTION DECIMALS=8, LIMROW=0, LIMCOL=0, SYSOUT=OFF, SOLPRINT=OFF;
SET N row dimension of x: (n) / 1*3 /;
SET K row dimension of A1: (k) / 1*1 /;
SET M row dimension of A2: (m) / 1*5 /;
SET T maximum number of MILP iterations / 1*20 /;
ALIAS(T,TC);
SCALAR BM big-M parameter / 1000 /;
PARAMETER c(N) vector c / 1 0, 2 0, 3 1 /;
PARAMETER b1(K) vector b1 / 1 0 /;
PARAMETER b2(M) vector b2 / 1 8, 2 4, 3 6, 4 2, 5 6 /;
TABLE A1(K,N)   matrix A1
   1     2     3
1  0     0     0;
TABLE A2(M,N) matrix A2
   1     2     3
1  2     1     1
2  1    -2     1
3 -1     2    -1
4 -1    -2    -1
5  1     1     1;
PARAMETER XL(N) lower bounds of decision variables / 1 -5, 2 -5, 3 0 /;
```

```
PARAMETER XU(N) upper bounds of decision variables / 1 5, 2 5, 3 500 /;
PARAMETER EPSF alternative solution discrimination criterion / 11 /;
VARIABLE F objective function value;
VARIABLE X(N) decision variables (vector x);
POSITIVE VARIABLES sL(N) lower slcaks
                   sU(N) upper slcaks
                   s(M)  inequality slcaks;
EQUATIONS
          EQCON(K)   equalities
          INEQCON(M) inequalities
          SLSUCON(N) sL-sU=xU-xL equation
          XTOSEQ(N)  x=sL+xL equation
          OBJ objective function definition;
EQCON(K) .. SUM(N, (A1(K,N)*sL(N)))=E= b1(K) - SUM(N, (A1(K,N)*xL(N)));
INEQCON(M) .. SUM(N, (A2(M,N)*sL(N))) + s(M) =E= b2(M) -
              SUM(N, (A2(M,N)*xL(N)));
SLSUCON(N) .. sL(N) + sU(N) =E= xU(N) - xL(N);
XTOSEQ(N)  .. X(N) =E= sL(N) + xL(N);
OBJ        .. SUM(N, (c(N)*sL(N))) + SUM(N, (c(N)*xL(N))) =E= F;
MODEL NONCANONICALLP / EQCON, INEQCON, SLSUCON, XTOSEQ, OBJ /;
OPTION LP  = XA;
PARAMETER TOTAL_CPU total CPU used;
TOTAL_CPU = 0;
SOLVE NONCANONICALLP USING LP MINIMIZING F;
TOTAL_CPU = NONCANONICALLP.RESUSD;
DISPLAY TOTAL_CPU;
DISPLAY F.L, X.L;
DISPLAY sL.L, sU.L, s.L;
```

A.2.2. MATLAB Code

```
clear all; clc;
InputType = 0;
c  = [0; 0; 1];
b1 = 0;
b2 = [8; 4; 6; 2; 6];
A1 = zeros(1,3);
A2 = [2  1  1
      1 -2  1
     -1  2 -1
```

```
     -1 -2 -1
      1  1  1];
XL = [-5; -5; 0];
XU = [5; 5; 500];
T = 20;
EPSF = 11;
BM = 1000;
if InputType == 0
    a = 0;
    B = 0;
    q = 0;
end
if InputType == 1
    A1 = 0;
    A2 = 0;
    b1 = 0;
    b2 = 0;
    c = 0;
end
[XSOL,FVALUE] = RecMILP(a,B,q,c,A1,A2,b1,b2,XL,XU,T,EPSF,BM,InputType);
```

A.3. Refinery Blending Problem

A.3.1. GAMS Code

```
$OFFSYMLIST OFFSYMXREF OFFUELLIST OFFUELXREF
OPTION DECIMALS=8, LIMROW=0, LIMCOL=0, SYSOUT=OFF, SOLPRINT=OFF;
SET N row dimension of x: (n) / 1*6 /;
SET K row dimension of A1: (k) / 1*4 /;
SET M row dimension of A2: (m) / 1*3 /;
SET T maximum number of MILP iterations / 1*20 /;
ALIAS(T,TC);
SCALAR BM big-M parameter / 1000000 /;
PARAMETER c(N) vector c /1 24.934, 2 16.112, 3 -36, 4 -24, 5 -21, 6 -10/;
PARAMETER b1(K) vector b1 /1 0, 2 0, 3 0, 4 0/ ;
PARAMETER b2(M) vector b2 /1 24000, 2 2500, 3 6000/ ;
TABLE A1(K,N) matrix A1
      1       2    3   4   5   6
1  0.80    0.44   -1   0   0   0
```

```
2  0.081  0.108    0  -1   0   0
3   0.10   0.36    0   0  -1   0
4  0.019  0.092    0   0   0  -1;
TABLE A2(M,N) matrix A2
    1  2  3  4  5  6
1   0  0  1  0  0  0
2   0  0  0  1  0  0
3   0  0  0  0  1  0;
PARAMETER XL(N) lower bounds of decision variables / 1 0, 2 0, 3 0, 4 0,
5 0, 6 0/;
PARAMETER XU(N) upper bounds of decision variables / 1 50000, 2 50000, 3
50000, 4 50000, 5 50000, 6 50000/;
PARAMETER EPSF alternative solution discrimination criterion / 250001 /;
VARIABLE F     objective function value;
VARIABLE X(N)  decision variables (vector x);
POSITIVE VARIABLES sL(N) lower slcaks
                   sU(N) upper slcaks
                   s(M)  inequality slcaks ;
EQUATIONS
          EQCON(K)   equalities
          INEQCON(M) inequalities
          SLSUCON(N) sL-sU=xU-xL equation
          XTOSEQ(N)  x=sL+xL equation
          OBJ objective function definition;
EQCON(K)   .. SUM(N, (A1(K,N)*sL(N))) =E= b1(K) -
              SUM(N, (A1(K,N)*xL(N)));
INEQCON(M) .. SUM(N, (A2(M,N)*sL(N))) + s(M) =E= b2(M) -
              SUM(N, (A2(M,N)*xL(N)));
SLSUCON(N) .. sL(N) + sU(N) =E= xU(N) - xL(N);
XTOSEQ(N)  .. X(N) =E= sL(N) + xL(N) ;
OBJ        .. SUM(N, (c(N)*sL(N))) + SUM(N, (c(N)*xL(N))) =E= F;
*OBJ        .. SUM(N, (c(N)*sL(N))) =E= F ;
MODEL NONCANONICALLP / EQCON, INEQCON, SLSUCON, XTOSEQ, OBJ /;
OPTION LP = XA;
PARAMETER TOTAL_CPU total CPU used;
TOTAL_CPU = 0;
SOLVE NONCANONICALLP USING LP MINIMIZING F;
TOTAL_CPU = NONCANONICALLP.RESUSD;
DISPLAY TOTAL_CPU;
DISPLAY F.L, X.L;
```

```
DISPLAY sL.L, sU.L, s.L;
```

A.3.2. MATLAB Code

```
clear all; clc;
c  = [24.934; 16.112; -36; -24; -21; -10];
b1 = [0; 0; 0; 0];
b2 = [24000; 2500; 6000];
A1 = [0.80   0.44  -1  0  0  0
     0.081  0.108   0 -1  0  0
      0.10   0.36   0  0 -1  0
     0.019  0.092   0  0  0 -1];
A2 = [0 0 1 0 0 0
      0 0 0 1 0 0
      0 0 0 0 1 0];
XL = [0; 0; 0; 0; 0; 0];
XU = [500000; 500000; 500000; 500000; 500000; 500000];
T = 20;
EPSF = 250001;
BM = 1000000;
InputType = 0;
if InputType == 0
    a = 0;
    B = 0;
    q = 0;
end
if InputType == 1
    A1 = 0;
    A2 = 0;
    b1 = 0;
    b2 = 0;
    c = 0;
    XU = 0;
end
[XSOL,FVALUE] = RecMILP(a,B,q,c,A1,A2,b1,b2,XL,XU,T,EPSF,BM,InputType);
```

A.4. Thermal Cracker Problem

A.4.1. GAMS Code

```
$OFFSYMLIST OFFSYMXREF OFFUELLIST OFFUELXREF
OPTION DECIMALS=8, LIMROW=0, LIMCOL=0, SYSOUT=OFF, SOLPRINT=OFF;
SET N row dimension of x: (n)  / 1*7 /;
SET K row dimension of A1: (k) / 1*3 /;
SET M row dimension of A2: (m) / 1*3 /;
SET T  maximum number of MILP iterations / 1*20 /;
ALIAS(T,TC);
SCALAR  BM  big-M parameter / 1000000 /;
PARAMETER c(N) vector c /1 -0.7376, 2 1.4751, 3 4.3084, 4 0.0476, 5
-7.2876, 6 -8.2549, 7 0/ ;
PARAMETER b1(K) vector b1 /1 0, 2 0, 3 20000000/ ;
PARAMETER b2(M) vector b2 /1 200000, 2 50000, 3 20000/ ;
TABLE A1(K,N) matrix A1
         1        2        3        4        5        6        7
1      0.4     0.06     0.04     0.05     -0.6     0.06        0
2        0      0.1     0.01     0.01        0     -0.9        0
3  -6857.6      364     2032    -1145  -6857.6      364    21520;
TABLE A2(M,N) matrix A2
      1        2        3        4        5        6        7
1   1.1      0.9      0.9        1      1.1      0.9        0
2   0.5     0.35      0.2     0.25      0.5     0.35        0
3  0.01     0.15     0.15     0.18     0.01     0.15        0;
PARAMETER XL(N) lower bounds of decision variables / 1 0, 2 0, 3 0, 4 0,
5 0, 6 0, 7 0/;
PARAMETER XU(N) upper bounds of decision variables / 1 200000, 2 200000,
3 200000, 4 200000, 5 200000, 6 200000, 7 200000/;
PARAMETER EPSF alternative solution discrimination criterion / 335760 /;
VARIABLE F     objective function value ;
VARIABLE X(N)  decision variables (vector x) ;
POSITIVE VARIABLES sL(N) lower slcaks
                   sU(N) upper slcaks
                   s(M)  inequality slcaks ;
EQUATIONS
          EQCON(K)   equalities,
          INEQCON(M) inequalities,
```

```
          SLSUCON(N)  sL-sU=xU-xL equation
          XTOSEQ(N)   x=sL+xL equation
          OBJ objective function definition;
EQCON(K) .. SUM(N, (A1(K,N)*sL(N))) =E= b1(K) - SUM(N, (A1(K,N)*xL(N)));
INEQCON(M) .. SUM(N, (A2(M,N)*sL(N))) + s(M) =E= b2(M) -
              SUM(N, (A2(M,N)*xL(N))) ;
SLSUCON(N) .. sL(N) + sU(N) =E= xU(N) - xL(N) ;
XTOSEQ(N)  .. X(N) =E= sL(N) + xL(N) ;
OBJ .. SUM(N, (c(N)*sL(N))) + SUM(N, (c(N)*xL(N))) =E= F ;
MODEL NONCANONICALLP / EQCON, INEQCON, SLSUCON, XTOSEQ, OBJ /;
OPTION LP  = XA;
PARAMETER TOTAL_CPU total CPU used;
TOTAL_CPU = 0;
SOLVE NONCANONICALLP USING LP MINIMIZING F;
TOTAL_CPU = NONCANONICALLP.RESUSD;
DISPLAY TOTAL_CPU;
DISPLAY F.L, X.L;
DISPLAY sL.L, sU.L, s.L;
```

A.4.2. MATLAB Code

```
clear all; clc;
c  = [-0.7376; 1.4751; 4.3084; 0.0476; -7.2876; -8.2549; 0];
b1 = [0; 0; 20000000];
b2 = [200000; 50000; 20000];
A1 = [0.4  0.06   0.04  0.05     -0.6   0.06      0
        0   0.1   0.01  0.01        0   -0.9      0
  -6857.6   364   2032 -1145  -6857.6    364  21520];
A2 = [1.1   0.9   0.9     1   1.1  0.9  0
      0.5  0.35   0.2  0.25   0.5  0.35 0
     0.01  0.15  0.15  0.18  0.01  0.15 0];
XL = [0; 0; 0; 0; 0; 0; 0];
XU = [200000; 200000; 200000; 200000; 200000; 200000; 200000];
T = 20;
EPSF = 335760;
BM = 1000000;
InputType = 0;
if InputType == 0
    a = 0;
    B = 0;
```

```
    q = 0;
end
if InputType == 1
    A1 = 0;
    A2 = 0;
    b1 = 0;
    b2 = 0;
    c = 0;
    XU = 0;
end
[XSOL,FVALUE] = RecMILP(a,B,q,c,A1,A2,b1,b2,XL,XU,T,EPSF,BM,InputType);
```

A.5. Metabolic Engineering Example

A.5.1. GAMS Code

```
$OFFSYMLIST OFFSYMXREF OFFUELLIST OFFUELXREF
OPTION DECIMALS=8, LIMROW=0, LIMCOL=0, SYSOUT=OFF, SOLPRINT=OFF;
SET N row dimension of x: (n) / 1*33 /;
SET K row dimension of A1: (k) / 1*28 /;
SET M row dimension of A2: (m) / 1*1 /;
SET T maximum number of MILP iterations / 1*50 /;
ALIAS(T,TC);
SCALAR  BM  big-M parameter / 1000000 /;
PARAMETER c(N) vector c /1 0, 2 0, 3 0, 4 0, 5 0, 6 0, 7 0, 8 0, 9 0, 10
0, 11 0, 12 0, 13 0, 14 0, 15 0, 16 0, 17 0, 18 1, 19 0, 20 0, 21 0, 22
0, 23 0, 24 0, 25 0, 26 0, 27 0, 28 0, 29 0, 30 0, 31 0, 32 0, 33 0/;
PARAMETER b1(K) vector b1 /1 0, 2 0, 3 0, 4 0, 5 0, 6 0, 7 0, 8 0, 9 0,
10 0, 11 0, 12 0, 13 0, 14 0, 15 0, 16 0, 17 0.205, 18 0.0709, 19 0.129,
20 1.493, 21 0.7197, 22 0.897, 23 0.361, 24 2.833, 25 2.928, 26 1.078, 27
1.786, 28 7.2/;
PARAMETER b2(M) vector b2 /1 -13.3/;
TABLE A1(K,N)   matrix A1
   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  26  27  28  29  30  31  32  33
1  1  -1  -1   0   0   0   0   0   0  -1   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
2  0   0   0   0   0   0   1   1   0   1  -1  -1   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
```

3	0	0	0	0	0	0	0	0	0	0	0	0	0	1	-1	-1	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
4	-1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	-1	-1
	0	0	0	0	0	0	0	0	0	0	0	0	-1	0	0			
5	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
	-1	-1	-1	0	0	0	0	0	0	0	0	0	0	1	0			
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	1	0	0	0	0	0	1	-1	0	-1			
7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	1	-1	-1	-1	0	0	0	0	0	0	0	0	0			
8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	1	-1	0	0	0	0	0	0	0	0			
9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	1	-1	-1	0	0	0	0	0	0			
10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	1	-1	-1	0	0	0	0			
11	0	1	0	-1	-1	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
12	0	0	0	1	0	-1	-1	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
13	0	0	0	0	1	0	-1	-1	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
14	0	0	0	0	0	0	1	-1	-1	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
15	0	0	0	0	0	0	0	1	0	0	0	2	-1	-1	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	1	0	0	-1	-1			
17	0	0	2.5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
18	0	0	0	0	0	0	0	0	0	0	2.5	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
19	0	0	0	0	0	0	0	0	0	0	0	0	2.5	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2.5	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2.5	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
22	0	0	0	0	0	2.5	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			

```
23 0   0   0   0   0   0   0   0 2.5   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
24 0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0 2.5   0   0   0   0   0   0   0   0   0   0   0   0   0
25 0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0 2.5   0   0   0   0   0   0   0   0   0   0   0
26 0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0 2.5   0   0   0   0   0   0   0
27 0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0 2.5   0   0   0
28 0   2   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   1   0   0   0   0   0   0   1   0;
TABLE A2(M,N) matrix A2
   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  26  27  28  29  30  31  32  33
1  0   0   0   0   0   0   0   0   0   0   0   1   0  -3   0   0   0  -1
   2   0  -2  -1  -1   0   0   0  -1   0  -1   0  -1   0  -2;
PARAMETER XL(N) lower bounds of decision variables / 1 0, 2 0, 3 0, 4 0,
5 0, 6 0, 7 0, 8 0, 9 0, 10 -20, 11 0, 12 0, 13 0, 14 0, 15 0, 16 0, 17
0, 18 0, 19 0, 20 0, 21 0, 22 0, 23 0, 24 0, 25 0, 26 0, 27 0, 28 0, 29
0, 30 0, 31 0, 32 0, 33 -20/;
PARAMETER XU(N) upper bounds of decision variables / 1 20, 2 20, 3 20, 4
20, 5 20, 6 20, 7 20, 8 20, 9 20, 10 20, 11 20, 12 20, 13 20, 14 20, 15
20, 16 20, 17 20, 18 20, 19 20, 20 20, 21 20, 22 20, 23 20, 24 20, 25 20,
26 20, 27 20, 28 20, 29 20, 30 20, 31 20, 32 20, 33 20/;
PARAMETER EPSF alternative solution discrimination criterion / 8 /;
VARIABLE F     objective function value ;
VARIABLE X(N)  decision variables (vector x) ;
POSITIVE VARIABLES sL(N) lower slcaks
                   sU(N) upper slcaks
                   s(M)  inequality slcaks ;
EQUATIONS
          EQCON(K)   equalities,
          INEQCON(M) inequalities,
          SLSUCON(N) sL-sU=xU-xL equation
          XTOSEQ(N)  x=sL+xL equation
          OBJ objective function definition;
EQCON(K) .. SUM(N, (A1(K,N)*sL(N))) =E= b1(K) - SUM(N, (A1(K,N)*xL(N)));
INEQCON(M) .. SUM(N, (A2(M,N)*sL(N))) + s(M) =E= b2(M) -
              SUM(N, (A2(M,N)*xL(N)));
```

```
SLSUCON(N) .. sL(N) + sU(N) =E= xU(N) - xL(N) ;
XTOSEQ(N)  .. X(N) =E= sL(N) + xL(N) ;
OBJ        .. SUM(N, (c(N)*sL(N))) + SUM(N, (c(N)*xL(N))) =E= F ;
MODEL NONCANONICALLP / EQCON, INEQCON, SLSUCON, XTOSEQ, OBJ /;
OPTION LP = XA;
PARAMETER TOTAL_CPU total CPU used;
TOTAL_CPU = 0;
SOLVE NONCANONICALLP USING LP MINIMIZING F;
TOTAL_CPU = NONCANONICALLP.RESUSD;
DISPLAY TOTAL_CPU;
DISPLAY F.L, X.L;
DISPLAY sL.L, sU.L, s.L;
```

A.5.2. MATLAB Code

```
clear all; clc;
InputType = 0;
c  = [0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;1;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0];
b1 = [0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0.205;0.0709;0.129;1.493;0.7191;
      0.897;0.361;2.833;2.928;1.078;1.786;7.2];
b2 = -13.3;
A1 = [1  -1  -1   0   0   0   0   0   0  -1   0   0   0   0   0   0   0
      0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
      0   0   0   0   0   0   1   1   0   1  -1  -1   0   0   0   0   0
      0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
      0   0   0   0   0   0   0   0   0   0   0   0   0   1  -1  -1   0
      0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
     -1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   1  -1
     -1   0   0   0   0   0   0   0   0   0   0   0   0  -1   0   0
      1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
      1  -1  -1  -1   0   0   0   0   0   0   0   0   0   0   1   0
      0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
      0   0   0   0   0   0   1   0   0   0   0   0   1  -1   0  -1
      0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
      0   0   0   1  -1  -1  -1   0   0   0   0   0   0   0   0   0
      0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
      0   0   0   0   0   0   1  -1   0   0   0   0   0   0   0   0
      0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
      0   0   0   0   0   0   0   1  -1  -1   0   0   0   0   0   0
      0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
```

```
0   0   0   0   0   0   0   0   0   1  -1  -1   0   0   0   0
0   1   0  -1  -1   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   1   0  -1  -1   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   1   0  -1  -1   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   1  -1  -1   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   1   0   0   0   2  -1  -1   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   1   0   0  -1  -1
0   0 2.5   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0 2.5   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0 2.5   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0 2.5   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 2.5
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0 2.5   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0 2.5   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0 2.5   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0 2.5   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0 2.5   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0 2.5 0   0   0
0   2   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   1   0   0   0   0   0   0   1   0];
A2 = [0 0 0 0 0 0 0 0 0 0 0 1 0 -3 0 0 0 -1 2 0 -2 0 -1 0 0 0 -3 0 -1 0
     -1 0 -2];
XL = [0;0;0;0;0;0;0;0;0;-20;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;
```

```
     -20];
XU = [20;20;20;20;20;20;20;20;20;20;20;20;20;20;20;20;20;20;20;20;20;20;
      20;20;20;20;20;20;20;20;20;20;20];
T = 100;
EPSF = 8;
BM = 90;
if InputType == 0
    a = 0;
    B = 0;
    q = 0;
end
if InputType == 1
    A1 = 0;
    A2 = 0;
    b1 = 0;
    b2 = 0;
    c = 0;
    XU = 0;
end
[XSOL,FVALUE] = RecMILP(a,B,q,c,A1,A2,b1,b2,XL,XU,T,EPSF,BM,InputType);
```

APPENDIX B: SOLUTIONS OF THE METABOLIC ENGINEERING EXAMPLE

Table A.1. Solutions from S1 to S9 for 54 solutions of the metabolic engineering example.

	S1	S2	S3	S4	S5	S6	S7	S8	S9
r_1	4.01	4.94	8.78	8.78	4.22	4.22	4.22	4.22	4.22
r_2	2.75	0.65	0.65	0.65	3.38	3.38	3.38	3.38	3.38
r_3	0.08	0.08	0.08	0.08	0.08	0.08	0.08	0.08	0.08
r_4	1.20	0.50	0.50	0.50	1.42	1.42	1.42	1.42	1.42
r_5	1.54	0.14	0.14	0.14	1.97	1.97	1.97	1.97	1.97
r_6	0.36	0.36	0.36	0.36	0.36	0.36	0.36	0.36	0.36
r_7	0.84	0.14	0.14	0.14	1.06	1.06	1.06	1.06	1.06
r_8	0.70	0.00	0.00	0.00	0.91	0.91	0.91	0.91	0.91
r_9	0.14	0.14	0.14	0.14	0.14	0.14	0.14	0.14	0.14
r_{10}	1.18	4.21	8.05	8.05	0.76	0.76	0.76	0.76	0.76
r_{11}	0.03	0.03	0.03	0.03	0.03	0.03	0.03	0.03	0.03
r_{12}	2.70	4.33	8.17	8.17	2.70	2.70	2.70	2.70	2.70
r_{13}	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05
r_{14}	6.04	8.61	16.29	16.29	6.25	6.25	6.25	6.25	6.25
r_{15}	0.60	0.60	0.60	0.60	0.60	0.60	0.60	0.60	0.60
r_{16}	5.44	8.01	15.69	15.69	5.65	5.65	5.65	5.65	5.65
r_{17}	0.29	0.29	0.29	0.29	0.29	0.29	0.29	0.29	0.29
r_{18}	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
r_{19}	0.00	0.00	0.57	0.57	1.49	1.49	0.00	1.49	1.49
r_{20}	1.13	1.13	1.13	1.13	1.13	1.13	1.13	1.13	1.13
r_{21}	2.88	5.44	7.08	7.08	1.60	1.60	3.09	1.60	1.60
r_{22}	1.17	1.17	1.17	1.17	1.17	1.17	1.17	1.17	1.17
r_{23}	0.00	0.00	0.00	0.00	0.00	0.00	1.49	0.00	0.00
r_{24}	1.70	4.27	5.90	5.90	0.43	0.43	0.43	0.43	0.43
r_{25}	1.70	4.27	5.90	5.90	0.43	0.43	0.43	0.43	0.43
r_{26}	0.43	0.43	0.43	0.43	0.43	0.43	0.43	0.43	0.43
r_{27}	1.27	3.84	5.47	5.47	0.00	0.00	0.00	0.00	0.00
r_{28}	0.00	0.00	5.47	5.47	0.00	0.00	0.00	0.00	0.00
r_{29}	1.27	3.84	0.00	0.00	0.00	0.00	0.00	0.00	0.00
r_{30}	0.71	0.71	0.71	0.71	0.71	0.71	0.71	0.71	0.71
r_{31}	1.15	2.78	6.62	6.62	1.15	1.15	1.15	1.15	1.15
r_{32}	0.00	1.63	0.00	0.00	0.00	0.00	0.00	0.00	0.00
r_{33}	1.27	2.21	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Y_C	0.69	0.56	0.32	0.32	0.66	0.66	0.66	0.66	0.66
Y_{ATP}	13.35	7.28	5.21	5.21	22.94	22.94	16.08	22.94	22.94
ratp	29.96	54.93	76.74	76.74	17.44	17.44	24.87	17.44	17.44

Table A.2. Solutions from S10 to S18 for 54 solutions of the metabolic engineering example (cont.).

	S10	S11	S12	S13	S14	S15	S16	S17	S18
r_1	8.78	4.22	8.78	8.78	8.78	4.22	8.78	8.78	8.21
r_2	0.65	3.38	0.65	0.65	0.65	3.38	0.65	0.65	0.65
r_3	0.08	0.08	0.08	0.08	0.08	0.08	0.08	0.08	0.08
r_4	0.50	1.42	0.50	0.50	0.50	1.42	0.50	0.50	0.50
r_5	0.14	1.97	0.14	0.14	0.14	1.97	0.14	0.14	0.14
r_6	0.36	0.36	0.36	0.36	0.36	0.36	0.36	0.36	0.36
r_7	0.14	1.06	0.14	0.14	0.14	1.06	0.14	0.14	0.14
r_8	0.00	0.91	0.00	0.00	0.00	0.91	0.00	0.00	0.00
r_9	0.14	0.14	0.14	0.14	0.14	0.14	0.14	0.14	0.14
r_{10}	8.05	0.76	8.05	8.05	8.05	0.76	8.05	8.05	7.48
r_{11}	0.03	0.03	0.03	0.03	0.03	0.03	0.03	0.03	0.03
r_{12}	8.17	2.70	8.17	8.17	8.17	2.70	8.17	8.17	7.60
r_{13}	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05
r_{14}	16.29	6.25	16.29	16.29	16.29	6.25	16.29	16.29	15.14
r_{15}	0.60	0.60	0.60	0.60	0.60	0.60	0.60	0.60	0.60
r_{16}	15.69	5.65	15.69	15.69	15.69	5.65	15.69	15.69	14.54
r_{17}	0.29	0.29	0.29	0.29	0.29	0.29	0.29	0.29	0.29
r_{18}	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
r_{19}	0.00	1.49	11.52	0.00	11.52	0.00	0.57	0.00	0.00
r_{20}	1.13	1.13	1.13	1.13	1.13	1.13	1.13	1.13	1.13
r_{21}	13.12	1.60	1.60	7.65	1.60	3.09	7.08	13.12	7.08
r_{22}	1.17	1.17	1.17	1.17	1.17	1.17	1.17	1.17	1.17
r_{23}	11.52	0.00	0.00	0.57	0.00	1.49	0.00	11.52	0.00
r_{24}	0.43	0.43	0.43	5.90	0.43	0.43	5.90	0.43	5.90
r_{25}	0.43	0.43	0.43	5.90	0.43	0.43	5.90	0.43	5.90
r_{26}	0.43	0.43	0.43	0.43	0.43	0.43	0.43	0.43	0.43
r_{27}	0.00	0.00	0.00	5.47	0.00	0.00	5.47	0.00	5.47
r_{28}	0.00	0.00	0.00	5.47	0.00	0.00	5.47	0.00	4.90
r_{29}	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.57
r_{30}	0.71	0.71	0.71	0.71	0.71	0.71	0.71	0.71	0.71
r_{31}	6.62	1.15	6.62	6.62	6.62	1.15	6.62	6.62	6.05
r_{32}	5.47	0.00	5.47	0.00	5.47	0.00	0.00	5.47	0.00
r_{33}	-5.47	0.00	-5.47	0.00	-5.47	0.00	0.00	-5.47	0.57
Y_C	0.32	0.66	0.32	0.32	0.32	0.66	0.32	0.32	0.34
Y_{ATP}	5.40	22.94	24.20	5.02	24.20	16.08	5.21	5.40	5.25
ratp	74.13	17.44	16.53	79.60	16.53	24.87	76.74	74.13	76.16

Table A.3. Solutions from S19 to S27 for 54 solutions of the metabolic engineering example (cont.).

	S19	S20	S21	S22	S23	S24	S25	S26	S27
r_1	4.22	8.21	4.94	8.78	8.78	8.78	10.64	10.64	10.64
r_2	3.38	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65
r_3	0.08	0.08	0.08	0.08	0.08	0.08	0.08	0.08	0.08
r_4	1.42	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50
r_5	1.97	0.14	0.14	0.14	0.14	0.14	0.14	0.14	0.14
r_6	0.36	0.36	0.36	0.36	0.36	0.36	0.36	0.36	0.36
r_7	1.06	0.14	0.14	0.14	0.14	0.14	0.14	0.14	0.14
r_8	0.91	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
r_9	0.14	0.14	0.14	0.14	0.14	0.14	0.14	0.14	0.14
r_{10}	0.76	7.48	4.21	8.05	8.05	8.05	9.91	9.91	9.91
r_{11}	0.03	0.03	0.03	0.03	0.03	0.03	0.03	0.03	0.03
r_{12}	2.70	7.60	4.33	8.17	8.17	8.17	10.03	10.03	10.03
r_{13}	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05
r_{14}	6.25	15.14	8.61	16.29	16.29	16.29	20.00	20.00	20.00
r_{15}	0.60	0.60	0.60	0.60	0.60	0.60	0.60	0.60	0.60
r_{16}	5.65	14.54	8.01	15.69	15.69	15.69	19.40	19.40	19.40
r_{17}	0.29	0.29	0.29	0.29	0.29	0.29	0.29	0.29	0.29
r_{18}	0.00	0.00	0.00	0.00	0.00	0.00	1.86	1.86	1.86
r_{19}	0.00	0.00	0.00	0.00	0.00	0.00	4.29	0.00	15.23
r_{20}	1.13	1.13	1.13	1.13	1.13	1.13	1.13	1.13	1.13
r_{21}	3.09	7.08	5.44	7.65	13.12	13.12	7.08	11.36	1.60
r_{22}	1.17	1.17	1.17	1.17	1.17	1.17	1.17	1.17	1.17
r_{23}	1.49	0.00	0.00	0.57	11.52	11.52	0.00	4.29	0.00
r_{24}	0.43	5.90	4.27	5.90	0.43	0.43	5.90	5.90	0.43
r_{25}	0.43	5.90	4.27	5.90	0.43	0.43	5.90	5.90	0.43
r_{26}	0.43	0.43	0.43	0.43	0.43	0.43	0.43	0.43	0.43
r_{27}	0.00	5.47	3.84	5.47	0.00	0.00	5.47	5.47	0.00
r_{28}	0.00	4.90	0.00	5.47	0.00	0.00	5.47	5.47	0.00
r_{29}	0.00	0.57	3.84	0.00	0.00	0.00	0.00	0.00	0.00
r_{30}	0.71	0.71	0.71	0.71	0.71	0.71	0.71	0.71	0.71
r_{31}	1.15	6.05	2.78	6.62	6.62	6.62	6.62	6.62	6.62
r_{32}	0.00	0.00	1.63	0.00	5.47	5.47	0.00	0.00	5.47
r_{33}	0.00	0.57	2.21	0.00	-5.47	-5.47	0.00	0.00	-5.47
Y_C	0.66	0.34	0.56	0.32	0.32	0.32	0.26	0.26	0.26
Y_{ATP}	16.08	5.25	7.28	5.02	5.40	5.40	4.97	3.93	19.76
ratp	24.87	76.16	54.93	79.60	74.13	74.13	80.45	101.88	20.24

Table A.4. Solutions from S28 to S36 for 54 solutions of the metabolic engineering example (cont.).

	S28	S29	S30	S31	S32	S33	S34	S35	S36
r_1	10.64	10.64	10.64	10.64	10.64	10.64	10.64	10.64	10.64
r_2	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65
r_3	0.08	0.08	0.08	0.08	0.08	0.08	0.08	0.08	0.08
r_4	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50
r_5	0.14	0.14	0.14	0.14	0.14	0.14	0.14	0.14	0.14
r_6	0.36	0.36	0.36	0.36	0.36	0.36	0.36	0.36	0.36
r_7	0.14	0.14	0.14	0.14	0.14	0.14	0.14	0.14	0.14
r_8	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
r_9	0.14	0.14	0.14	0.14	0.14	0.14	0.14	0.14	0.14
r_{10}	9.91	9.91	9.91	9.91	9.91	9.91	9.91	9.91	9.91
r_{11}	0.03	0.03	0.03	0.03	0.03	0.03	0.03	0.03	0.03
r_{12}	10.03	10.03	10.03	10.03	10.03	10.03	10.03	10.03	10.03
r_{13}	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05
r_{14}	20.00	20.00	20.00	20.00	20.00	20.00	20.00	20.00	20.00
r_{15}	0.60	0.60	0.60	0.60	0.60	0.60	0.60	0.60	0.60
r_{16}	19.40	19.40	19.40	19.40	19.40	19.40	19.40	19.40	19.40
r_{17}	0.29	0.29	0.29	0.29	0.29	0.29	0.29	0.29	0.29
r_{18}	1.86	1.86	1.86	1.86	1.86	1.86	1.86	1.86	1.86
r_{19}	4.29	0.00	15.23	0.00	4.29	0.00	4.29	0.00	15.23
r_{20}	1.13	1.13	1.13	1.13	1.13	1.13	1.13	1.13	1.13
r_{21}	7.08	16.84	1.60	11.36	7.08	11.36	7.08	11.36	1.60
r_{22}	1.17	1.17	1.17	1.17	1.17	1.17	1.17	1.17	1.17
r_{23}	0.00	15.23	0.00	4.29	0.00	4.29	0.00	4.29	0.00
r_{24}	5.90	0.43	0.43	5.90	5.90	5.90	5.90	5.90	0.43
r_{25}	5.90	0.43	0.43	5.90	5.90	5.90	5.90	5.90	0.43
r_{26}	0.43	0.43	0.43	0.43	0.43	0.43	0.43	0.43	0.43
r_{27}	5.47	0.00	0.00	5.47	5.47	5.47	5.47	5.47	0.00
r_{28}	5.47	0.00	0.00	5.47	5.47	5.47	5.47	5.47	0.00
r_{29}	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
r_{30}	0.71	0.71	0.71	0.71	0.71	0.71	0.71	0.71	0.71
r_{31}	6.62	6.62	6.62	6.62	6.62	6.62	6.62	6.62	6.62
r_{32}	0.00	5.47	5.47	0.00	0.00	0.00	0.00	0.00	5.47
r_{33}	0.00	-5.47	-5.47	0.00	0.00	0.00	0.00	0.00	-5.47
Y_C	0.26	0.26	0.26	0.26	0.26	0.26	0.26	0.26	0.26
Y_{ATP}	4.97	4.15	19.76	3.93	4.97	3.93	4.97	3.93	19.76
ratp	80.45	96.41	20.24	101.88	80.45	101.88	80.45	101.88	20.24

Table A.5. Solutions from S37 to S45 for 54 solutions of the metabolic engineering example (cont.).

	S37	S38	S39	S40	S41	S42	S43	S44	S45
r_1	10.64	10.64	10.64	10.64	10.64	10.64	10.64	10.64	10.64
r_2	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65
r_3	0.08	0.08	0.08	0.08	0.08	0.08	0.08	0.08	0.08
r_4	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50
r_5	0.14	0.14	0.14	0.14	0.14	0.14	0.14	0.14	0.14
r_6	0.36	0.36	0.36	0.36	0.36	0.36	0.36	0.36	0.36
r_7	0.14	0.14	0.14	0.14	0.14	0.14	0.14	0.14	0.14
r_8	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
r_9	0.14	0.14	0.14	0.14	0.14	0.14	0.14	0.14	0.14
r_{10}	9.91	9.91	9.91	9.91	9.91	9.91	9.91	9.91	9.91
r_{11}	0.03	0.03	0.03	0.03	0.03	0.03	0.03	0.03	0.03
r_{12}	10.03	10.03	10.03	10.03	10.03	10.03	10.03	10.03	10.03
r_{13}	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05
r_{14}	20.00	20.00	20.00	20.00	20.00	20.00	20.00	20.00	20.00
r_{15}	0.60	0.60	0.60	0.60	0.60	0.60	0.60	0.60	0.60
r_{16}	19.40	19.40	19.40	19.40	19.40	19.40	19.40	19.40	19.40
r_{17}	0.29	0.29	0.29	0.29	0.29	0.29	0.29	0.29	0.29
r_{18}	1.86	1.86	1.86	1.86	1.86	1.86	1.86	1.86	1.86
r_{19}	0.00	15.23	4.29	0.00	15.23	4.29	0.00	0.00	0.00
r_{20}	1.13	1.13	1.13	1.13	1.13	1.13	1.13	1.13	1.13
r_{21}	11.36	1.60	7.08	16.84	1.60	7.08	16.84	16.84	16.84
r_{22}	1.17	1.17	1.17	1.17	1.17	1.17	1.17	1.17	1.17
r_{23}	4.29	0.00	0.00	15.23	0.00	0.00	15.23	15.23	15.23
r_{24}	5.90	0.43	5.90	0.43	0.43	5.90	0.43	0.43	0.43
r_{25}	5.90	0.43	5.90	0.43	0.43	5.90	0.43	0.43	0.43
r_{26}	0.43	0.43	0.43	0.43	0.43	0.43	0.43	0.43	0.43
r_{27}	5.47	0.00	5.47	0.00	0.00	5.47	0.00	0.00	0.00
r_{28}	5.47	0.00	5.47	0.00	0.00	5.47	0.00	0.00	0.00
r_{29}	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
r_{30}	0.71	0.71	0.71	0.71	0.71	0.71	0.71	0.71	0.71
r_{31}	6.62	6.62	6.62	6.62	6.62	6.62	6.62	6.62	6.62
r_{32}	0.00	5.47	0.00	5.47	5.47	0.00	5.47	5.47	5.47
r_{33}	0.00	-5.47	0.00	-5.47	-5.47	0.00	-5.47	-5.47	-5.47
Y_C	0.26	0.26	0.26	0.26	0.26	0.26	0.26	0.26	0.26
Y_{ATP}	3.93	19.76	4.97	4.15	19.76	4.97	4.15	4.15	4.15
ratp	101.88	20.24	80.45	96.41	20.24	80.45	96.41	96.41	96.41

Table A6. Solutions from S45 to S54 for 54 solutions of the metabolic engineering example (cont.).

	S46	S47	S48	S49	S50	S51	S52	S53	S54
r_1	5.76	5.76	5.76	11.10	11.10	11.10	11.10	10.64	10.64
r_2	0.65	0.65	0.65	3.38	3.38	3.38	3.38	0.65	0.65
r_3	0.08	0.08	0.08	0.08	0.08	0.08	0.08	0.08	0.08
r_4	0.50	0.50	0.50	1.42	1.42	1.42	1.42	0.50	0.50
r_5	0.14	0.14	0.14	1.97	1.97	1.97	1.97	0.14	0.14
r_6	0.36	0.36	0.36	0.36	0.36	0.36	0.36	0.36	0.36
r_7	0.14	0.14	0.14	1.06	1.06	1.06	1.06	0.14	0.14
r_8	0.00	0.00	0.00	0.91	0.91	0.91	0.91	0.00	0.00
r_9	0.14	0.14	0.14	0.14	0.14	0.14	0.14	0.14	0.14
r_{10}	5.03	5.03	5.03	7.63	7.63	7.63	7.63	9.91	9.91
r_{11}	0.03	0.03	0.03	0.03	0.03	0.03	0.03	0.03	0.03
r_{12}	5.15	5.15	5.15	9.57	9.57	9.57	9.57	10.03	10.03
r_{13}	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05
r_{14}	10.24	10.24	10.24	20.00	20.00	20.00	20.00	20.00	20.00
r_{15}	0.60	0.60	0.60	0.60	0.60	0.60	0.60	0.60	0.60
r_{16}	9.64	9.64	9.64	19.40	19.40	19.40	19.40	19.40	19.40
r_{17}	0.29	0.29	0.29	0.29	0.29	0.29	0.29	0.29	0.29
r_{18}	2.45	2.45	2.45	6.87	6.87	6.87	6.87	7.33	7.33
r_{19}	0.00	0.00	0.00	0.81	15.23	0.81	15.23	6.28	9.76
r_{20}	1.13	1.13	1.13	1.13	1.13	1.13	1.13	1.13	1.13
r_{21}	7.08	7.08	7.08	16.03	1.60	16.03	1.60	10.55	7.08
r_{22}	1.17	1.17	1.17	1.17	1.17	1.17	1.17	1.17	1.17
r_{23}	0.00	0.00	0.00	14.42	0.00	14.42	0.00	3.48	0.00
r_{24}	5.90	5.90	5.90	0.43	0.43	0.43	0.43	5.90	5.90
r_{25}	5.90	5.90	5.90	0.43	0.43	0.43	0.43	5.90	5.90
r_{26}	0.43	0.43	0.43	0.43	0.43	0.43	0.43	0.43	0.43
r_{27}	5.47	5.47	5.47	0.00	0.00	0.00	0.00	5.47	5.47
r_{28}	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
r_{29}	5.47	5.47	5.47	0.00	0.00	0.00	0.00	5.47	5.47
r_{30}	0.71	0.71	0.71	0.71	0.71	0.71	0.71	0.71	0.71
r_{31}	1.15	1.15	1.15	1.15	1.15	1.15	1.15	1.15	1.15
r_{32}	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
r_{33}	5.47	5.47	5.47	0.00	0.00	0.00	0.00	5.47	5.47
Y_C	0.48	0.48	0.48	0.25	0.25	0.25	0.25	0.26	0.26
Y_{ATP}	5.25	5.25	5.25	3.87	12.83	3.87	12.83	3.87	4.66
ratp	76.16	76.16	76.16	103.30	31.19	103.30	31.19	103.30	85.92

REFERENCES

Akman, U., N. Okay and Ö. Hortaçsu, 2008, "Hierarchical Clustering Analysis for the Distribution of Origanum-Oil Components in Dense CO_2", *Korean Journal of Chemical. Engineering*, Vol. 25, No. 2, pp. 329-244.

Dantzig, G. B., 1963, *Linear Programming and Extensions*, Princeton University Press, Princeton, New Jersey.

Dantzig, G. B. and M. N. Thapa, 1997, *Linear Programming, 1: Introduction*, Springer, New York.

Edgar, T. F., D. M. Himmelblau and L. S. Lasdon, 2001, *Optimization of Chemical Processes*, 2nd ed., McGraw-Hill, New York.

Ferris, M. C., 2005, *MATLAB and GAMS: Interfacing Optimization and Visualization Software*, http://pages.cs.wisc.edu/~ferris/matlabgams.pdf, accessed at May 2012.

Ferris, M. C., J. Rishabh and S. Dirske, 2011, *GDXMRW: Interfacing GAMS and MATLAB*, http://www.gams.com/dd/docs/tools/gdxmrw.pdf, accessed at May 2012.

GAMS Development Corporation, 2012, *GAMS GDX Facilities and Tools*, http://www.gams.com/dd/docs/tools/gdxutils.pdf, accessed at May 2012.

Goel, A., M. M. Domach, W. Hanley, J. W. Lee and M. M. Ataai, 1996, "Coordination of Glycolysis and TCA Cycle Reaction Networks Citrate-Glucose Cometabolism Eliminates Acids and Reveals Potential Metabolic Engineering Strategies", *Annals of the New York Academy of Sciences*, Vol. 782, pp. 1-16.

Herrgard, M. J., S. S. Fong and B. Ø. Palsson, 2006, "Identification of Genome-Scale Metabolic Network Models Using Experimentally Measured Flux Profiles", *PLoS Computational Biology*, Vol. 2, No. 7, e72, pp. 676-686.

Ingraham, J. L., O. Maaloe and F. C. Neidhardt, 1983, *Growth of the Bacterial Cell*, Sinauer Associates, Sunderland, Massachusetts.

Lee, S., C. Phalakornkule, M. M. Domach and I. E. Grossmann, 2000, "Recursive MILP Model for Finding all the Alternate Optima in LP Models for Metabolic Networks", *Computers and Chemical Engineering*, Vol. 24, No. 2-7, pp. 711-716.

Mahadevan R. and C. H. Schilling, 2003, "The Effects of Alternate Optimal Solutions in Constraint-Based Genome-Scale Metabolic Models", *Metabolic Engineering*, Vol. 5, No. 4, pp. 264-276.

Mandelstam, J., K. McQuillen and I. Dawes, 1982, *Biochemistry of Bacterial Growth*, Halsted Press, New York.

Orth, J. D., I. Thiele and B. Ø. Palsson, 2010, "What is Flux Balance Analysis?", *Nature Biotechnology*, Vol. 28, No. 3, pp. 245-248.

Phalakornkule, C., S. Lee, T. Zhu, R. Koepsel, M. M. Ataai and I. E. Grossmann, and M. M. Domach, 2001, "A MILP-Based Flux Alternative Generation and NMR Experimental Design Strategy for Metabolic Engineering", *Metabolic Engineering*, Vol. 3, No. 2, pp. 124-137.

Reed, J. L. and B. Ø. Palsson, 2004, "Genome-Scale in Silico Models of E. coli Have Multiple Equivalent Phenotypic States: Assessment of correlated Reaction Subsets That Comprise Network States", *Genome Research*, Vol. 14, No. 9, pp.1797-1805.

Sahinidis, N. V., M. Tawarmalani and M. Yu, 2003, "Design of Alternative Refrigerants via Global Optimization", *AIChE Journal*, Vol. 49, No. 7, pp. 1761-1775.

Sahinidis, N. V. and M. Tawarmalani, 2010, *BARON 9.0.4: Global Optimization of Mixed-Integer Nonlinear Programs, User's Manual*, http://www.gams.com/dd/docs/solvers/baron.pdf, accessed at July 2012.

Sahinidis, N. V. and M. Tawarmalani, 2012, *BARON* http://www.gams.com/dd/docs/solvers/baron.pdf, accessed at July 2012.

Taha, H. A., 2007, *Modeling with Linear Programming*, Operations Reserach: An Intrıduction, 8th ed., Prentice Hall, New Jersey, pp. 11-80.

Tawarmalani, M. and N. V. Sahinidis, 2005, "A Polyhedral Branch-and-Cut Approach to Global Optimization", Vol. 103, No. 2, pp. 225-249.

Tsai, J. F., M. H. Lin and Y. C. Hu, 2008, "Finding Multiple Solutions to General Integer Linear Programs", *European Journal of Operational Research*, Vol. 184, No. 2, pp. 802-809.

Vo, T. D., H. J. Greenberg and B. Ø. Palsson, 2004, "Reconstruction and Functional Characterization of the Human Mitochondrial Metabolic Network Based on Proteomic and Biochemical Data", *The Journal of Biological Chemistry*, Vol. 279, No. 38, pp. 39532-39540.