1

PM-1 3½"x4" PHOTOGRAPHIC MICROCOPY TARGET
NBS 1010a ANSI/ISO #2 EQUIVALENT

National Library
of Canada

Acquisitions and
Bibliographic Services Branch

395 Wellington Street
Ottawa, Ontario
K1A 0N4

Bibliothèque nationale
du Canada

Direction des acquisitions et
des services bibliographiques

395, rue Wellington
Ottawa (Ontario)
K1A 0N4

Votre bibliothèque

Our library

NOTICE

The quality of this microform is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

Reproduction in full or in part of this microform is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30, and subsequent amendments.

AVIS

La qualité de cette microforme dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

La reproduction, même partielle, de cette microforme est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30, et ses amendements subséquents.

Canada

A Methodology for the Formulation and Solution of Global Optimization Problems

by

Erik Willy Dravnieks, B.Sc., M.L.I.S., M.Sc.

A thesis submitted to
the Faculty of Graduate Studies and Research
in partial fulfilment of
the requirements for the degree of
Doctor of Philosophy

Ottawa-Carleton Institute for Electrical Engineering
Faculty of Engineering
Department of Systems and Computer Engineering
Carleton University
Ottawa, Ontario, Canada

December 12, 1994

© Erik Willy Dravnieks, 1994

National Library
of Canada

Acquisitions and
Bibliographic Services Branch

395 Wellington Street
Ottawa, Ontario
K1A 0N4

Bibliothèque nationale
du Canada

Direction des acquisitions et
des services bibliographiques

395, rue Wellington
Ottawa (Ontario)
K1A 0N4

Your file *Votre référence*

Our file *Notre référence*

THE AUTHOR HAS GRANTED AN
IRREVOCABLE NON-EXCLUSIVE
LICENCE ALLOWING THE NATIONAL
LIBRARY OF CANADA TO
REPRODUCE, LOAN, DISTRIBUTE OR
SELL COPIES OF HIS/HER THESIS BY
ANY MEANS AND IN ANY FORM OR
FORMAT, MAKING THIS THESIS
AVAILABLE TO INTERESTED
PERSONS.

L'AUTEUR A ACCORDE UNE LICENCE
IRREVOCABLE ET NON EXCLUSIVE
PERMETTANT A LA BIBLIOTHEQUE
NATIONALE DU CANADA DE
REPRODUIRE, PRETER, DISTRIBUER
OU VENDRE DES COPIES DE SA
THESE DE QUELQUE MANIERE ET
SOUS QUELQUE FORME QUE CE SOIT
POUR METTRE DES EXEMPLAIRES DE
CETTE THESE A LA DISPOSITION DES
PERSONNE INTERESSEES.

THE AUTHOR RETAINS OWNERSHIP
OF THE COPYRIGHT IN HIS/HER
THESIS. NEITHER THE THESIS NOR
SUBSTANTIAL EXTRACTS FROM IT
MAY BE PRINTED OR OTHERWISE
REPRODUCED WITHOUT HIS/HER
PERMISSION.

L'AUTEUR CONSERVE LA PROPRIETE
DU DROIT D'AUTEUR QUI PROTEGE
SA THESE. NI LA THESE NI DES
EXTRAITS SUBSTANTIELS DE CELLE-
CI NE DOIVENT ETRE IMPRIMES OU
AUTREMENT REPRODUITS SANS SON
AUTORISATION.

ISBN 0-315-98473-2

Name Erin Willy Brown - k's

Dissertation Abstracts International is arranged by broad, general subject categories. Please select the one subject which most nearly describes the content of your dissertation. Enter the corresponding four-digit code in the spaces provided.

A Methodology for the Formulation and Solution of
Global Optimization Problems **SUBJECT TERM**

0796 **U·M·I**
SUBJECT CODE

Subject Categories

THE HUMANITIES AND SOCIAL SCIENCES

COMMUNICATIONS AND THE ARTS

Architecture 0729
Art History 0377
Cinema 0900
Dance 0378
Fine Arts 0357
Information Science 0723
Journalism 0391
Library Science 0399
Mass Communications 0708
Music 0413
Speech Communication 0459
Theater 0465

EDUCATION

General 0515
Administration 0514
Adult and Continuing 0516
Agricultural 0517
Art 0273
Bilingual and Multicultural 0282
Business 0688
Community College 0275
Curriculum and Instruction 0727
Early Childhood 0518
Elementary 0524
Finance 0277
Guidance and Counseling 0519
Health 0680
Higher 0745
History of 0520
Home Economics 0278
Industrial 0521
Language and Literature 0279
Mathematics 0280
Music 0522
Philosophy of 0998
Physical 0523

Psychology 0525
Reading 0535
Religious 0527
Sciences 0714
Secondary 0533
Social Sciences 0534
Sociology of 0340
Special 0529
Teacher Training 0530
Technology 0710
Tests and Measurements 0288
Vocational 0747

LANGUAGE, LITERATURE AND LINGUISTICS

Language
General 0679
Ancient 0289
Linguistics 0290
Modern 0291
Literature
General 0401
Classical 0294
Comparative 0295
Medieval 0297
Modern 0298
African 0316
American 0591
Asian 0305
Canadian (English) 0352
Canadian (French) 0355
English 0593
Germanic 0311
Latin American 0312
Middle Eastern 0315
Romance 0313
Slavic and East European 0314

PHILOSOPHY, RELIGION AND THEOLOGY

Philosophy 0422
Religion
General 0318
Biblical Studies 0321
Clergy 0319
History of 0320
Philosophy of 0322
Theology 0469

SOCIAL SCIENCES

American Studies 0323
Archaeology
Archaeology 0324
Cultural 0326
Physical 0327
Business Administration
General 0310
Accounting 0272
Banking 0770
Management 0454
Marketing 0338
Canadian Studies 0385
Economics
General 0501
Agricultural 0503
Commerce-Business 0505
Finance 0508
History 0509
Labor 0510
Theory 0511
Folklore 0358
Geography 0366
Gerontology 0351
History
General 0578

Art and Media 0579
Medieval 0581
Modern 0582
Black 0328
African 0331
Asia, Australia and Oceania 0332
Canadian 0334
European 0335
Latin American 0336
Middle Eastern 0333
United States 0337
History of Science 0585
Law 0398
Political Science
General 0615
International Law and Relations 0616
Public Administration 0617
Recreation 0814
Social Work 0452
Sociology
General 0626
Criminology and Penology 0627
Demography 0938
Ethnic and Racial Studies 0631
Individual and Family Studies 0628
Industrial and Labor Relations 0629
Public and Social Welfare 0630
Social Structure and Development 0700
Theory and Methods 0344
Transportation 0709
Urban and Regional Planning 0999
Women's Studies 0453

THE SCIENCES AND ENGINEERING

BIOLOGICAL SCIENCES

Agriculture
General 0473
Agronomy 0285
Animal Culture and Nutrition 0475
Animal Pathology 0476
Food Science and Technology 0359
Forestry and Wildlife 0478
Plant Culture 0479
Plant Pathology 0480
Plant Physiology 0817
Range Management 0777
Wood Technology 0746
Biology
General 0306
Anatomy 0287
Biostatistics 0308
Botany 0309
Cell 0379
Ecology 0329
Entomology 0353
Genetics 0369
Limnology 0793
Microbiology 0410
Molecular 0307
Neuroscience 0317
Oceanography 0416
Physiology 0433
Radiation 0821
Veterinary Science 0778
Zoology 0472
Biophysics
General 0786
Medical 0760

EARTH SCIENCES

Biogeochemistry 0425
Geochemistry 0996

Geodesy 0370
Geology 0372
Geophysics 0373
Hydrology 0388
Mineralogy 0411
Paleobotany 0345
Paleoecology 0426
Paleontology 0418
Paleozoology 0985
Palynology 0427
Physical Geography 0368
Physical Oceanography 0415

HEALTH AND ENVIRONMENTAL SCIENCES

Environmental Sciences 0768
Health Sciences
General 0566
Audiology 0500
Chemotherapy 0992
Dentistry 0567
Education 0350
Hospital Management 0769
Human Development 0758
Immunology 0982
Medicine and Surgery 0564
Mental Health 0347
Nursing 0569
Nutrition 0570
Obstetrics and Gynecology 0380
Occupational Health and Therapy 0354
Ophthalmology 0381
Pathology 0571
Pharmacology 0419
Pharmacy 0572
Physical Therapy 0382
Public Health 0573
Radiology 0574
Recreation 0575

Speech Pathology 0460
Toxicology 0383
Home Economics 0386

PHYSICAL SCIENCES

Pure Sciences
Chemistry
General 0485
Agricultural 0749
Analytical 0486
Biochemistry 0487
Inorganic 0488
Nuclear 0738
Organic 0490
Pharmaceutical 0491
Physical 0494
Polymer 0495
Radiation 0754
Mathematics 0405
Physics
General 0605
Acoustics 0986
Astronomy and Astrophysics 0606
Atmospheric Science 0608
Atomic 0748
Electronics and Electricity 0607
Elementary Particles and High Energy 0798
Fluid and Plasma 0759
Molecular 0609
Nuclear 0610
Optics 0752
Radiation 0756
Solid State 0611
Statistics 0463

Applied Sciences

Applied Mechanics 0346
Computer Science 0984

Engineering
General 0537
Aerospace 0538
Agricultural 0539
Automotive 0540
Biomedical 0541
Chemical 0542
Civil 0543
Electronics and Electrical 0544
Heat and Thermodynamics 0348
Hydraulic 0545
Industrial 0546
Marine 0547
Materials Science 0794
Mechanical 0548
Metallurgy 0743
Mining 0551
Nuclear 0552
Packaging 0549
Petroleum 0765
Sanitary and Municipal 0554
System Science 0790
Geotechnology 0428
Operations Research 0796
Plastics Technology 0795
Textile Technology 0994

PSYCHOLOGY

General 0621
Behavioral 0384
Clinical 0622
Developmental 0620
Experimental 0623
Industrial 0624
Personality 0625
Physiological 0969
Psychobiology 0349
Psychometrics 0632
Social 0451

The undersigned hereby recommend to the Faculty of
Graduate Studies and Research acceptance of the thesis,

**"A Methodology for the Formulation and Solution
of Global Optimization Problems"**

submitted by

Erik Willy Dravnieks, B.Sc., M.L.I.S., M.Sc.

in partial fulfillment of the requirements for the degree of Doctor of Philosophy

Professor S. A. Mahmoud
Chair, Department of Systems and Computer Engineering

Professor J. W. Chinneck
Thesis Co-Supervisor

Professor G. M. Karam
Thesis Co-Supervisor

Professor W. A. Lodwick
External Examiner

Abstract

The state of the art in global optimization is such that *a priori* assumptions on the differentiability, continuity, etc. of the functions involved are required. In addition, for the majority of these methods the problem is required to be feasible and bounded. Little, if any, formulation information is obtained if a global (or local) minimum is obtained, a premature termination of the solution process occurs, or if an infeasible or unbounded result occurs. This limits the usefulness of these methods from a modelling perspective.

A global optimization system (GOS) is defined which addresses the issues of formulation and usage of an automaton while allowing for a more general problem type (reduced *a priori* assumptions). A global optimization system based on a branch and bound framework (GOS-BB) is developed along with a prototype implementation. The GOS-BB allows for infeasible and unbounded problems, can be used without assumptions about differentiability, continuity, etc., and provides formulation information (stability factors, causal structures for infeasible and unbounded problems, etc.), thus providing useful information from a modelling perspective.

In addition, an adaptive bounding strategy is developed as well as improvements and extensions to existing bounding methods (for the range of a function) in order to improve upon the performance of the base branch and bound algorithm. Finally, a test set is developed which extends the known test sets to include infeasible and unbounded problems.

Acknowledgments

I express my gratitude to my mother, Ilse, for inspiring a neverending curiosity and to my father, Olgers, for inspiring pride in doing a job well and generating an interest in mathematics. I would also like to thank my brothers Olaf and Ivar, my grandmother Ilse, my uncle Walter and my aunts Franka, Edith, and Yvonne for their support and love.

I wish to thank my friends, Elizabeth, Simon, Suhayya, Ken, Nancy, André, Richard, Yayoi, Leon, Hazel, Joanne, Gareth, Kathy, Garth, Meghan, Jane, Paul, Nelly, Bruce, Adrienne, Darlene, Simon, Scott, Pat, Tony, Susan and Aaron for their friendship and support. I especially wish to thank Yugo, Tayseer, Robin, Rachel, David, Emma, Colin, and Zachary for their youthful exuberances, wet diapers and unique perspectives on life. As well, I wish to thank my fellow students, too many to name, with whom I have shared the trials and tribulations of academic studies.

I would like to thank my supervisors, John and Gerald, for their support, their 'ruthless' comments above and beyond the call of duty and for their often helpful suggestions. I would also like to thank my external Professor Weldon Lodwick and the rest of my defense committee for their comments and suggestions.

I wish to acknowledge the following sources of funding which I obtained during various portions of the research: Ontario Graduate Scholarships, the Systems and Computer Engineering Department at Carleton University, and Natural Sciences and Engineering Research Council of Canada Grants awarded to Professors J. W. Chinneck and G. M. Karam.

This thesis is dedicated to the late Dr. Percy Cuttle, for his love, kindness, understanding and support.

Contents

List of Figures	x
List of Tables	xi
List of Symbols	xiii
1 Introduction	1
1.0.1 Global Optimization Problems	1
1.0.2 Research Motivation	2
1.1 Research Objective	4
1.2 Overview of the Dissertation	5
2 Background	6
2.1 Introduction	6
2.2 Set Theory, Real and Interval Analysis	6
2.2.1 Set Theory and Real Analysis	6
2.3 Function Procedures and Codeable Functions	8
2.3.1 Interval Analysis	9
2.3.2 Extended Interval Arithmetic	12
2.4 Bounding Functions	13
2.4.1 Natural Interval Extension	14
2.4.2 Krawczyk's Centered Form	16
3 Background : Global Optimization and Automata	18
3.1 Optimization	18
3.2 Global Optimization	20

3.2.1	Possible Outcomes of a Global Optimization Problem	21
3.2.2	Local Optimization	23
3.2.3	Problem Test Sets	23
3.3	Automata and Global Optimization	24
3.3.1	FINITENESS	25
3.3.2	DILEMMA OF EQUALITY	27
3.3.3	FUNCTION REPRESENTATION	28
3.3.4	RESTRICTED FUNCTION DOMAINS	30
4	Research Problem	31
4.1	Introduction	31
4.2	Analysis of the State of the Art in Global Optimization Methods	31
4.2.1	Covering Methods and the Interval Newton's Method	33
4.2.2	Stochastic Methods	34
4.2.3	Penalty and Trajectory Methods	36
4.2.4	Global Boundary Methods	38
4.2.5	Conclusions	39
4.3	Towards a Global Optimization System	41
4.3.1	Introduction to Global Optimization Problems	41
4.3.2	Global Optimization and Automata	42
4.3.3	Formulation Aids for a Global Optimization System	45
4.3.4	Definition of a Global Optimization System	47
4.3.5	Testing a Global Optimization System	49
4.4	Problem Statement	51
5	A Practical Global Optimization Test Set	54
5.1	Introduction	54
5.2	Evaluation of Test Set	54
5.2.1	Problem Sizes	55
5.2.2	Conclusion	56
6	Improved Bounding Functions	59
6.1	Extensions to Functions with Logical Connectives	59
6.1.1	Bounding Functions with Logical Connectives	59

6.1.2	Krawczyk Centered Form	64
6.1.3	Example	65
6.1.4	Conclusion	67
6.2	Modified Krawczyk Centered Form	67
6.2.1	Evaluation	68
6.2.2	Conclusion	73
7	Branch and Bound Based Global Optimization System	74
7.1	Issues Arising from the Use of an Automaton	74
7.1.1	Finiteness	75
7.1.2	Dilemma of Equality	76
7.1.3	Function Representation	79
7.1.4	Restricted Function Domains	79
7.2	An Outline of GOS-BB	80
7.3	Feasibility Test for a Box	83
7.4	Application of a Bounding Function Strategy	85
7.5	Components of the Branch and Bound Framework	86
7.5.1	STEP 1: Initialize	87
7.5.2	STEP 2: Test Initial Box	88
7.5.3	STEP 3: Branch	89
7.5.4	STEP 4: Fathom	91
7.5.5	STEP 5: Update	91
7.6	Searching for an Incumbent Solution	92
7.7	Deletion Filter IISU ₀ Detection Algorithm	93
7.7.1	Evaluation	95
7.8	Unbounded Point Detection	96
7.8.1	Identification of Candidate Unbounded Domain Points	97
7.8.2	Verification of Unbounded Domain Points	99
7.8.3	Unbounded Point Detection Algorithm	101
7.8.4	Evaluation	102
7.9	Formulation Aids	103
7.9.1	Flatness of the Objective Function	103
7.9.2	Sensitivity Analysis	105

7.9.3	Snap Shot Evaluations	106
7.10	Indeterminate Outcomes	108
8	Branching and Bounding Strategies for GOS-BB	110
8.1	Bounding Functions	110
8.1.1	Basic Set	110
8.1.2	Evaluation	112
8.1.3	Adaptive Bounding Methodology	112
8.2	Subdivision Strategies	122
8.2.1	S1: Classic Division into Two Equal Parts	123
8.2.2	S2: Division by Narrow Band about a Local Minimum	123
8.2.3	S3: Division into Thirds, by Functional Value	123
8.2.4	S4: Division into Thirds, by Functional and Slope Values	124
8.2.5	Evaluation	124
9	An Integrated Global Optimization Formulation and Solution Methodology	133
9.1	GOPS: A Software Prototype for a GOS-BB	133
9.1.1	Set of Boxes	133
9.1.2	Bounding and Branching Methods	134
9.1.3	Stopping Conditions	134
9.1.4	Search for Incumbent Solution	135
9.1.5	Algorithms	135
9.1.6	Statistics Kept	135
9.1.7	Defaults	136
9.2	Evaluation	136
9.2.1	Infeasible Problems	136
9.2.2	Domain Unbounded Problems	139
9.2.3	Global Minimum Problems	141
9.2.4	Flatness Factors	150
9.2.5	Case Studies	151
9.2.6	Indeterminate Outcomes	156
9.2.7	Conclusions	156

10 Concluding Remarks	158
10.1 Dissertation Summary	158
10.2 Contributions	159
10.3 Future Research	160
A State of the Art in Global Optimization Methods	165
A.1 Background	166
A.2 Global Boundary Optimization	167
A.2.1 Outer Approximation Method	169
A.2.2 The Relief Indicator Method	169
A.2.3 Discussion	170
A.3 Covering Methods	170
A.3.1 A Branch and Bound Prototype	171
A.3.2 Covering Method Algorithms	175
A.3.3 Discussion	177
A.4 Methods of Generalized Descent	177
A.4.1 Trajectory Methods	178
A.4.2 Penalty Methods	178
A.4.3 Interval Newton Based Methods	179
A.4.4 Discussion	182
A.5 Stochastic Methods	183
A.5.1 Pure Random Search	184
A.5.2 Multistart	184
A.5.3 Clustering Methods	185
A.5.4 Simulated Annealing	188
A.5.5 Genetic Algorithms	189
A.5.6 Tabu Search	189
A.5.7 Discussion	189
A.6 Conclusions	191
B Test Set	192
B.1 Problems from Existing Test Sets	192
B.1.1 Problems from Floudas and Pardalos	192

B.1.2	Problems from Schwefel	198
B.1.3	Problems from other Sources	199
B.2	New Test Set	201
B.2.1	Unbounded Problems	201
B.2.2	Infeasible Problems	201
B.2.3	Feasible and Bounded Test Set	206
B.2.4	Constraint Sets	212
C	GOS-BB : An Algorithm for a Global Optimization System	219
C.1	Procedures for GOS-BB	223
C.1.1	Determine Feasibility of a Box, B	223
C.1.2	Application of a Bounding Function Strategy, BFS , over A Box, B	224
C.1.3	Deletion Filter IISU ₀ Algorithm	226
C.1.4	Unbounded Point Detection Algorithm	227
D	Interval Branch and Bound Line Search Method	228
D.1	Example	229
D.2	Discussion	231
	Bibliography	233

List of Figures

6.1	domain components of $f(x)$ for intervals A and B	62
7.1	Constraint Set 6	96
7.2	$f(x) = \frac{-1}{(x_1-2)^2+(x_2-4)^2}$	103

List of Tables

5.1	Problem Test Set : Sizes and Outcomes : INFEASIBLE	56
5.2	Problem Test Set : Sizes and Outcomes : DOMAIN UNBOUNDED	56
5.3	Problem Test Set : Sizes and Outcomes : GLOBAL MINIMUM	57
5.4	Sets of Constraints : Size and Feasibility	58
6.1	Bound test of Modified vs Unmodified Krawczyk Centered Forms : 6.2.1-1 .	70
6.2	Bound test of Modified vs Unmodified Krawczyk Centered Forms : 6.2.1-2 .	71
6.3	Performance test of Modified vs Unmodified Krawczyk Centered Forms: . . .	72
8.1	Bound Comparissons for the 4 basic bounding set	113
8.2	Validity tests for ABA	119
8.3	<i>BBF</i> 's and <i>cbs</i> 's for ABA	120
8.4	Performance tests for ABA	121
8.5	Performance tests for ABA	122
8.6	Subdivision tests for normal GOS-BB operation	126
8.7	Subdivision tests for normal GOS-BB operation	127
8.8	Subdivision tests for normal GOS-BB operation	128
8.9	Subdivision tests for Deletion Filtering operation	130
8.10	Subdivision tests for Deletion Filtering operation	131
9.1	INFEASIBLE problem tests	137
9.2	DOMAIN UNBOUNDED problem tests	140
9.3	GLOBAL MINIMUM problem tests	142
9.4	Flatness Factors	150
9.5	Melville Shipping Ltd. Problem - global minimum	155
9.6	Melville Shipping Ltd. Problem - alternate global minima	155

9.7 Melville Shipping Ltd. Problem - Flatness Factors 155

List of Symbols

$\Leftrightarrow$	if and only if
R, R^1	The 1-dimensional Euclidean space.
R^n	The n-dimensional Euclidean space.
z_i	The i^{th} element of the n-dimensional object, z
$N(x_i, \epsilon)$	An ϵ -neighbourhood of x_i , that is, an open ball about x_i of radius $\epsilon > 0$.
$f, f(x)$	$f : D \subseteq R^n \rightarrow R$ - a real valued function from a domain, D , to R - Note: We denote real valued functions by lower case letters.
$f \in C^p$	Denotes that the function, f , has continuous partial derivatives of order p .
∇f	The gradient of the function, f .
I, I^1	$\{t \mid a_1 \leq t \leq a_2, a_1, a_2 \in R\}$ - The 1-dimensional space of intervals.
I^n	The n-dimensional space of intervals.
$lb(A), ub(A)$	The lower and upper bound of the interval $A \in I^n$, sometimes called the lower and upper points as $lb(A), ub(A) \in R^n$.
$w(A)$	$\max\{w(A_i) \mid i = 1, \dots, n\}$ - The width of the interval $A \in I^n$.
$m(A)$	$(m(A_1), \dots, m(A_n))$ - The midpoint of the interval $A \in I^n$.
$\circ$	Denotes one of the set of arithmetic operators, $\{+, -, \times, \div\}$.
$I(D)$	The set of all intervals in the set $D \subseteq R^n$

$F, F(X)$	$F : I(D) \rightarrow I$ - an interval function from an interval domain, $I(D)$, to I - Note: we denote interval functions by upper case letters.
$\square f(X)$	$\{f(x) \mid f : D \in R^n \rightarrow R\} \ X \ x \in X$ - The range of the real valued function, f , over X .
$F^I(X)$	Denotes an inclusion function for $f(x)$ over the interval, X .
$F_{NIE}(X)$	The natural interval extension of f over $X \in I(D)$.
lbN_Θ	The smallest machine representable number for the automaton Θ .
ubN_Θ	The largest machine representable number for the automaton Θ .
∇x	An operator which returns the largest machine representable number $\leq x$, $x \leq ubN_\Theta$.
Δx	An operator which returns the smallest machine representable number $\geq x$, $x \geq lbN_\Theta$.
$M_{\nabla, \Delta}(x)$	An operator which returns the smallest machine representable n -dimensional interval which includes x , $\{lbN_\Theta \leq x \leq ubN_\Theta\}$.
$\triangleleft x$	An operator which returns the next smallest machine representable number from x .
$\triangleright x$	An operator which returns the next largest machine representable number from x .
$F_K(X)$	Denotes the Krawczyk centered form for the function, f , over the interval X .
C	The set of all inequality and equality constraints.
VB	The set of all variable bounds. A variable bound is a constraint of the form $x_i \leq a$ or $x_i \geq a$ for $a \in R$
$C \cup S$	The set $C \cup VB$.
$card(A)$	The cardinality of the set A .

Chapter 1

Introduction

This dissertation's goal is to develop a *global optimization system*, a methodology which aids in the formulation and solution of global optimization problems within a modelling environment. By *global optimization* we mean the process of finding a globally minimum value when many local minima may exist (we note that any maximization problem can be converted into a minimization problem). A *global optimization problem* is a formal mathematical definition, or model, of what is to be optimized and under what conditions. By *modelling* we mean the process of using a mathematical model to represent a system. There are numerous classes of mathematical models, including linear programming, combinatorial optimization problems, non-linear programs, etc.

1.0.1 Global Optimization Problems

Global optimization problems consist of a real valued objective function as well as a set of constraints (see Section 2.2). A point that satisfies all of the constraints is called a *feasible point* and the set of all feasible points is called the *feasible region*. In general, the global optimization problem will have many local minima as well as a set of non-contiguous irregularly shaped feasible regions.

The desired solution of a global optimization problem is a global minimum, a feasible point for which no other feasible point has a smaller functional value. However, it may well be the case that a finite global minimum does not exist. For example, the feasible region may be empty (an infeasible problem) or the function may be unbounded for some point of the feasible region. We define an *outcome* as the type of solution that is obtained by the solution

process. There are three classes of outcomes: infeasible, unbounded, or feasible & bounded (a finite global minimum exists). The *incumbent solution* is the best feasible solution found so far at any point in the solution process.

We note that the only means by which a candidate global minimum for general global optimization problems can be validated is if the feasible region is, explicitly or implicitly, exhaustively searched. Furthermore, the complexity of a general global optimization problem as well as of the solution process necessitates the use of an automaton for problems of a practical size. This introduces errors (possibly logical in nature, that is, an error in a decision, as well as numerical) due to the nature of the automaton and necessitates that the global optimization problem be solved in the finite space of the automaton.

Global optimization problems can arise either from a 'practical' source, namely from problems in the areas of economics, engineering, etc. [22] or from a 'theoretical' source, namely from problems in areas such as mathematics, etc. We are interested in global optimization problems that arise from a practical source.

For example, economics deals with the allocation of limited resources (modelled by the set of constraints) so as to optimize some process or system modelled by the objective function. Engineering or design problems are related in that a set of possible options are available for the design of some object. Each possible configuration will provide an object of some relative value, determined by the objective function. When some or all of the variables of choice are continuous in nature, the mathematical model of the choices is a global optimization problem. Constraints are used to specify the types of configurations that are to be examined.

1.0.2 Research Motivation

For the modeller, the process of determining a solution to a problem can be generalized as follows:

- 1 · Design a mathematical model of the system to be optimized.
- 2 · Obtain an outcome to the mathematical model by the use of an optimization system.
- 3 · Evaluate the outcome in relation to the system to be optimized and determine if the model needs to be modified. If so, go to 1. Otherwise Stop.

The modeller not only needs to obtain an outcome to the model, but also needs to evaluate the outcome for its correctness and appropriateness relative to the system being

optimized. We now examine features which would be desirable in an optimization system from a modelling perspective.

In general, the characteristics of the functions used (such as the continuity, differentiability, and existence of Lipschitz constants, etc.) as well as the characteristics of the problem (feasibility and boundedness, etc.) may not be known *a priori*. Thus the optimization system must be able to solve problems for which little information is known *a priori* about the function and/or problem characteristics.

The optimization system must be able to validly identify which of the outcomes has occurred, despite errors introduced by the use of the automaton, as well as provide information on the accuracy of the incumbent solution if the outcome is feasible and bounded. A measure of the accuracy of the incumbent solution can be used to indicate if the solution process was sufficient or if further work is required. If the outcome is infeasible (or unbounded), then information on the cause of an infeasibility (or unboundedness) can be used to determine if the model is infeasible (unbounded) due to a syntactic error, a modelling error, or due to the process or system being modelled, and thus address both the correctness and appropriateness of the outcome.

In most cases, the model parameters (the constants and coefficients of the functions used in the model) are known to vary by a certain amount. Thus a measure of the sensitivity of the incumbent solution to small perturbations can be of use in determining the appropriateness of the solution.

We say an optimization system is *valid*, if it is guaranteed to provide correct outcomes. Furthermore, we say it provides *causal information* if the output can be used to analyze an infeasible or unbounded outcome.

How does the state of the art in global optimization fare against these requirements (see Section 4.2 and Appendix A for more detailed examinations)? The state of the art is a collection of methods which solve various specific problem types, require various restrictions on the set of functions, and assume a feasible and bounded problem. We note that it is in general difficult to ascertain whether the problem actually satisfies the various restrictions commonly used, such as differentiability, continuity, etc.

State of the art methods concentrate on feasible and bounded problems, reasoning that feasibility can be determined by a transformation into a problem of equivalent difficulty and that unboundedness can be determined if a very large negative number is returned as the

solution. While essentially correct, this rationale requires that the user must have a certain level of expertise in global optimization theory and practice. The state of the art in global optimization does not meet the modelling requirements, so it is no wonder that “we have been highly unsuccessful at convincing practitioners to use nonlinear [global optimization] models and sophisticated software” to quote A. R. Conn [15].

The motivation for the research in this dissertation is to provide a ‘useful’ global optimization system, by providing one that addresses the needs of modellers.

1.1 Research Objective

We define a *global optimization system* as a methodology for the formulation and solution of global optimization problems such that the following conditions are fulfilled:

- The system requires little, if any, *a priori* information on the characteristics of the function (such as differentiability, etc.) or the problem (such as feasibility, etc.) to be solved.
- The system can validly determine which of the possible outcomes (infeasible, unbounded, or feasible and bounded) has occurred.
- The system accounts for the errors introduced by the use of the automaton.
- The system provides a measure of the accuracy of the incumbent solution if the problem is feasible and bounded.
- The system provides some formulation assistance, such as causal information for infeasible and unbounded outcomes, a measure of the objective function ‘flatness’, sensitivity analysis, etc.
- The system can be implemented on an automaton such that it completes its task within a ‘reasonable’ amount of time and does not exceed the space limitations imposed by the automaton.

The objective of the dissertation is to develop such a global optimization system.

1.2 Overview of the Dissertation

The dissertation is divided into 10 chapters, of which the first provides an overall introduction to global optimization and the need for a global optimization system as described above. Chapter 2 provides a brief mathematical background in the areas of real analysis, interval arithmetic as well as background on bounding functions, methods by which the range of a function may be bounded over some domain. Chapter 3 provides an introduction to Global Optimization as well as an examination of the effect of using an automaton to solve global optimization problems.

Chapter 4 formally defines the problem addressed by the dissertation. Chapter 5 examines a set of infeasible and unbounded test problems, developed to enhance the existing feasible and bounded test sets and thus to allow us to test a global optimization system.

Chapter 6 examines improvements to existing bounding functions. Chapter 7 examines a branch and bound based global optimization system. Chapter 8 examines branching and bounding strategies that can be used by the global optimization system described in Chapter 7 and develops an adaptive bounding methodology.

Chapter 9 provides the results of empirical testing of the global optimization system developed in the dissertation and finally, Chapter 10 provides a summary of the results of the dissertation and the contributions made in the dissertation and closes with a brief description of ideas for future research.

.

.

Chapter 2

Background

2.1 Introduction

In this chapter basic background information on set theory, real analysis, codeable functions, interval analysis and bounding functions is presented. A summary of the symbols used is given in the List of Symbols (located before Chapter 1).

2.2 Set Theory, Real and Interval Analysis

2.2.1 Set Theory and Real Analysis

Let R^1 or R denote the set of real numbers and R^n the set of n -dimensional real vectors. By $\|x\|$ we mean the Euclidean norm of the vector x . That is, $\|x\| = (x^T x)^{1/2}$.

A sequence of vectors of R^n , $x_1, x_2, \dots, x_k, \dots$ (usually denoted by $\{x_k\}$ or $\{x_k\}_{k=1}^\infty$) is said to converge to the limit x if $\|x_k - x\| \rightarrow 0$ as $k \rightarrow \infty$. If $\{x_k\}$ converges to x , we use $x_k \rightarrow x$ to denote this. A point x_l is a limit point of the sequence $\{x_k\}$ if there exists a subsequence of $\{x_k\}$ that converges to x_l .

An open ball is an n -dimensional sphere (or n -sphere) around a point $x \in R^n$ and consists of the set of points of the form $\{y \in R^n \mid \|y - x\| < \epsilon\}$ for some $\epsilon > 0$. An open ball about a point x_0 with a radius ϵ is also called an ϵ -neighbourhood of x_0 , sometimes denoted as $N(x_0, \epsilon)$. A neighbourhood of a point x_0 is an open set that contains the point x_0 .

A set $S \subset R^n$ is called open if around every point of S there is an open ball that is contained in S . A set $S \subset R^n$ is closed if its relative complement $R^n \setminus S$ (the set of points in

R^n not in S) is open. A set is bounded if it can be contained within an n -sphere of finite radius. A set is compact if it is both closed and bounded.

A set $C \subset R^n$ is convex if for every $x_1, x_2 \in C$ the point $\alpha x_1 + (1 - \alpha)x_2$ is also in C for every $\alpha \in R, 0 \leq \alpha \leq 1$.

Functions

A real valued function is a function $f : D \subseteq R^n \rightarrow R$ where D is the domain of the function for which every $x \in D, f(x) \in R$ (that is, the function is finite at every point of D). We will also use $f(x) = f(x_1, x_2, \dots, x_n)$ to denote the above function.

A real valued function defined on an open set D_o is bounded from below if for all $x_0 \in cl(D_o)$ there exists a $M \in R$ and a $\delta(M) > 0$ such that $f(x) \geq M$ for all $x \in D_o \cap N(x_0, \delta(M)), x \neq x_0$. It is unbounded from below if the above is not true.

A function f defined on a convex set C is convex if for every $x_1, x_2 \in C$ and every $\alpha \in R, 0 \leq \alpha \leq 1$ $f(\alpha x_1 + (1 - \alpha)x_2) \leq \alpha f(x_1) + (1 - \alpha)f(x_2)$. It is concave if for every $x_1, x_2 \in C$ and every $\alpha \in R, 0 \leq \alpha \leq 1$ $f(\alpha x_1 + (1 - \alpha)x_2) \geq \alpha f(x_1) + (1 - \alpha)f(x_2)$.

A real valued function $f : D \rightarrow R$ is continuous at a point $x \in D \subset R^n$ if $x_k \rightarrow x$ implies $f(x_k) \rightarrow f(x)$. If f is continuous at every point of D , then f is called continuous and we denote it by $f \in \mathcal{C}$.

Constraints

A constraint C in R^n is a partition of R^n into two disjoint sets defined by some relation. Those points of R^n that satisfy the relation are the elements of one of these sets, while the remaining points are the elements of the other set. If REL denotes one of the elements of the set $\{=, \geq, \leq\}$, then the class of constraints defined by $g(x) REL 0$, where $g : D \subseteq R^n \rightarrow R$ is a real valued function and $x \in D$, are of special interest. A point, x' , satisfies a constraint if $g(x') REL 0$ is true.

The constraints of the form $g(x) \leq 0$ and $g(x) \geq 0$ are labelled inequality constraints while those of the form $g(x) = 0$ are labelled equality constraints. We need only consider inequality constraints of the form $g(x) \leq 0$ as any constraint of the form $g(x) \geq 0$ is equivalent to $-g(x) \leq 0$.

Equality constraints define a region (or set of points) of dimensionality less than the space in which they reside. For example, the equality constraint $x_1^2 + x_2^2 + x_3^2 - 4 = 0$ defines

the surface of a three dimensional sphere of radius 2 in R^3 . The region resides in R^3 but it is of dimension 2. Furthermore, R^3 is the minimal real space in which it can reside.

A constraint is said to be relaxed if by the introduction of additional variables (or constants) to the constraint, the set of points that satisfy the constraint has been increased. For example, the constraint, $g(x) \leq 0$ is relaxed if (for an $\alpha > 0 \in R$) the constraint is rewritten as $g(x) - \alpha \leq 0$.

Constraints of the form $a \leq x_i$ or $x_i \leq b$, for $a, b \in R$ are called variable bounds. We use the set C to denote the set of all inequality and equality constraints (exclusive of variable bounds) and VB to denote the set of all variable bounds.

2.3 Function Procedures and Codeable Functions

We now examine a system for representing real valued functions as a sequence of functions. The sequence of functions will be called a *function procedure*, and a *codeable function* (also referred to as a *factorable function*) is real valued function that is the last function in a function procedure [14, 65, 72, 73, 76, 91, 96].

The function procedure, S , for a codeable function represented by f_s , is a finite sequence of functions, $\{f_1, \dots, f_s\}$ defined as follows:

- The set of operators are the arithmetic operators, $\{+, -, \times, \div\}$ as well as a set of *a priori* defined unary functions, T .
- The set of objects are a finite set of constants, b_i , and a finite set of variables, x_j .
- The allowable sequences for a function of n variables and t constants are of the form :

$$* f_i := x_i \text{ for } i = 1, \dots, n$$

$$* f_i := f_j \circ f_k \text{ for some } j, k < i, \circ \in \{+, -, \times, \div\} \text{ and for } i = n + t + 1, \dots, s$$

$$* f_i(x) = T(f_j(x)) \text{ for } n + t < j < i$$

where any constants that are required are integrated into the set T , for example, the function $a - x_i$, where a is a constant would be a function that is an element of the set T . For the definitions of codeable functions currently in use, the set T of library functions is usually restricted such that each library function is differentiable.

Discussion

Since the set of library functions may in general include functions that require operators that are not arithmetic operators (for example they may contain logical connectives), the set of codeable functions may contain functions that are not rational functions.

The restriction to a differentiable function set, T , is usually required so that a system of automatic differentiation could be developed. However, this restricts the set of library functions and thus the set of functions that are representable by the set of codeable functions.

The concept of a codeable function can be extended so that library functions with more than one argument are possible [62]. Thus functions such as $\min(.,.)$, $\max(.,.)$, etc. can be included into this class of functions. A second extension is to include piecewise codeable functions [62]. Piecewise codeable functions allow domain components, for which the restriction of the function over each component is a codeable function and the union of the domain components contains the complete domain. With these extensions along with the removal of the implicit restriction of differentiability of a library function a very general class of representable functions can be defined. We define this class of functions as *extended codeable functions*.

2.3.1 Interval Analysis

Interval analysis was introduced by Moore [81] in 1966. The main motivation was to eliminate (or at least to account for) the errors generated in a numerical based calculation. What is achieved is a method which allows one to bound the set of correct answers in a numerical calculation.

Interval analysis is a generalization of real analysis. An interval is an ordered set (usually of elements from the set of reals). Each of these intervals becomes in turn an element of the n -dimensional interval, just as a one dimensional variable is expanded to n -dimensions. The resulting n -dimensional interval, or interval vector, can be thought of as a right parallelepiped parallel to the axis in the n -dimensional space of the reals, which for succinctness is sometimes called a *box* or *interval*. As real analysis is concerned with transformations of real vectors onto real vectors, interval analysis is concerned with the transformation of interval vectors. An important subset of interval analysis is interval arithmetic, which is concerned with the interval operations of addition, subtraction, multiplication and division.

The least element of an interval, A , is labelled the *lower bound*, $lb(A)$ while the greatest

element is labelled the *upper bound*, $ub(A)$. A *point interval* is an interval with only one element, that is $lb(A) = ub(A)$. Let us label the set of intervals produced by interval analysis for a numerical calculation the *resultant interval*.

The proper implementation of interval analysis guarantees the *correctness* of the resultant interval. Correctness is the property that the set of correct values are contained within the resultant interval. The resultant interval is in general not exact. Thus one of the goals of interval analysis research is to develop efficient methods by which the bounds can be made sharp or exact (within a specified tolerance).

Within the next section the basic definitions, theorems and concepts of interval analysis relative to global optimization are developed. The topics covered have been taken from the following frequently referenced sources; [3, 84, 87, 96, 97].

Basic Principles

An interval A is a subset of R of the form

$$A = [a_1, a_2] = [lb(A), ub(A)] = \{t \mid a_1 \leq t \leq a_2, a_1, a_2 \in R\}.$$

The set of all such intervals is denoted by I . We note that each interval is compact.

By definition, two intervals are equal if they are equal in the set theoretic sense. Thus $A = B \Leftrightarrow a_1 = b_1$ and $a_2 = b_2$. As well, $A \leq B \Leftrightarrow a_2 \leq b_1$.

In addition we define the following :

- The *width* of an interval A is denoted by $w(A) = a_2 - a_1$.
- The *midpoint* of an interval A is denoted by $m(A) = \frac{a_1 + a_2}{2}$.
- We define the *lower point* and *upper point* of an interval A as $lb(A)$ and $ub(A)$ respectively. This notation is used to differentiate the multiple uses of the terminology 'lower bound', etc.
- Let $\circ$ denote an element of $\{+, -, \times, \div\}$.

We begin by extending the normal arithmetic operations to interval arithmetic operations. We define the interval arithmetic operations by the following expression :

$$A \circ B = \{a \circ b \mid a \in A, b \in B, 0 \notin B \text{ for } \circ = \div\} \text{ for } A, B \in I$$

This definition ensures correctness, as it follows that for any $\alpha \in A, \beta \in B$ that $\alpha \circ \beta \in A \circ B$ (if $0 \notin B$). Moore [80] proved that $A \circ B \in I$ if $0 \notin B$. In addition, Moore [80] also proved that the above definition is equivalent to the following more practical rules for computation.

1. $[a_1, a_2] + [b_1, b_2] = [a_1 + b_1, a_2 + b_2]$
2. $[a_1, a_2] - [b_1, b_2] = [a_1 - b_2, a_2 - b_1]$
3. $[a_1, a_2] \times [b_1, b_2] = [\min(a_1b_1, a_1b_2, a_2b_1, a_2b_2), \max(a_1b_1, a_1b_2, a_2b_1, a_2b_2)]$
4. $[a_1, a_2] \div [b_1, b_2] = [a_1, a_2] \times [1/b_2, 1/b_1]$ if $0 \notin [b_1, b_2]$

These operations are labelled as the *interval arithmetic operations*. It is easy to see that these reduce to the natural arithmetic operators when point intervals are used.

A rough comparison between interval arithmetic and real arithmetic shows that they are similar in the following features:

- Commutative and associative for addition and multiplication.
- Division by zero is undefined and therefore not allowed.

and they are different in the following features:

- Subtraction (division) is not the inverse operation of addition (multiplication) for Interval arithmetic.
- While real arithmetic is distributive with respect to addition and multiplication, interval arithmetic is only sub-distributive. That is, for three intervals I, J, K one has $I * (J + K) \subseteq I * J + I * K$ but not necessarily $I * (J + K) \supseteq I * J + I * K$.

The next step is to extend the discussion to n-dimensional interval space. We do so component-wise. Thus an n-dimensional interval $A = (A_1, A_2, \dots, A_n)$ denotes an element of I^n with each $A_i \in I$. Each such interval defines a *box* in R^n .

Similar extensions to the definitions of the width, midpoint, etc. are summarized in the List of Symbols.

We now develop our notation for interval functions and for the range of a function.

Let $D \subseteq R^n$. Let $I(D)$ denote the set of all intervals in the set D . Let $F(X)$ denote an interval function, or $F : I(D) \rightarrow I$.

Let $D \subseteq R^n$, $f : D \rightarrow R$ be some function, then we let $\square f(X) = \{f(x) \mid x \in X\}$ be the range of the function f over $X \in I^n$. The range of a function is just the set of all possible values that the function has over the set X .

An interval function $F : I(D) \rightarrow I$ is called an *inclusion function* for f if $\square f(X) \subseteq F(X) \forall X \in I^n$. We denote this inclusion function as $F^I(X)$. There are two main classes on interval inclusions, the natural interval extension and centered forms.

By definition, $F(X)$ is *quadratic convergent* if $w(F(X)) - w(\square f(X)) = O(w(X)^2)$. That is, the inclusion can be improved quadratically if the domain of X is shortened.

Inclusion functions bound the range of the function while satisfying the requirement for correctness on every interval. As such, we will usually refer to them as *bounding functions* when they are used to obtain a bound on the range of a given function in the context of solving an optimization problem. In general, exact bounds are not realizable, due mainly because each occurrence of a variable in a function expression is treated as an independent variable. Thus when a variable occurs more than once in an expression, information is lost and the resultant interval will in general expand beyond the necessary limits of correctness.

Let $D \subseteq I^n$ and let $F : I(D) \rightarrow I$ be an interval function. F is *inclusion isotone* if $X \subseteq Y$ implies $F(X) \subseteq F(Y)$ for all $X, Y \in I(D)$.

The concept of inclusion isotonicity is important in actual computations as an interval may need to be rounded to a larger interval in order to be representable on an automaton. In addition it is this property that allows one to define the natural interval extension.

For example, it follows directly from the definition of the interval arithmetic operators that the interval arithmetic operators are inclusion isotonic. That is, $A \subseteq B, C \subseteq D \Rightarrow A \circ C \subseteq B \circ D$ (if $B \circ D$ is defined) (see [3] for a proof).

2.3.2 Extended Interval Arithmetic

Extended interval arithmetic is an extension of interval arithmetic to include unbounded intervals, thus it is sometimes referred to as infinite interval arithmetic. The basic idea is to extend the set of possible intervals from R to $R \cup \{-\infty, \infty\}$. In this way, intervals such as $[-\infty, 0]$ can be defined [38, 97]. Thus an arithmetic for infinite intervals can be developed which can be used to define a division for non-point intervals (contain at least two distinct

points) that contain zero. Division by a point interval equal to zero is still undefined.

When implemented on an automaton, we simply let the smallest and largest representable numbers become synonymous with $-\infty$ and ∞ respectively. Thus an interval of the form $[lbN_\Theta, 0]$ represents the interval $(-\infty, 0]$, $[ubN_\Theta, ubN_\Theta]$ represents the interval $[ubN_\Theta, \infty)$, etc. The notation $\langle a, b \rangle$ is used to indicate infinite or finite intervals depending on the values of a and b . Thus lbN_Θ and ubN_Θ are used to represent both infinite and finite intervals (an interval is finite if its width is finite).

The normal rules for infinite arithmetic is used to determine how the arithmetic operators are to be defined for infinite and finite intervals. Examples of the normal rules for infinite arithmetic are: $\pm\infty \pm \infty = \pm\infty$, $\infty \pm a = \pm\infty$ if $a \in R$, $0(\pm\infty) = 0$, $a/\pm\infty = 0$ if $a \in R$, etc.

Examples of the rules for the arithmetic operators are:

- $\frac{1}{\langle c, d \rangle} = (-\infty, 1/c] \cup [1/d, \infty)$ if $c < 0 < d$.
- $\langle a, b \rangle - \langle c, d \rangle = \langle a - d, b - c \rangle$.
- $\frac{1}{\langle c, 0 \rangle} = (-\infty, 1/c]$ if $c \neq 0$.
- $\frac{\langle a, b \rangle}{0}$ is undefined.

A more detailed examination of the rules as well as of the theory involved can be found in [97].

2.4 Bounding Functions

In this section we examine two inclusion functions, the natural interval extension and Krawczyk's centered form. These functions provide lower and/or upper bounds for the range of the functional values over a specified domain. There are other inclusion functions as well as other non-interval analysis based methods by which bounds on the range of a function can be determined.

Specifically, three classes of methods exist for finding a lower bound for the range of a function over a specified domain. We call such methods *bounding functions*.

The first class are inclusion functions and are usually based upon using a finite series expansion about a point of the function to approximate the function and then extending the

resultant series to interval analysis such that the range of the function over some box (or interval) is guaranteed to be within the output resultant interval. The theory of inclusion functions as well as various methods are discussed in [3, 6, 66, 81, 84, 87, 93, 94, 95, 96, 97, 98, 104, 105, 106].

The second class makes use of convex underestimating functions [55, 61, 73] and are based upon the creation of an convex function which is guaranteed to underestimate the function. That is, it bounds the function from below. By minimizing the convex function a lower bound for the function can be achieved. Similarly, a concave overestimating function can be used to determine an upper bound.

The third makes use of Lipschitz constants [55, 74], which can then be used to calculate a bound for the function over some hyper-sphere of the domain.

Methods which determine a bound for the range of a function can be thought of as naive global optimization methods in that if the bounds are exact, then the functional value of the global optima have been found. However, these methods fail to be effective as global optimization methods for two reasons. The first is that it is highly unlikely that the bounds produced are exact. The second is that only bounds are produced by these methods and thus it is difficult to determine the domain element which has the functional value equal to the global optima. In addition, it is difficult to determine if the bounds produced are exact. Thus these methods are not applicable as optimization methods in themselves, but are useful when used within an optimization methods.

2.4.1 Natural Interval Extension

The natural interval extension was introduced by Moore [81] as a means of approximating the range of a function. Consider the following theorem of Moore's, which has been called the fundamental theorem of interval analysis.

Let $f(x_1, x_2, \dots, x_n)$ be a rational function of n variables. Consider any sequence of arithmetic steps that serve to evaluate f with given arguments $x_1, x_2, \dots, x_n$. Suppose we replace the arguments x_i by corresponding intervals X_i for all $i = 1, \dots, n$ and replace the arithmetic steps in the sequence used to evaluate f by the corresponding interval arithmetic steps. The result will be an interval function $F(X_1, X_2, \dots, X_n)$ which satisfies $f(x_1, x_2, \dots, x_n) \in F(X_1, X_2, \dots, X_n) \quad \forall x_i \in X_i \quad \forall i$.

There exists a class of functions for which inclusion functions are already known. We

label them as the set, g . They include the well known functions such as $\sin(x)$, $\cos(x)$, e^x , etc. Neumaier [87] gives a listing of such elementary functions as well as the criteria required for a function to be called 'elementary' in this context. For this class of functions there are a corresponding set of pre-defined inclusion functions, G . In fact, in general, $G(X) = \Box g(x)$ will hold due to the nature of our knowledge of these functions.

Thus by the application of the appropriate replacement scheme one can generate an inclusion function for a codeable function. We now extend the fundamental theorem as follows:

Let $f : D \subseteq R^n \rightarrow R$ be a function, denoted by $f(x)$, which can be described as an explicit expression without the use of logical or conditional statements (for example, if-then, while, etc.), such that the expression is independent of the value of x and that the expression is composed solely of the following:

- The variable, x , or its components, x_i .
- Elements of R , that is, constants, coefficients, etc.
- One of the four arithmetic operations in R .
- One of the pre-declared functions, g .
- Auxiliary symbols, for punctuation, precedence, etc.

Let $X \in I(D)$, then the *natural interval extension* of f over X , denoted by $F_N(X)$, is obtained by replacing each :

- variable, x (x_i) by X (X_i),
- real arithmetic operator by the corresponding interval arithmetic operator.
- element of g , by the corresponding element of G .

It can be shown [3] that $x \in X \Rightarrow f(x) \in F_N(X)$. This is due in part to the inclusion isotonicity of the interval arithmetic operators, and in part to the set G being a set of inclusion functions. It also indicates that one can increase the scope of the natural interval extension by increasing the size of the set g and G .

The natural interval extension can be used to bound the range of a function over some domain, however, in general it is not as effective as the other interval inclusions when the

size of the domain is small. It is however, a very general method and can be applied to all extended codeable functions. Furthermore, in Section 6.1.1 we shall extend the natural interval extension to include functions which use logical or conditional statements.

2.4.2 Krawczyk's Centered Form

Krawczyk's centered form [65] is based on parsing a function into a function procedure and then building the function up again using interval arithmetic techniques to obtain a bound for the original function. In the original version, the function procedure (see Section 2.3) used was restricted to computer arithmetic operators only.

Consider the function procedure, S , defined as follows :

- $u_i := x_i$ for $i = 1, \dots, n$
- $u_i := b_{i-n}$ for $i = n + 1, \dots, n + t$ $b_{i-n} \in R$
- $u_i := u_j \circ u_k$ for some $j, k < i$, $\circ \in \{+, -, \times, \div\}$ and for $i = n + t + 1, \dots, s$

If the variable x is replaced by an interval X in the function procedure, S , then the resultant sequence of instructions is called an interval function procedure.

Let $e_i \in I^n$ denote the i^{th} unit vector, $0 \in R^n$ denote the null vector and $c = m(X)$ (or any point in X). Let us be given some function procedure S . Then the interval slope $G = G_i \in I^n$ of S over X is recursively defined by

- $G_i = e_i$ for $i = 1, \dots, n$
- $G_i = 0$ for $i = n + 1, \dots, n + t$
- and for $i = n + t + 1, \dots, s$ one has

$$\begin{aligned}
 G_i &= G_j \pm G_k \text{ if } u_i := u_j \pm u_k \\
 G_i &= G_j F_k(X) + G_k f_j(X) \text{ if } u_i := u_j \times u_k \\
 G_i &= \frac{G_j}{F_k(X)} - \frac{G_k f_j(X)}{F_k(X) F_k(c)} \text{ if } u_i := u_j \div u_k
 \end{aligned}$$

if no division by zero occurs. Again, if division by zero occurs, then the method fails to be valid. We can apply Krawczyk's Centered Form to any function which can be expressed in such a manner.

Let f be a function of the function procedure S and G be the interval slope of S over $X \in I^m$. Then the interval $F_K(X) = f(c) + G \cdot (X - c)$ is called Krawczyk's centered form.

The major disadvantage of this method is that it is not an explicit form. For each function, a function procedure must be found. The fact that the function procedure will in general not be unique and, as shown in Section 3.3.3, since each function procedure will generate a different inclusion interval, the choice of which function procedure to generate is difficult to make.

The advantages of this method are that it is computationally better than most other inclusion functions and as it has been extended to include unary functions [122] and Lipschitz continuous functions [86], it can be applied to a broad range of functions.

Chapter 3

Background : Global Optimization and Automata

In this chapter the basic background for global optimization and the use of automata in relation to global optimization is presented. A summary of the symbols used is given in the List of Symbols.

3.1 Optimization

Consider a real valued function, $f : D \subseteq R^n \rightarrow R$ (we assume $D \neq \emptyset$), and a set of constraints, $g(x) \leq 0$ and $h(x) = 0$.

We define the following as an *optimization problem*:

$$\begin{aligned} \min_{x \in D^*} \quad & f : D \subseteq R^n \rightarrow R \\ \text{s.t.} \quad & g(x) \leq 0 \\ & h(x) = 0 \\ & l_1 \leq x_1 \leq u_1 \\ & \vdots \\ & l_n \leq x_n \leq u_n \end{aligned}$$

where $D^* \subseteq D$ and the constraint functions $g_j(x)$ and $h_k(x)$ are defined for each $x \in D$, and $l_i \leq x_i \leq u_i$ are called *variable bounds* with $l_i, u_i \in \{R, -\infty, \infty\}$.

The function f is usually labelled as the *objective function*. The set defined by the inequality and equality constraints, denoted by C , along with the set of variable bounds, denoted VB , is called the *constraint set*, denoted by CS . A point, $x' \in D$ that satisfies all of the constraints of CS is called a *feasible point*. The set of feasible points is called the *feasible region*. Although the variable bounds with real bounds are technically inequality constraints, they are separated from the inequality constraints as they usually can be dealt with more efficiently by the 'solution process' than the inequality constraints.

The goal of the optimization problem is to find a feasible point x^* such that $f(x^*) \leq f(x) \forall x \in D^*$.

The 'solution' to an optimization problem is obtained by applying an algorithm or methodology to the problem. The process of obtaining a 'solution' is called the *solution process*. At any time during the solution process, the best feasible point that has been found is called the *incumbent solution*.

The above formulations include maximization problems as a maximization problem can be transformed into a minimization problem by simply minimizing the negative of the objective function.

The 'solution' to an optimization problem can be in one of three classes of 'solutions':

O_1 The feasible region is empty (problem is infeasible).

O_2 The feasible region contains at least one point and no x^* can be found such that $f(x^*)$ is finite (problem is unbounded).

O_3 The feasible region contains at least one point and a point x^* can be found such that $f(x^*)$ is finite (problem is feasible and bounded).

We call O_1 , O_2 and O_3 the classes of *outcomes* and by an *outcome* of an optimization problem we will mean a 'solution' that is an element of one of the classes of outcomes.

We label the classes of outcomes as follows: $O_1 = \text{infeasible}$, $O_2 = \text{unbounded}$ and $O_3 = \text{feasible and bounded}$.

By constrained optimization it is meant that the set of inequality and equality constraints, C , is non-empty. By unconstrained optimization it is meant that the set of constraints, CS , is empty. For computational reasons, it is usually assumed that the variables are bound by finite lower and upper bounds. Thus by variable-bounded optimization we mean that the set of constraints, CS , consists only of the variable bounds, VB (with finite bounds

on each variable). A set of finite variable bounds for each variable defines a feasible region that is a box of R^n . For unconstrained or variable-bounded optimization a feasible point is easily obtained either by choosing any point in the domain (unconstrained) or by choosing an element of the box. For constrained optimization the task of finding a feasible point is in general more difficult. We define the *constraint type* of an optimization problem as one of unconstrained, variable-bounded or constrained.

An algorithm that is used to solve an optimization problem may require certain restrictions on the feasible region. For example, some linear programming codes require that all of the variables are non-negative. Thus, non-negativity constraints are enforced by the solution process. We call the set of *user constraints* as the subset of constraints, FC of CS that are not enforced by the solution process.

We define an *IIS* as a minimal set of inconsistent constraints, that is, there does not exist any point of R^n such that all of the constraints in the *IIS* are satisfied.

We define an *IISU* as a minimal set of inconsistent user constraints. To indicate that the *IISU* is dependent on the solution process used, we sometimes use a subscript to indicate the non-user constraints that are enforced. Thus, $IISU_{0 \leq x}$ refers to an *IISU* relative to linear programming codes which require non-negativity.

There are many classes of optimization problems. For example:

- if the objective and constraint functions are all linear functions, and $D^* = D = R^n$ we have linear optimization.
- if D^* is defined to be a neighbourhood about an x^* then we have local optimization.
- if $D^* = D$ we have global optimization.

3.2 Global Optimization

By *global optimization* we mean the task of finding at least one feasible point of the domain of a function such that the functional value of that point is less than or equal to that of any other feasible point of the domain of the function.

Classically, global optimization has been defined as follows:

$$\text{Find at least one } x^* \in A \subset R^n \text{ such that } f(x^*) \leq f(x) \forall x \in A \quad (3.1)$$

where

- A is a compact set defined by $A = \{x \in R^n \mid g_j(x) \leq 0 \ \forall j, \ h_k(x) = 0 \ \forall k, \ l_i \leq x_i \leq u_i, \ l_i, u_i \in R \ \forall i\}$.
- f is a continuous function defined from some domain, D to R , where $A \subseteq D \subseteq R^n$. That is, the function is defined (and thus bounded) for every point of the feasible region.
- $g_i(x)$ and $h_j(x)$ are continuous functions from R^n to R .
- An x^* that satisfies the above is called a *global minimum*.

A more general definition of global optimization occurs if the restrictions of a compact domain is reduced to that of a closed domain, and the restrictions of continuity for the objective function and the constraint functions are removed. We will refer to this generalized definition as *global optimization* and the more restrictive definition as *classical global optimization*.

The major challenge for global optimization is to find the most useful information that can be obtained about the function concerning its behaviour from local information. For example, the rate of change of a function evaluated at a point provides one with either a direction in which strictly lower values can be achieved or the conclusion that the point is a critical point which is inherently local behaviour since this information does not indicate anything about other regions of the function. Except for some special cases (a convex objective function) no global information can be obtained. Thus there does not exist a ‘testable’ condition by which a candidate ‘solution’ can be verified, short of exhaustively searching the feasible region or some subset of the feasible region which is guaranteed to contain all of the global minima.

3.2.1 Possible Outcomes of a Global Optimization Problem

We now examine the types of outcomes that can occur within a global optimization problem. By global optimization we shall refer to the generalized definition (see above Section).

The *infeasible* class has a single outcome labelled INFEASIBLE, the *unbounded* class has two outcomes of DOMAIN UNBOUNDED and UNBOUNDED AT ∞ and the *feasible and bounded* class has two outcomes of GLOBAL MINIMUM and ASYMPTOTIC as defined below. An examination of these outcomes (by class) follows:

- *infeasible*. The set of constraints are inconsistent. That is, the feasible region is empty. For example, the set of constraints may include $x_1 \geq 4$ and $x_1^2 + x_2^2 \leq 4$ for which no x_1 can satisfy both constraints.
- *unbounded*. The feasible region is non-empty but the objective function is not bounded from below for at least one point of the feasible region. That is, there exists a feasible point unbounded from below. Two possible sub-cases can occur.
 - DOMAIN UNBOUNDED. There exists a finite point feasible unbounded from below. For example, consider the objective function $-1/x^2$ over the feasible region $-10 \leq x \leq 10$ for which the function is unbounded from below at $x = 0$. We call a finite feasible point unbounded from below an *unbounded domain point*.
 - UNBOUNDED AT ∞ . The function is unbounded as some variables go to $\pm\infty$. For example, consider the objective function $-x^2$ over the feasible region $x \geq 0$ for which the function is not bounded as $x \rightarrow \infty$.
- *feasible and bounded*. The feasible region is non-empty and the objective function is bounded from below for all feasible points. Two possible sub-cases can occur.
 - GLOBAL MINIMUM. At least one global minimum exists. The infimum (the greatest lower bound) of the function over the feasible region is attained by a feasible point of the function. For example, consider the objective function e^x over the feasible region $x \geq 0$ which has a global minimum at $x = 0$.
 - ASYMPTOTIC A global minimum does not exist. What has happened is that the infimum for the function over the feasible region is never attained by the function. For example, consider the function e^x to be minimized for $x \leq 0$. The function is bounded from below but there is no global minimum as the infimum of e^x over $x \leq 0$ is 0, but $e^x > 0$ for all $x \leq 0$. The problem with this example is that the feasible region is unbounded. However, even with a bounded feasible region this can occur. Consider the following function:

$$f(x) = \begin{cases} 2 & \text{for } x \leq 0 \\ x^2 & \text{for } x > 0 \end{cases}$$

This function has an infimum of 0, but no global minimum. In this example the problem is that as $x \rightarrow 0$ from the positive side, the functional value goes to zero, but never attains zero. At $x = 0$, the function is discontinuous.

We note that a problem may have more than one global minimum. In fact there may be an infinite number of them, possibly disjoint.

We note that classical global optimization (as defined by Equation 3.1) can only have outcomes INFEASIBLE and GLOBAL MINIMUM. An UNBOUNDED outcome does not occur as the objective function is defined (and thus bounded) for all elements of the feasible region and the ASYMPTOTIC outcome does not occur as the objective function is continuous and the feasible region is restricted to a compact set.

3.2.2 Local Optimization

We define a local minimum x_l^* as a point of the feasible region D such that there exists an $\epsilon > 0$ such that $f(x) \geq f(x_l^*)$ for all feasible $x \in D$ within a distance ϵ of x_l^* , that is, for all feasible points within a ϵ -neighbourhood, $N(x_l^*, \epsilon)$ of x_l^* . By *local optimization* we mean the task of finding at least one feasible point, x_o of the domain of a function such that the functional value of that point is less than or equal to that of any other feasible point within an ϵ -neighbourhood of x_o . By a *descent method* we mean the application of a strictly descending local search method to a point of the domain. Thus a descent method is a special class of local optimization methods.

We note that the solutions generated by most local optimization methods may be saddle points (where the curvature of the function changes and the derivative is zero) as well as local minima, as they employ a stopping condition that is zero or the derivative in all feasible directions is greater than or equal to zero.

Local minima are important as the set of all local minima must include all of the global minima. An important necessary set of conditions for a point to be a local minimum are the Kuhn-Tucker Conditions [71] for differentiable functions (see Appendix A.1).

3.2.3 Problem Test Sets

There are many existing test sets for Global Optimization. For example, Conn [16], Floudas and Pardalos [26], Gomez and Levy [35], Hansen [46], Hock and Schittkowski [49], Himmel

blau [48], Levy, Montalvo, Gomez and Calderon [69], Schwefel [108], and Törn and Žilinskas [115] (see Appendix B for samples from some of these test sets). These problems are of various constraint types: unconstrained, constrained, variable-bounded, however, all have FEASIBLE AND BOUNDED outcomes.

3.3 Automata and Global Optimization

Murty and Kabadi [85] show that several problems within quadratic nonlinear programming are NP-Complete. Thus the general global optimization problem is at least that complex. As closed form solutions for general global optimization problems do not exist, an automaton (or computer) is required to carry out the solution process. The use of an automaton complicates the problem as it requires that :

- the algorithm terminates after a finite number of operations.
- the finite space limitations of the automaton can not be exceeded.
- only a finite set of numbers, variables and functions can be represented by the automaton.

For a given automaton, Θ , we label the set of numbers that can be represented as N_Θ . This set (assumed to be non-empty) contains both a least and greatest element, labelled lbN_Θ and ubN_Θ respectively.

In addition we define a hyper-rectangle, $H_\Theta \in I^n$, of R^n as the region of R^n that is 'bounded' by the set N_Θ as follows: $H_\Theta = \{x \in R^n \mid lbN_\Theta \leq x_i \leq ubN_\Theta \text{ for } i = 1, \dots, n\}$. Thus the region H_Θ contains all of the real vectors that can be bound by an interval whose endpoints are representable by the automaton, Θ . We label the space of all intervals of I^n whose endpoints are representable by Θ as I_Θ^n .

By $HISU_\Theta$ we will mean a minimal inconsistent set of free constraints relative to H for the automaton Θ . That is, the constraints $lbN_\Theta \leq x_i$ and $x_i \leq ubN_\Theta$ for $i = 1, \dots, n$ are enforced by the algorithm that is implemented on the automaton Θ .

We now consider a mapping, $M_{\nabla, \Delta}$, from H_Θ to I_Θ^n defined as follows :

$$M_{\nabla, \Delta}(x) = ([\nabla x_1, \Delta x_1], \dots, [\nabla x_i, \Delta x_i], \dots, [\nabla x_n, \Delta x_n]) \quad x \in H_\Theta$$

where ∇ (resp. Δ) means to round down (resp. round up) to the next number representable by the automaton, Θ . That is, every real number in H is mapped to the smallest interval of I_Θ^n that contains it.

In addition we define the following two operators, $\triangleleft$ and $\triangleright$, acting on a representable number, x_r , on Θ as the operation of finding the next smallest and largest representable number from x_r respectively. For completeness we define $\triangleleft(lb.N_\Theta) = lb.N_\Theta$ and $\triangleright(ub.N_\Theta) = ub.N_\Theta$.

The use of the automaton in global optimization introduces sources of error into the solution process. As well, the functions themselves must be represented on the automaton, as some finite sequence of operations. Each function may have many possible representations, each with a different amount of error associated with it. Thus, the choice of which representation to use can effect the overall performance of the solution process.

We classify the effects of the use of an automaton for global optimization into four classes: FINITENESS, DILEMMA OF EQUALITY, FUNCTION REPRESENTATION, and RESTRICTED FUNCTION DOMAINS. Technically RESTRICTED FUNCTION DOMAINS are not effects due to the use of an automaton, but we include them as they are themselves affected by the use of the automaton. We now examine each of these classes.

3.3.1 FINITENESS

The first effect due to the use of automata is brought about by the finite nature of the automaton. The finiteness is exhibited in that only a finite set of numbers, variables, and operators, are allowed and that only finite sequences can be evaluated.

The restriction to a finite set of numbers basically ensures that the result of any arithmetic operator is not closed for the set of numbers. That is, the multiplication (over the reals) of two numbers from the set will result in a number that is not an element of the set. Thus, a systematic method for representing the result of the operation must be employed. The set of numbers (n -dimensional vectors) that are representable on the automaton can be thought of as an irregular screen of points that covers some finite section of the real number line (of R^n). The result of an arithmetic operation can result in a point that (in general) lies between the screen points and therefore can not be represented by the automaton. The problem is to determine how to represent this result. This type of error is called a round-off error.

The traditional method is to choose a rounding scheme that will obtain a 'close' approx-

imation to the result of the operation by choosing an appropriate rounding scheme. That is, the result is rounded to one of the two screen points that the point lies between. The difficulty with this approach is that this type of rounding does not provide a method by which the amount of error that is caused by the rounding can be measured.

An alternative method is to represent the result not as a single point but as a range of points. That is, the result is the interval of points between the two enclosing screen points. That is, interval analysis can be used to bound the amount of error in the operation. Thus the result of each operation is an interval that can be guaranteed to contain the correct answer.

The least and greatest number representable by the automaton limit not only the range of values that each variable can take, but also limit the range of functional values. Although it may be reasonable to impose finite bounds on the variables, it is difficult to justify finite bounds on the functional values as the given problem may be unbounded.

The restriction to a finite set of operators coupled with the restriction to finite sequences results in only a finite set of functions that are representable by the automaton. Thus only a finite subset of the set of all real functions can be represented by the automaton. Thus, if there is a function in the global optimization problem that is not representable by the automaton, then an approximation to the function must be made if a solution is to be determined. In addition, the sequence used to represent a function is not unique. Each representation corresponds to a sequence of operations. The amount of error that occurs for each sequence depends on the sequence of operations and thus each sequence will produce a different amount of error. Thus the choice of the representation used for a function will affect the accuracy of the result (see Section 3.3.3).

The restriction to finite sequences requires the premature termination of infinite sequences. Thus, as most solution processes are designed so that they converge to a solution after an infinite number of iterations, these processes must be prematurely terminated. The termination occurs when an *a priori* determined condition occurs. In most cases this condition is determined by successive elements of the sequence. For some sequences this means that the solution obtained after the termination of the sequence is 'relatively' close to the proper solution. However, if the sequence is irregular in that two or more successive steps are small enough to trigger the termination condition but they are not near the proper solution, then the resultant error due to the premature termination can be quite large. Fortunately,

for many of these processes, an error term exists which can be used to bound the amount of error introduced by a truncation of the process.

3.3.2 DILEMMA OF EQUALITY

The second effect is the dilemma of equality. The problem is in determining if an equality relation is satisfied on a finite set of numbers. It is essentially guaranteed that an equality relation defined originally over the reals will be unsatisfiable over a finite number system. As the equality relation is used to test for termination criteria, gradients being zero, and feasibility, its use is unavoidable. If the equality relation is relaxed so that it is considered to be satisfied within a certain tolerance level, there is the risk of considering the relation satisfied when in truth it is not. If the relation is not relaxed, then there is the risk of considering the relation unsatisfied when in truth it is. Regardless of the decision taken errors in the logical operation of the solution process will likely occur.

As the error is logical in nature its effect will go beyond the simple addition of extra uncertainty in the final result, but may actually produce erroneous results. For example, the solution process may say that the problem is infeasible but in truth it is not.

Equality Constraints

A special subclass of the DILEMMA OF EQUALITY deals with the problem of equality constraints. As mentioned in Section 2.2, equality constraints define a region of dimensionality less than the space in which they reside. When this equality constraint is evaluated on the automaton, it is unlikely that this lower dimensional object will intersect any of the points representable by the automaton. Thus, the problem is to determine if the reason that the equality constraint is not satisfied is due to the finite nature of the automaton. This is a difficult problem as the process of evaluating the constraint is not error free and adds to the degree of difficulty.

One obvious 'solution' to this is to try and eliminate the equality constraint from the problem. The equality constraint can be eliminated by solving the constraint for one of its variables and then substituting the resultant expression for each occurrence of that variable into the global optimization problem. This has to been done either by the use of symbolic manipulation (see Section 10.3) or *a priori* by the user of the solution process. If this can not be done other approaches are required.

Another such approach is to relax each constraint by some small amount. This can be carried out by replacing the equality constraint $h(x) = 0$ with the two inequality constraints $h(x) \leq 0$ and $-h(x) \leq 0$ and then relaxing each of the new constraints by some amount, say α , to get $h(x) - \alpha \leq 0$ and $-h(x) - \alpha \leq 0$.

The problem with this approach is that relaxation has introduced the possibility of a logical error into the solution process. For example, the relaxation may have created a feasible region where none should exist. If this type of logical error is suspected, then by reducing the value for α more information can be obtained about the validity of relaxing the equality constraints. However, adjusting the amount of relaxation may not provide any answers to the problem, partially due to the round-off errors that are present during the evaluation of the constraint and partially due to the finite graininess of the screen of representable points.

In an attempt to decide if a constraint is satisfied within a certain region (or if the problem is feasible), a test for the existence of a solution of a set of equality constraints (based on the Brouwer fixed-point theorem) (see [45, 83, 97]) could be used. This test is robust, but it may require a large amount of time. This test for the existence of a solution is applicable if the equality constraints are all elements of C^1 , that is, they are continuously differentiable everywhere, which allows the definition of a Jacobian matrix as required by the interval Newton's Method (see Appendix A.4.3) used by the method.

3.3.3 FUNCTION REPRESENTATION

An automaton is restricted to a finite set of operators, numbers and variables. These sets of operators and variables are the building blocks that are available for the creation of a representation for a function. The set of variables and operators as well as a set of implicit rules that govern the use of the operators will in turn restrict the type of function that can be represented. Thus the choice of the set of operators, variables and rules will determine the set of functions that can be represented by the automaton.

As noted in Section 2.3, the sequence of functions that define a codeable function can be used to represent the function on an automaton. However, in general, the representation will not be unique and one must choose one representation out of a large set of possible representations. Automated methods for the generation of a representation for a function have been developed by Rall [92] and Griewank [36]. The input to their methods is a

function in *expression* form (for example $f = (x * y + \sin(x - z) + 4)^3 / (3 - \ln(x^2))$). Their prime motivation has been in connection with automatic differentiation, where the sequence of functions that are generated are useful, in that each function has a *simple* derivative and each operator has a chain rule associated with it.

At first glance an alternative method for representing a function uses a listing of the functional values for some finite set of representable domain points. At one extreme, all of the feasible domain points are listed, while at the other, only a few domain points are listed and a means of interpolating a functional value for a non-listed domain point is then provided. However, the listing is a set of constant points (which can be listed in some sequential form) and the interpolation is a sequence of operations representable by the automaton. Thus, this type of representation is just a special case of a codeable function.

A function may have a large number of representations. For arithmetic on an automaton, the result of an evaluation of the function will be dependent on the representation used as each representation will have its own set of roundings associated with its evaluation, which will in differ from those roundings associated with the other representations.

Effect of Function Representation on Bounding Functions

We examine the function [97] $f(x) = 1/x^2$ to show how the choice of function procedure or representation has an effect on the Krawczyk's centered form and on the natural interval extension respectively. In general, the choice of function procedure or representation will affect all bounding functions in that they will produce different bounds.

Let $f(x) = 1/x^2$ be defined on the interval $X = [2, 4]$ with $c = m(X) = 3$. Let a function procedure, Fp_a , be given by $f_1 := x$, $f_2 := 1$, $f_3 := f_1 \times f_1$ and $f_4 := f_2 \div f_3$ and a second function procedure, Fp_b , be given by $f_1 := x$, $f_2 := 1$, $f_3 := f_2 \div f_1$ and $f_4 := f_3 \div f_1$.

For Fp_a (respectively Fp_b), Krawczyk's centered form generates $G_a = [-7/36, -5/144]$ (resp. $G_b = [-5/36, -7/144]$) and since $F(X) = f(c) + G(X - c)$, $F_a(X) = [-1/12, 11/36]$ (resp. $F_b(X) = [-1/36, 1/4]$). Thus we have that $F_b(X) \subset F_a(X)$.

For $f(x) = x - x^2$ [97] let one function representation, f_p_a , be $x - x^2$ and let a second representation, $f_p_b(x)$, be $x(1 - x)$. The natural interval extensions for f_a and f_b on $X = [2, 4]$ are respectively $F_a(X) = X - X^2 = [-14, 0]$ and $F_b(X) = X(1 - X) = [-12, -2]$.

In general, the bounds determined by the different representations will not form subset relationships. If the bounding functions are correct, then they will all contain the correct

bounds. Thus the intersection of the bounds determined for different representations will be correct bounds as well.

Thus the choice of representation or function procedure will affect the accuracy of the interval inclusion. The issue of which representation to use has largely been ignored. The only advice that is given is in connection with interval extensions, where one is advised to keep the expression form of the function as *simple* as possible, meaning to keep the number of occurrences of a variables at a minimum.

We note that a function representation and a function procedure are equivalent concepts. They differ in that the function procedure makes explicit the sequence of operations, while the representation relies on the implicit precedences of the operators used. For example, for the representation $f(x) = x - x^2$, the corresponding function procedure is $f_1 = x$, $f_2 = f_1 \times f_1$, $f_3 = f_1 - f_2$ as the exponent operator has precedence over the subtraction operator.

3.3.4 RESTRICTED FUNCTION DOMAINS

There is an additional consideration dealing with the implementation of non-linear functions on an automaton and that is that the domain of the function implemented may be restrictive (hence this effect is called RESTRICTED FUNCTION DOMAINS). For example, the logarithm function is only defined for strictly positive reals. Thus the use of the a function of the form, $\ln(f(x))$, in the objective function or in one of the constraints will implicitly imply a constraint of the form $f(x) > 0$. We call this type of constraint a 'hidden' constraint.

Problems arise if the domain space of the method used to solve the problem contains points which correspond to the restricted region of a hidden constraint of a given function as the function (or bounding function) is not defined for these regions.

Chapter 4

Research Problem

4.1 Introduction

In this chapter we examine the state of the art in global optimization methods considering the issues of required problem restrictions and the information produced when implemented on automata. We then develop the concept of a global optimization system by examining the characteristics of a global optimization problem, how the use of an automaton affects global optimization, and the issue of providing 'useful' modelling information. We then discuss how such a global optimization system can be tested. We conclude with a statement of the main research goal of the dissertation.

4.2 Analysis of the State of the Art in Global Optimization Methods

All of the methods that attempt to solve global optimization problems are necessarily based upon an exhaustive search of the domain of the problem (or of a subset of the domain for which it is known that the global minima are elements thereof). This is due to the lack of any other condition by which a point can be verified to be a global minimum.

We now summarize the state of the art. A more detailed examination of these methods is available in Appendix A. Specifically, for each class of methods we examine the following characteristics :

- RESTRICTIONS

- *outcome and constraint types* - The classes of outcomes (infeasible, unbounded, feasible and bounded) (see Section 3.2.1) and the types of constraints (variable bounds, inequality, equality) that are allowed.
- *problem size* - The maximum size of the problems that can, in general, be solved as well as the reason for the limitation. Specifically, is it the time required or space requirements that limit the effective problem size solvable?
- *functional* - The restrictions on the class of functions that can be used, if any, as well as the reason for the restriction are examined. For example, a procedure may require that all of the functions be differentiable.

- INFORMATION PRODUCED (When implemented on an automaton)

- *output upon termination* - Are the methods robust, that is, are they guaranteed to 'succeed'? What information is output for the possible outcomes? We note that effectively, all methods can fail due to space and/or time limitations and may exit with no output when such a failure occurs. What information is present at any given time in the solution process that can be output, with minimal computation, if imminent failure due to space and/or time limitations is detected? We call such a termination a *machine initiated termination*.
- *formulation assistance features* - How 'useful' is the output from a modelling point of view? Is causal information for infeasible and/or unbounded outcomes provided? If not, can causal information be provided by modifications to the methods? Is any information available which would provide information on the accuracy of the incumbent solution for feasible and bounded outcomes? Is a type of 'sensitivity analysis' available?

The methods have been classed together according to their characteristics. The 'size' of a global optimization problem is defined as follows: small has ≤ 10 variables, medium has 11 to 100 variables and large is > 100 variables.

4.2.1 Covering Methods and the Interval Newton's Method

The covering methods [11, 24, 40, 41, 51, 52, 54, 55, 56, 59, 77, 81, 89, 90, 97, 99, 110, 111, 121] employ a base branch and bound algorithm to systematically search the problem space. An underestimating cover of the objective function is created over the feasible region of the problem. The minimization of this cover over a region is used to bound the functional values for the specified region. Branching is usually carried out by splitting each region into similarly shaped sub-regions.

The interval Newton's method [39, 43, 44, 64, 79, 81] is Newton's method extended by the use of interval analysis so as to maintain a list of regions which must include the set of global minima.

• RESTRICTIONS

- *outcome and constraint types* - These methods can generally be structured and/or modified so that they can be used to detect all possible outcome types and can be used with all types of constraints.
- *problem size* - These methods are applicable only to small to medium sized problems as the number of sub-regions that must be stored at any particular instance grows exponentially in terms of the number of variables.
- *functional* - These methods restrict the classes of functions by the procedure(s) used to bound the functions for the covering methods and by formulation of Newton's method in the interval Newton's method. Some of the covering methods require *a priori* knowledge (for example, Lipschitz constants or differentiability), but very general covering methods, applicable to codeable functions, are available. Optional procedures such as local optimization routines and other acceleration devices may add further restrictions such as differentiability, etc. The interval Newton's method requires that the functions are differentiable.

• INFORMATION PRODUCED

- *output upon termination* - These methods are in general robust. Upon machine initiated termination these methods will (or can) return the value of the incumbent solution as well as a value for the lower bound and the set of regions that have not been eliminated, from which the current state of the boundedness and feasibility of

the problem can be determined. Currently, some methods will detect and report on infeasible problems. Some will indirectly report an unbounded problem if the incumbent solution is less than some *a priori* determined cutoff value. For feasible and bounded outcomes, these methods usually return the incumbent solution, as well as a lower bound on the functional values. The Interval Newton's method also returns the set of regions which contain the set of global minima and can also return local minima.

- *formulation assistance features* - These methods do not provide any causal information for an infeasible outcome, but provide partial causal information for an unbounded outcome if the point for which the 'cutoff' is exceeded is reported as this value can possibly be used to verify that the problem is unbounded. For a feasible and bounded outcome, the accuracy of the incumbent solution is provided by a lower bound on the global minimum. The covering methods can generally be structured so that they can be used to provide causal information for infeasible and unbounded outcomes. This may also be possible for the interval Newton's method, although it is not likely to be a simple modification. If the outcome is feasible and bounded and the incumbent solution, the sub-region in which it resides and the functional bounds for that sub-region are output, then this information is useful to the modeller as a bound on the 'flatness' of the objective function about the solution is available from examining the size of the region in relation to the variation in the functional values over the region. Sensitivity analysis is partially available if the parameters are modelled as intervals for interval covering methods and the interval Newton's method.

4.2.2 Stochastic Methods

The stochastic methods [5, 8, 9, 47, 78, 101, 107, 113, 116, 115] (except for simulated annealing [4, 18, 63, 75, 117] and tabu search [30, 31]) implicitly search the feasible region using a uniform sample as a base. This base is then expanded upon to locate local minima, the best of which is the incumbent solution. The more sophisticated methods employ clustering techniques and local optimization methods to reduce the sample size required.

In simulated annealing a trajectory is generated that attempts to find the region of attraction of the global minimum and then finds the global minimum by applying a local

optimization method. Tabu search is a collection of adaptive procedures used to modify other search methods so as to prevent the search from being trapped at a local optimum or in a previously searched region.

● RESTRICTIONS

- *outcome and constraint types* - These methods are best suited for variable-bounded problems (and thus feasible problems), in part due to the requirement for a uniform sampling. The use of penalty functions will allow these methods to be used with constrained problems. However, if the feasible region of the problem is small compared to that of the region defined by the variable bounds, then these methods become inefficient as they will search in infeasible regions as well as in feasible regions. They can detect unbounded problems purely by chance, or by the failure of a local optimization routine and are ineffective in determining if a problem is infeasible, although as soon as one feasible point is found, it can 'detect' a feasible problem. Thus they are best suited for variable-bounded feasible and bounded problems only. Simulated annealing and tabu search differ in that they can possibly be applied to problems with infeasible regions if a mechanism that allows the trajectory to find and then jump across infeasible sub-regions to a new feasible sub-region could be developed as well as a means of locating an initial feasible point.
- *problem size* - These methods are applicable only to small to moderately sized problems due to either the time required to complete a uniform sampling of the feasible region and/or the time required for the multiple use of local optimization methods. Clustering techniques do reduce the number of local optimization procedures and sample points that are required, but introduce procedures to find the nearest neighbour which are practical only for small to moderately sized problems due to time constraints. The simulated annealing, genetic algorithms, and tabu search methods have been applied to some large problems with mixed success, in part due to the difficulty of properly modifying the control parameters.
- *functional* - These methods restrict the classes of functions that can be used in the procedures to find the nearest neighbour and/or in any local optimization methods used, usually requiring differentiability or continuity of the functions, although the

simplistic stochastic methods can be applied to any codeable function.

- **INFORMATION PRODUCED**

- *output upon termination* - In relation to variable-bounded feasible and bounded problems, these methods may fail to produce the global minimum if the initial sample is not large enough, or not enough iterations are carried out to effectively cover the problem space. Thus they are not robust, but will usually produce a good incumbent solution relatively quickly. The output is the incumbent solution for normal or machine initiated termination. An unbounded problem may be detected if the incumbent solution returned is less than some *a priori* determined cutoff value.
- *formulation assistance features* - No causal information is provided and the inclusion of such information seems unlikely. Partial causal information for an unbounded outcome is provided if the point which signals a possible unbounded point is returned. For the simulated annealing method as well as the simpler stochastic Methods, only a set of points (not necessarily local optima) can be returned, and thus very little information is returned from which a modeller can determine something about the characteristics of the function. If enough points are returned the resultant grid may be of use to determine an outline of the function, but is likely to be impractical for anything but small problems. For the more sophisticated methods, information on the location of other local minima, as determined by the use of local optimization methods could be of interest to a modeller as a means of examining the general shape of the function, although only a rough idea of the function can be gleaned in such a manner as information on the function between the local minima is not provided.

4.2.3 Penalty and Trajectory Methods

The penalty methods [27, 28, 34, 35, 68, 119] attempt to locate a local minimum and if found, then the objective function is modified so as to remove that local minimum. The process is repeated until no more local minima can be found. The 'best' of the set of local minima produced is the incumbent solution. The trajectory methods [10, 37, 58, 57] are based on the solution of a differential equation which describes the motion of a point mass on the surface

of the function. We note that a 'penalty method' is different from a penalty function, where the objective function is modified so as to penalize any points that are infeasible.

● RESTRICTIONS

- *outcome and constraint types* - These methods are not easily modified to detect infeasible or unbounded problems as they are designed primarily for unconstrained or variable bound problems as each point is assumed to be feasible. Constrained problems can be solved, by transforming the problem with a penalty function. However, this complicates the solution of the resulting differential equation and the determination of the next local minimum. Thus these methods are suited primarily for variable-bounded feasible are bounded problems.
- *problem size* - These methods are practical in general only for small to medium sized problems due to space requirements of the procedures that are used to search for the next local minimum or for those problems with a small number of local minima.
- *functional* - These methods restrict the set of functions in the procedures used to solve the differential equations or to determine the next local minima. Thus the method usually requires differentiable functions.

● INFORMATION PRODUCED

- *output upon termination* - These methods are in general not robust, as either difficulties occur in determining the next local minima (for trajectory methods) or the parameters for determining the trajectory are set so that the global minimum is bypassed. Information is limited to the incumbent solution and to the set of local minima that have been found, for both normal and machine initiated terminations. An unbounded problem may be detected if the incumbent solution returned is less than some *a priori* determined cutoff value.
- *formulation assistance features* - No causal information is given and the inclusion of such information seems unlikely. Partial causal information for an unbounded outcome is provided if the point which signals a possible unbounded point is returned. Information on the location of other local minima, as determined by the use of local optimization methods could be of interest to a modeller as a means

of examining the general shape of the function, although only a rough idea of the function can be gleaned in such a manner as information on the function between the local minima is not provided.

4.2.4 Global Boundary Methods

Those methods which attempt to transform the problem such that the global solution is on the boundary of the feasible region of the transformed problem are the outer approximation methods [53, 55] and the relief indicator methods [114].

● RESTRICTIONS

- *outcome and constraint types* - These methods are restricted to feasible and bounded problems, in part due to the approximation methods used and in part to the requirements of the bounding functions used. These methods have been developed for the Global Optimization Problem which has an open-ball feasible region, that is, an open ball of some finite radius can be found such that each point of the open ball is feasible. This restriction is required to ensure the convergence of these methods to a global minimum. As a consequence no equality constraints are allowed, and some sets of inequality constraints are also disallowed. Unbounded problems are essentially not allowed, as the transformation of a general problem requires differentiability at every point, and thus the only unbounded points possible are those at infinity. Thus the methods are applicable only to a special sub-class of constrained feasible and bounded problems.
- *problem size* - These methods are restricted to moderately sized problems due to the large growth in the number of constraints (and thus extreme points) that make up the approximation to the boundary.
- *functional* - These methods restrict the classes of functions that can be used in the procedures used to generate the next separating constraint and to transform the problem to a boundary problem and as such usually require differentiability.

● INFORMATION PRODUCED

- *output upon termination* - These methods are generally robust if their conditions of open-ball feasibility and boundedness are met. For both normal and machine ini-

tiated termination an incumbent solution along with a lower bound on the global minimum is output. An unbounded problem may be detected if the incumbent solution returned is less than some *a priori* determined cutoff value.

- *formulation assistance features* - Accuracy of the incumbent solution is provided by a lower bound of the global minimum. No additional information on which to determine the sensitivity of the solution and/or function information is output and the likelihood of such information being found is small due to the complication of the required transformation.

4.2.5 Conclusions

In summary:

● RESTRICTIONS

- *outcome and constraint types* - The majority of the methods are applicable only to variable-bounded feasible and bounded problems. Only the covering methods and the interval Newton's method are (or can be easily modified to be) applicable to general problem types with any constraint types.
- *problem size* - The robust methods are limited to small to medium sized problems, due essentially to the requirement for an exhaustive search. Non-robust methods can be applied to larger problems, but lack means by which the solution can be validated.
- *functional* - The majority of the robust methods require differentiability, while the non-robust methods can be used with more general functions. The exception is the covering methods, which are robust and can be used on codeable functions.

● INFORMATION PRODUCED

- *output upon termination* - Most methods will output only the incumbent solution and/or a set of local minima. Some methods, such as the covering methods and the interval Newton's method will provide a lower bound on the global minimum, if it exists.

- *formulation assistance features* - Accuracy of the incumbent solution is provided by a lower bound for the global minimum in the covering methods, the interval Newton's method and the global boundary methods. Causal information for an unbounded outcome is partial, in the sense that most of the methods can be modified such that if a large negative incumbent solution is found, it can possibly be used to identify a domain unbounded point. However, causal information for an infeasible outcome does not exist, although it can be introduced into the general covering methodology. For the majority of methods little, if any information, is given which will be useful to a modeller. The exceptions are the covering methods and the interval Newton's method which can provide information which can be used to determine a bound on the 'flatness' of the objective function about the solution.

Discussion

Because most methods assume that a finite global minimum exists, the state of the art in global optimization methods can not be naively applied to any global optimization problem, as in general, no information will initially be available on the feasibility and/or boundedness of the problem. Even if the system or problem is feasible and bounded, syntactic errors as well as other modelling errors may produce a global optimization problem that is infeasible or unbounded. This implies that any real application of these methods can only be undertaken with a good understanding of the problems and issues of global optimization. This opinion is corroborated by Conn [15].

Furthermore, as most methods do not provide any causal information about infeasible or unbounded outcomes, the task of determining the underlying cause of the outcome is left to the modeller as well. Thus we conclude that the state of the art in global optimization methods is not practical for a general modeller, as the methods are limited in their scope and require extensive knowledge for their use.

The most promising (by type of output) of the methods currently being developed for global optimization are:

- the cluster-based stochastic methods and simulated annealing to generate a good incumbent solution quickly for variable-bounded and bounded problems.

- the interval Newton's method and the branch and bound methods to provide robust results for small to medium general problems.
- the global boundary methods for small to medium differentiable open-ball feasible problems.

4.3 Towards a Global Optimization System

In this section we formally develop a definition for a global optimization system, that is, a methodology for the formulation and solution of global optimization problems. We examine how the use of an automaton affects global optimization, specifically, how the set of possible outcomes is affected, how the finiteness of the automaton affects the solution process, simplifications that arise, and discuss formulation aids such as causal information (see Section 1.0.2), solution accuracy, objective function 'flatness' and sensitivity analysis. A global optimization system is then formally defined and the testing of such a system is discussed.

4.3.1 Introduction to Global Optimization Problems

A global optimization problem, as we have defined it, (see Chapter 1) arises from an application of mathematical modelling to solve some problem in engineering, economics, etc. As such it is reasonable to make the following assumptions:

- Little if any information may be known *a priori* about the outcome of a solution process applied to a global optimization problem. That is, the problem may not have a global minimum as it may be infeasible or unbounded.
- In general, *a priori* knowledge about conditions such as the differentiability, the continuity, or the existence of a Lipschitz constant for the functions used is known, unless they are simple (for example, linear and polynomial functions are known to be differentiable, continuous, etc.).
- The resources are finite. This translates into finite lower and upper bounds on all of the variables. This is reasonable in that it is unlikely that an infinite supply of any resource is available.

- The problem is sufficiently complex that a solution can only be obtained by the use of an automaton.

We note that an exact global solution to a global optimization problem is not realistic from a practical perspective due to the fact that no finite method can identify the global minimum of any function for the global optimization problem [21]. Thus from a practical perspective (which requires a finite method) the goal of finding a global minimum must be modified.

Two possible modifications are :

1. to identify a global minimum within some *a priori* determined limit or to the maximum precision possible.
2. to place restrictions on the functions and constraints of the problem.

From a practical perspective, any restrictions placed on the problem must be verifiable and the set of restrictions that are usually proposed (such as convexity, continuity, and differentiability) are often too restrictive as well as difficult to determine. As a global optimization method would be used to solve general global optimization problems and as the modelling process is in itself fraught with errors and inaccuracies, a goal of identifying the global minimum within some specified limit is the more reasonable and practical of the two choices. In addition, formulation aids, which help examine the outcome of the solution process as well as the problem space, are desired, as in this way the modeller can examine how well the model has performed and decide if any changes are necessary.

4.3.2 Global Optimization and Automata

In this section we examine the effects of using an automaton to solve global optimization problems and the implications for a global optimization system. Specifically, we examine the set of possible outcomes as well as issues and simplifications arising from the use of the automaton.

Let Θ be the automaton used to solve the global optimization problem and lbN_{Θ} and ubN_{Θ} be respectively, the smallest and largest numbers representable on the automaton Θ .

Global Optimization Problem Outcomes

We now explore how the various outcomes of a global optimization problem, { INFEASIBLE, DOMAIN UNBOUNDED, UNBOUNDED AT ∞ , GLOBAL MINIMUM, ASYMPTOTIC }, (see Section 3.2.1) are affected by the use of an automaton. We note that ASYMPTOTIC cannot occur due to the finiteness of the automaton and that UNBOUNDED AT ∞ does not occur as the automaton requires that the variables are bound by finite values. In addition, as the automaton is a finite machine, an additional outcome exists, labelled INDETERMINATE, in which the automaton terminates before one of the outcomes of a global optimization problem are identified.

The set of outcomes of the problem center on the status of the feasibility and boundedness of the problem. We refer to the *state* of the problem as the feasibility and boundedness of the problem. The feasibility (boundedness) status can be feasible (bounded), infeasible (unbounded) or unknown. The possible states for each outcome are as follows, denoted by (feasibility status, boundedness status):

- INFEASIBLE - I (infeasible, unknown); II (infeasible, bounded)
- DOMAIN UNBOUNDED - III (feasible, unbounded)
- GLOBAL MINIMUM - IV (feasible, bounded)
- INDETERMINATE - V (unknown, bounded); VI (unknown, unknown); VII (feasible, unknown)

Note that a problem can only be unbounded if a feasible point exists, as we are only interested in the boundedness of the set of feasible points and thus states (infeasible, unbounded) and (unknown, unbounded) are not possible.

Usage of Automaton

The use of the automaton for solving Global Optimization Problems raises the following issues (see Section 3.3 for a more complete discussion on these issues):

- FINITENESS - the set of numbers representable, the number of calls to any procedures, and the number of iterations of an algorithm are required to be finite. This leads to roundoff errors, and termination errors.

- **DILEMMA OF EQUALITY** - when are two calculated values equivalent? Most methods require the ability to identify equivalent values, but any calculation involving real numbers introduces roundoff errors. Thus it is highly unlikely that the result of any calculation will equal any specified value, so a certain amount of relaxation of all equality constraints is required. However, this leads to the possibility of accepting the equivalence of two values which are not equivalent, so errors of a logical nature (errors in decisions based upon the equivalence of two values) are possible.
- **RESTRICTED FUNCTION DOMAINS** - The objective and/or constraint functions may include functions whose domains are restricted. Thus there are constraints to the domain of the problem which are not explicitly part of the constraint set. This can affect the process of determining a functional value or a functional bound for a function over a region.
- **FUNCTION REPRESENTATION** - For general functions, more than one representation is always possible. These different representations will generate different functional values, different bounds of functional values, etc., for the same point or region of the domain. The use of one representation instead of another may lead to a decrease in accuracy, and thus require more computational effort than necessary.

Overall, the use of an automaton introduces errors (rounding to the set of machine representable numbers, equality error tolerances, etc.) and/or inefficiencies (due to a 'poor' choice of function representation, etc.) into the solution process. These errors must be accounted for and minimized if the given problem is to be analyzed properly.

Simplifications due to the Use of Automata

As well as introducing errors into the solution process, the use of an automaton also simplifies the problem. For example, only extended codeable functions can be represented on the automaton, for which there exist interval analysis methods for bounding the range of the function. In addition, there is usually a large amount of information available about an extended codeable function (or its coded representation) and thus important characteristics such as inflection points, local and global optima are readily available. This information may be useful in determining lower bounds as well as unbounded points of the function within the feasible region.

The most important simplification is that the problem has been reduced to a finite problem. That is, there is only a finite number of possible feasible points, etc. This has important consequences for determining proper stopping conditions for the solution process. For example, as long as the solution process will not examine any point more than a finite number of times then as long as the solution process does not exceed the space capacity of the automaton, the solution process will terminate in a finite number of iterations.

Conclusions

A global optimization problem is formulated by choosing an objective function and a set of constraints which properly model the process or system to be optimized. In this way, a very general type of global optimization problem can arise which will have one of five possible outcomes (see Section 3.2.1). The use of an automaton modifies the set of possible outcomes, resulting in four possible outcomes.

The use of the automaton also introduces various types of error and thus a global optimization system must be able to validly identify which of the possible outcomes has occurred, ideally it would never return an INDETERMINATE outcome. It must do so within the finite restrictions on space and time that result from use of an automaton and other modelling considerations. Wherever possible, a global optimization system should take advantage of the simplification introduced by the automaton. Thus while the use of the automaton has disadvantages it also has advantages that are useful from a practical perspective.

4.3.3 Formulation Aids for a Global Optimization System

As previously noted (see Section 1.0.2), means by which the characteristics of the problem space as well as the outcome of the solution process can be obtained are desirable from a modelling perspective. In this section we examine the formulation aids of output reports on solution accuracy, causal information (see Section 1.0.2), objective function 'flatness' and sensitivity analysis in the context of a global optimization system.

Solution Accuracy

Due to the presence of errors, a measure of the accuracy of the incumbent solution is required so that the modeller can determine the 'appropriateness' of the incumbent solution. In

addition this measure of accuracy can be used to determine when to stop the solution process. Ideally, a lower bound on the global minimum is sought, with the difference between the lower bound and the incumbent solution providing a measure of accuracy. Other measures based on a statistical analysis are possible, but are by their nature less robust. Thus for a GLOBAL MINIMUM outcome, it is desirable to generate a lower bound on the global minimum as well as the incumbent solution.

Causal Information

For INFEASIBLE (and DOMAIN UNBOUNDED) outcomes, the modeller may wish to know what has 'caused' the infeasibility (unboundedness) in order to determine if the cause is a modelling or syntactic error or due to the structure of the system being optimized. This information is used to improve the model.

We now examine if and how causal information can be obtained for INFEASIBLE and DOMAIN UNBOUNDED outcomes.

- **INFEASIBLE** - A problem is infeasible if and only if a set of inconsistent constraints exists. Thus, causal information can be presented in the form of a set of inconsistent constraints. Ideally a minimally inconsistent set is desirable as it provides a localization of the 'cause' of the infeasibility.
- **DOMAIN UNBOUNDED** - A problem is domain unbounded if and only if an unbounded domain point (feasible and unbounded from below) exists. Thus causal information can be provided by outputting such a point. If such a point can not be found, then a region which can be shown to include such a point is a possible alternative. This may be necessary as the exact unbounded domain point may not be representable on the automaton.

Thus, causal information can be provided for infeasible and domain unbounded outcomes.

Objective Function 'Flatness'

Besides the need for causal information of the outcome, the 'flatness' of the region (see Section 7.9.1) around a global minimum (if it exists) may be of interest for the modeller as a solution that is in a steep region may not be desirable, due to the fact that the model

or problem parameters (coefficients and constants within the functions and constraints) are estimates and not exact values.

By the *flatness* of the objective function about a point, x , we mean the change in functional values about some region, centered at x , normalized by the 'size' of the region. This notion can be extended to the entire problem space, as a measure of the flatness of the problem space, or to some other specified region by varying the size of the region about a point. Thus, the flatness can be used to examine the behaviour of the function about a region of some specified size. It would be desirable to be able to vary the size and location of the region.

Sensitivity Analysis

A small change in a parameter may drastically alter the solution point. Thus the sensitivity of a problem to perturbations in the problem parameters is desirable.

Summary

In summary, it is desirable for a global optimization system to provide formulation aids, such as causal information, flatness of the objective function and sensitivity analysis as they provide information by which the modeller can determine the veracity and 'appropriateness' of the outcome as well as information useful for modifying the existing model in light of the 'appropriateness' of the outcome.

4.3.4 Definition of a Global Optimization System

We formally define a global optimization system (GOS) as a methodology for the formulation and solution of global optimization problems such that the following conditions are fulfilled:

- The system requires little, if any, *a priori* information on the characteristics of the problem to be solved.
- The system must be able to validly determine if the problem outcome is INFEASIBLE, DOMAIN UNBOUNDED or GLOBAL MINIMUM.
- The system must address the problems that arise due to the use of the automaton, specifically the issues of: FINITENESS, DILEMMA OF EQUALITY, FUNCTION REPRESENTATION.

SENTATION & RESTRICTED FUNCTION DOMAINS.

- The system provides a measure of the accuracy of the incumbent solution for global minimum outcomes.
- Causal information is provided for infeasible or domain unbounded outcomes.
- The system provides additional formulation aids, such as flatness of the objective function and sensitivity analysis which can be used optionally.
- The system must complete its task within a reasonable amount of time and it must not exceed the space capacity of the automaton.

Discussion

A global optimization problem arises from an application of mathematical modelling to solve some problem in engineering, economics, or other profession and is then solved by an implementation of a global optimization method. In general, information on the characteristics of the problem will vary or may not be known *a priori*. In addition, the complexity of the problem will require the use of an automaton in order for a solution to be obtained.

The modeller must then determine the veracity of the outcome as the errors present in the solution process (due to the use of the automaton) are both numerical and logical in nature.

If the outcome is INFEASIBLE, then the set of constraints is inconsistent. Therefore there exists at least one minimally inconsistent set of constraints. A minimally inconsistent constraint set causes infeasibility, is useful in determining if the model needs to be revised, and is desirable because each element of the set is relevant. The 'best' answer would be to identify all of the minimally inconsistent sets of constraints. A 'good' answer would be to identify a minimally inconsistent set of constraints.

If the outcome is DOMAIN UNBOUNDED then there is a domain unbounded point. As the domain unbounded point causes unboundedness it can be used by the modeller to determine if the model needs to be revised. The 'best' answer is $M_{\nabla, \Delta}(x_u^*)$ (where $M_{\nabla, \Delta}(x)$ finds the largest machine representable number less than x and the smallest machine representable number larger than x). A 'good' answer would be to find a solution set, S , such that the 'size' of S is within some *a priori* defined limits and that $M_{\nabla, \Delta}(x_u^*) \subseteq S$. We note that

due to the finite nature of the automaton it may be impossible to exactly identify a domain unbounded point.

If the outcome is GLOBAL MINIMUM, then there exists a global minimum, x^* , to the Global Optimization Problem. The 'best' answer is an incumbent solution, x^* , within a region $M_{\nabla,\Delta}(x^*)$, and an accompanying lower bound, Lb, of value $\nabla(f(x^*))$ (where $\nabla(x)$ returns the largest machine representable number smaller than x). A 'good' answer would be to find an incumbent solution, x^* , a solution set, S , and a lower bound, Lb, such that the 'size' of S is within some *a priori* defined limits and that $M_{\nabla,\Delta}(x^*) \subseteq S$. Lb and x^* can then be used to provide the accuracy of the incumbent solution. We note that due to the finite nature of the automaton it may be impossible to exactly identify a global minimum.

Thus the following output is sought to satisfy the requirements for causal information for infeasible and unbounded outcomes and the accuracy of the incumbent solution:

- INFEASIBLE - A minimally inconsistent set of constraints.
- DOMAIN UNBOUNDED - A set of points and/or regions (of *a priori* determined size) which contain a domain unbounded point.
- GLOBAL MINIMUM - An incumbent solution, a lower bound on the global minimum and a set of regions (of 'sufficiently' small size) which contain a global minimum.

There may be several iterations of revising the model and evaluating the outcome, so validation mechanisms, flatness and sensitivity analysis are desirable from a modelling perspective because they help the modeller to analyze the current model for its 'appropriateness'. Finally, the global optimization system should be efficient in its time and space requirements. The more efficient the system, the larger the problems that can be solved.

4.3.5 Testing a Global Optimization System

There are two main aspects that need to be examined in determining the 'usability' of a global optimization system, that of validity and 'practicality'. By validity, we mean whether the GOS produces correct results in terms of the type of outcome, the correctness of the formulation information and the correctness of the methods used to account for the use of the automaton. In other words, does the GOS meet its claims? By practicality we mean a measure of the performance of the method, which involves a determination of the types

and 'sizes' of problems that can be solved as well as a comparison (when applicable) against existing methods.

Determination of the validity of a GOS

In order to test the validity of a GOS one needs to examine how a given method meets the above requirements. For most aspects of the GOS an examination of the rules and/or procedures used will suffice. However, empirical testing of an implementation of a GOS is required in order to determine that it can indeed validly determine the outcome of a given problem as well as provide 'useful' formulation information.

For this purpose, a test set of known characteristics, which covers the range of possible outcome and constraint types must be used. Because the current known test sets all have global minimum outcomes, infeasible and unbounded problems must be added. Random test problems could be used, but due to the vast range of possible global optimization problems, it is better to establish a good representative set of test problems. In addition, most global optimization problems will exhibit a fair amount of structure, due to the nature of the source of these problems.

Empirical testing is dependent upon the test problems, so care must be taken in choosing the test problem set. As we are interested in designing a GOS for general use by modellers, the test problem set must be comprehensive in nature.

Determination of the 'practicality' of a GOS

In order to determine the types and 'sizes' of problems that a GOS can solve, and thus determine its 'practicality', some measure of the method's performance must be determined. Ideally this measure would depend on the size and characteristics of the problem to be solved. Both absolute and relative measures are available for the evaluation of optimization methods.

Examples of absolute measures are worst case and average case analysis of the computational complexity of the method. The worst case analysis is usually easier to determine, but in many cases it does not properly reflect the practical nature of the method. For example, for linear programming, the simplex algorithm is at worst case an exponential time algorithm while the ellipsoid algorithm is polynomial. However, the simplex algorithm is the more practical algorithm of the two. The average case analysis will give a more accurate

picture, but is more difficult to determine, as it requires information on what constitutes an average problem. This is difficult to determine when the range of possible problems is extensive and varied, which is the case for global optimization problems.

Thus for global optimization, absolute measures are likely to be ineffective due to the fact that the range of possible problems is very large, even for restricted classes of problems. Furthermore, due to the large range of possible problems, the practicality of a method depends not only on the method, but on the particular problem that it is to be applied to as well.

Relative measures are used in comparing various methods against a set of representative test problems. A good set of test problems that contains many types of problems can be used to indicate those types of problems for which a given method is practical and those for which it is not.

The most common relative measures used for global optimization problems are as follows (see for example [48, 100, 115]):

- the amount of time required.
- the amount of space required.
- the number of objective function (constraint) evaluations.
- the correctness of the solution and/or outcome.
- the number of iterations required.
- the type of problems (and thus outcomes) that can be solved.

We will measure the practicality of a global optimization system by comparing it with the other known methods (when applicable) relative to the above set of measures against a set of test problems and by its performance against a set of new problems which will test its ability to identify and provide causal information for infeasible and unbounded problems.

4.4 Problem Statement

The problem that is addressed in this dissertation is that of developing a Global Optimization System, that is, a methodology for the formulation and solution of Global Optimization Problems.

Specifically, we want to develop a methodology that does the following:

- provides for the solution of global optimization problems by verifying that exactly one of the following outcomes has occurred :
 - GLOBAL MINIMUM - The solution set consists of an incumbent solution, a lower bound for the global minimum and a set of regions guaranteed to contain at least one global minimizer.
 - INFEASIBLE - The solution set consists of a minimally inconsistent set of constraints.
 - DOMAIN UNBOUNDED - The solution set consists of a set of points and/or regions, which contain at least one point for which the objective function is domain unbounded.
 - INDETERMINATE- A set of *promising* sub-regions and/or a set of starting points for local optimization methods, etc., are identified and/or the difficulty (in determining the outcome) has been localized to a set of small subregions.

For GLOBAL MINIMUM and DOMAIN UNBOUNDED outcomes, the size of the regions in the solution set is either within some *a priori* specified limits or as small as possible subject to the time and/or space restrictions imposed by the automaton and/or user restrictions.

- It provides formulation aids such as flatness of the objective function and sensitivity analysis.
- It addresses the problems that arise due to the use of the automaton which is required for the solution of practical global optimization problems, specifically the issues of: FINITENESS, DILEMMA OF EQUALITY, FUNCTION REPRESENTATION, and RESTRICTED FUNCTION DOMAINS.
- It can be implemented on an automaton such that the space capacity of the automaton will not be exceeded during the execution and that it will complete the task within a reasonable amount of time.

Motivational Summary

The state of the art in global optimization methods does not address the provision of information for the evaluation of the correctness and appropriateness of a solution to a global optimization problem. As such information is useful from a modelling perspective, a methodology which addresses these issues is desirable. Thus the goal of the dissertation is to develop a global optimization system which will provide the modeller with formulation and modelling information for the solution of global optimization problems.

Chapter 5

A Practical Global Optimization Test Set

5.1 Introduction

In this Chapter we evaluate the set of test problems (see Appendix B) to be used to evaluate the global optimization system developed in the dissertation. The set is comprised of problems from existing sources as well as new test problems developed for cases where no existing problems have been found.

5.2 Evaluation of Test Set

The test set has been chosen in order to span the possible outcomes of a Global Optimization Problem as well as the possible configurations of these outcomes in relation to the classes of constraints that are possible.

Thus, problems with DOMAIN UNBOUNDED (see Appendix B.2.1) and INFEASIBLE (see Appendix B.2.2) outcomes have been developed for a complete listing of the test set). The DOMAIN UNBOUNDED problems have been developed so that the unboundedness occurs

at a single point such that:

- a) the point is the only feasible point (Unbounded 2 cs2).
- b) the feasible region is unconstrained.

on the points of a 'surface' in R^n such that:

- a) the surface is only feasible at a single point (Unbounded 8 con21).
- b) the feasible region is unconstrained (Unbounded 7).

on multiple 'surfaces' (Unbounded 5 and Unbounded 6).

In addition, two problems with infeasible unbounded from below points have been developed (Unbounded 2 mod & Unbounded 2 mod2), each with an infeasible unbounded from below point near to the feasible region. As well one problem (Unbounded 2 cs2 mod) has been developed which has an unbounded from below point in an INFEASIBLE problem.

For the INFEASIBLE test set, problems with single IIS's (with either no extra constraints or with other constraints) and multiple IIS's have been developed. One of these (constraint set 13) contains only equality constraints.

Existing GLOBAL MINIMUM problems (see Appendix B.1) were chosen to include unconstrained problems (Schwefel No.'s 2.1 & 3.1), variable bounded problems (Six Hump Back, Three Hump Camel) and constrained problems with various numbers of equality and inequality constraints.

In addition to existing GLOBAL MINIMUM problems, nine new problems (see Appendix B.2.3) have been developed: Six Hump Back cs1, Six Hump Back cs11, Three Hump Camel cs3, Three Hump Camel cs7, Melville Shipping Ltd., Variation on Rosenbrock, Variation on Rosenbrock cs16, Variation on Rosenbrock cs17, and Example No. 1.

5.2.1 Problem Sizes

A list of all of the test problems is given in Tables 5.1, 5.2, and 5.3, indicating their size in terms of the number of variables, constraints and variable bounds, as well as their outcome type. UB denotes a problem from the Unbounded Test Set (see Appendix B.2.1), FP denotes a problem from the Floudas and Pardalos Test Set (see Appendix B.1.1), SCH denotes a problem from the Schwefel Test Set (see Appendix B.1.2). SHB, THC, and SDP are the Six Hump Back, Three Hump Camel, and Spring Design problems (see Appendix B.1.3), and VR, Ex. 1 and MSP are the Variation on Rosenbrock, Example No. 1, and Melville Shipping Ltd. problems (see Appendix B.2.3). In addition to the problem test set, a set of constraints have also been developed. Their sizes and feasibility are given in Table 5.4.

Problem	No. of			Outcome
	variables	constraints	variable bounds	
UB 2 cs2 mod	2	3	1	INFEASIBLE
UB 8 cs8	5	2	0	INFEASIBLE
FP 3.2 cs5	5	5	0	INFEASIBLE
SHB cs10	2	6	0	INFEASIBLE
THC cs12	2	18	0	INFEASIBLE

Table 5.1: Problem Test Set : Sizes and Outcomes : INFEASIBLE

Problem	No. of			Outcome
	variables	constraints	variable bounds	
UB 1	1	0	0	DOMAIN UNBOUNDED
UB 2	2	0	0	DOMAIN UNBOUNDED
UB 5	2	0	0	DOMAIN UNBOUNDED
UB 6	2	0	0	DOMAIN UNBOUNDED
UB 7	2	0	0	DOMAIN UNBOUNDED
UB 8	5	0	0	DOMAIN UNBOUNDED
UB 2 cs2	2	3	0	DOMAIN UNBOUNDED
UB 8 con21	5	1	0	DOMAIN UNBOUNDED

Table 5.2: Problem Test Set : Sizes and Outcomes : DOMAIN UNBOUNDED

5.2.2 Conclusion

The test set spans the outcome types of INFEASIBLE, DOMAIN UNBOUNDED and GLOBAL MINIMUM. For each outcome type, several different types of problems are present, in order to attempt to span the types of problems in each outcome class. In addition, several problems are 'close' to becoming a different outcome type, for example, Unbounded 2 cs2 contains only one feasible point and thus is close to being of the INFEASIBLE outcome.

The test set is limited in that it predominately contains small problems, partially due to the difficulty in analyzing larger problems. However, it is broad enough to provide a means

Problem	No. of			Outcome
	variables	constraints	variable bounds	
Ex. 1	1	2	2	GLOBAL MINIMUM
UB 3	1	0	0	GLOBAL MINIMUM
UB 4	1	0	0	GLOBAL MINIMUM
FP 3.1	8	6	16	GLOBAL MINIMUM
FP 3.2	5	6	10	GLOBAL MINIMUM
FP 4.6	2	2	4	GLOBAL MINIMUM
FP 4.7	2	1	4	GLOBAL MINIMUM
FP 4.7 - SUB	1	0	2	GLOBAL MINIMUM
FP 6.2	6	6	2	GLOBAL MINIMUM
FP 11.1 - Mod	50	49	100	GLOBAL MINIMUM
SCH 2.1	2	0	0	GLOBAL MINIMUM
SCH 3.1	3	0	0	GLOBAL MINIMUM
SHB	2	0	4	GLOBAL MINIMUM
THC	2	0	4	GLOBAL MINIMUM
Zhang 1	2	2	4	GLOBAL MINIMUM
SHB cs1	2	3	0	GLOBAL MINIMUM
SHB cs11	2	3	0	GLOBAL MINIMUM
THC cs3	2	2	2	GLOBAL MINIMUM
THC cs7	2	3	2	GLOBAL MINIMUM
UB 2 mod	2	0	1	GLOBAL MINIMUM
UB 2 mod2	2	0	1	GLOBAL MINIMUM
MSP	3	0	6	GLOBAL MINIMUM
SDP	4	8	8	GLOBAL MINIMUM
VR	50	0	100	GLOBAL MINIMUM
VR	50	25	100	GLOBAL MINIMUM
VR	50	50	100	GLOBAL MINIMUM

Table 5.3: Problem Test Set : Sizes and Outcomes : GLOBAL MINIMUM

Problem	No. of			Feasibility
	variables	constraints	variable bounds	
cs1	2	3	0	feasible
cs2	2	3	0	feasible
cs3	2	2	2	feasible
cs4	5	4	0	infeasible
cs5	5	5	0	infeasible
cs6	2	4	4	infeasible
cs7	2	3	2	feasible
cs8	5	2	0	infeasible
cs9	2	6	0	infeasible
cs10	2	6	0	infeasible
cs11	2	3	0	feasible
cs12	2	18	0	infeasible
cs13	2	3	0	infeasible
cs14	2	4	0	infeasible
cs15	6	5	0	infeasible
cs16	50	25	0	feasible
cs17	50	50	0	feasible

Table 5.4: Sets of Constraints : Size and Feasibility

by which the effectiveness of GOS can be evaluated.

Chapter 6

Improved Bounding Functions

Improvements to the accuracy of bounding functions are important from a practical perspective as an improvement on a bound (for the objective function and/or the constraint functions) can improve the performance of the solution process.

In this Chapter we extend both Krawczyk's centered form and classes of bounding functions to functions with logical connectives and we formalize a modification to Krawczyk's centered form in order to improve its accuracy and computational cost for function representations with a particular structure.

6.1 Extensions to Functions with Logical Connectives

In this section we examine how classes of bounding functions and Krawczyk's centered form can be extended to functions with logical connectives.

6.1.1 Bounding Functions with Logical Connectives

In this section we examine how classes of bounding functions can be applied to functions with logical connectives. Ratschek and Rokne [97, pages 37-38] state that a bounding function for any codeable function that uses logical connectives can be found. They did not attempt to prove this or to develop a general method to create these bounding functions, but merely stated that they have failed to find a counter example. Moore [84] gives a method for bounding functions of one variable. Moore's method has been extended to functions with more than one variable.

We shall now develop a methodology for the creation of an inclusion function for a codeable function with logical connectives under the condition that a bounding function exists for each of the *domain components* of the function. This will formalize the claim of Ratschek and Rokne. A function with logical connectives will partition the set of domain points into disjoint sets, such that a sub-function is applicable to each such set. We call each such subset of domain points a *domain component* and the function that corresponds to the domain component as a *sub-function*.

Let us examine the following function that makes use of a logical connective.

$$f(x) = \begin{cases} x^2 & \text{if } x \geq 0 \\ -1 & \text{otherwise} \end{cases}$$

This function is Lipschitz over any finite interval, but discontinuous at $x = 0$, with domain components of $\{x|x \geq 0\}$ and $\{x|x < 0\}$ and corresponding sub-functions of x^2 and -1 . If one naively applies the natural interval extension to this function one gets the following interval function.

$$F(X) = \begin{cases} X^2 & \text{if } X \geq 0 \\ [-1, -1] & \text{otherwise} \end{cases}$$

If $X = [2, 4]$ then as $X \geq 0$, $F(X) = [4, 16]$, which is true, however, if $X = [-1, 1]$ (or any X which straddles the discontinuity) then $X \not\geq 0$ and thus $F(X) = [-1, -1]$, which is false. Thus our function $F(X)$ is not a bounding function. Thus while this approach works for intervals contained within a component, it fails for intervals which straddle domain components. Thus what is required is a method for intervals which straddle one or more domain components.

Consider an interval X , such that X straddles n domain components of the function $f : R^n \rightarrow R$. The approach taken is to partition the interval, X , into sets of sub-intervals, SI_i , such that the i^{th} set of sub-intervals is contained in the elements of the i^{th} domain component, d_i , of the function f . Each interval in the set of sub-intervals is then applied to the corresponding bounding function of the sub-function, f_i , to obtain a set of bounds. The subsequent union of these bounds is a bound for the function over the interval X .

Our goal is to find a bounding function, $F(X)$, for some given interval X .

Let $SI_i(X) = \{B_j \mid B_j \in I^n, B_j \subseteq X, \cup B_j \supseteq (A \cap d_i)\}$, that is, a set of intervals, B_j , such that the union of the intervals is a) contained within the interval A and b) contains all of the elements of d_i .

Let $F(SI_i(X)) = \bigcup_{B_j \in SI_i(X)} F_i(B_j)$, that is, the union of the bounds resultant from applying the bounding function F_i over the set of intervals, B_j , of the set $SI_i(X)$.

Then $F(X) = \bigcup_i F(SI_i(X))$ is a bounding function for $f(x)$ if $\forall i, j$ $F_i(B_j)$ exists.

If a domain component, d_i is not an interval or is not equal to the union of a finite set of intervals, then a problem arises if the sub-function has an implicit constraint. For example, consider the following function:

$$f(x) = \begin{cases} \sqrt{x_1^2 + x_2^2 + 1} & \text{if } x_1^2 + x_2^2 \leq 1 & : d_1 \\ -1 & \text{otherwise} & : d_2 \end{cases}$$

In the above example, we see that any finite set SI_i of intervals will contain some elements which are elements of d_2 . The elements of a set SI_i which are not elements of the domain component, d_i , make up the *excess region* of SI_i . The amount of excess region in a set SI_i can be reduced if smaller intervals are used near the boundary of d_i . This has two effects. First, since smaller intervals are used, the accuracy of the bounding functions used will increase and second, the cost of applying a bounding function to the intervals of the set of intervals increases. Thus a balance needs to be determined between the amount of excess region allowed and the number of intervals to be used in the set.

If the sub-function of a domain component has an implicit constraint, such as for the above example where the square root requires that its argument is greater than or equal to zero, then problems may arise. In the above example, $x_1^2 + x_2^2 + 1 \geq 0$ for any x_1 and x_2 , but if the argument was instead $1 - x_1^2 - x_2^2$ then points such as (0.4, 0.92) (which is an element of Sets 1 and 2) cause problems as the function is not defined for these points and thus the bounding function for the function is not defined for intervals which contain these points. This is a problem of restricted function domains and some methods by which a bounding function may be definable for such intervals are discussed in Section 7.1.4. For codeable functions, the extra 'non-defined' region can be added to the appropriate domain.

Example 6.1.1

Consider the following function:

$$f(x) = \begin{cases} x_1^2 - x_2^2 & \text{if } x_1^2 + x_2^2 \leq 1 & : d_1 \\ x_1^3 + x_2 & x_2 \geq 2 & : d_2 \\ -1 & \text{otherwise} & : d_3 \end{cases}$$

with			
i	d_i	f_i	F_i
1	$\{x \mid x_1^2 + x_2^2 \leq 1\}$	$x_1^2 - x_2^2$	$X_1^2 - X_2^2$
2	$\{x \mid x_2 \geq 2\}$	$x_1^3 + x_2$	$X_1^3 + X_2$
3	$\{x \mid x_2 < 2, x_1^2 + x_2^2 > 1\}$	-1	$[-1, -1]$

Figure 6.1: domain components of $f(x)$ for intervals A and B

and the following intervals, $A = ([-2, 0], [0, 3])$ and $B = ([2, 3], [1, 3])$.

For interval A (See Figure 6.1):

We generate $SI_1 = \{([-0.8661, 0], [0, 0.5]), ([-0.8661, 0], [0.5, 1.0]),$
 $([-1, -0.8661], [0, 0.5])\}$, $SI_2 = \{([-2, 0], [2, 3])\}$ and $SI_3 = \{([-2, 0], [0, 2])\}$.

Thus $F(SI_1) = ([-0.8661, 0]^2 - [0, 0.5]^2) \cup ([-0.8661, 0]^2 - [0.5, 1.0]^2) \cup$
 $([-1, -0.8661]^2 - [0, 0.5]^2) = [-0.25, 0.7502] \cup [-1, 0.5002] \cup [0.7502, 1] = [-1, 1]$,
 $F(SI_2) = [-2, 0]^3 + [2, 3] = [-6, 3]$ and $F(SI_3) = [-1, -1]$ and thus $F(A) = [-6, 3]$.

For interval B (See Figure 6.1):

We generate $SI_1 = \emptyset$, $SI_2 = \{([2, 3], [2, 3])\}$ and $SI_3 = \{([2, 3], [1, 2])\}$.

Thus $F(SI_1) = \emptyset$, $F(SI_2) = [2, 3]^3 + [2, 3] = [10, 30]$ and $F(SI_3) = [-1, -1]$ and thus
 $F(B) = [-1, 30]$.

Conclusion

The above method can be used to provide a bounding function for a function with logical connectives under the conditions that bounding functions exist for the sub-functions, f_i , over the domain of the corresponding sets Sf_i .

This means that it can be used with any bounding function method if all of the logical connectives are at the top level of the function definition. For example, consider the following function:

$$f(x) = \begin{cases} g(x) & \text{if } x^2 \leq 1 \\ -1 & \text{otherwise} \end{cases}$$

where

$$g(x) = \begin{cases} x_1 + x_2^2 & \text{if } x_1 + x_2 \leq 0 \\ x_1^2 + x_2 & \text{otherwise} \end{cases}$$

For such a function, the above method could possibly be used recursively, except that the bounding function method used may involve procedures which are not valid or defined for such a method. For example, Krawczyk's centered form requires that a slope function is created for each element of the function procedure and thus if $f(x)$ or $g(x)$ is part of such a function procedure then a slope function for $f(x)$ or $g(x)$ is required. How such a slope function is to be generated is not defined by the above method. If all of the logical connectives are propagated to the top level as follows then Krawczyk's centered form could be used to determine a bound for each of the sub-functions and the final bound determined by a union of the bounds of the appropriate sub-functions. For $f(x)$, propagation of the logical connectives to the top level results in:

$$f(x) = \begin{cases} x_1 + x_2^2 & \text{if } x_1 + x_2 \leq 0 \text{ and } x^2 \leq 1 \\ x_1^2 + x_2 & \text{if } x_1 + x_2 > 0 \text{ and } x^2 \leq 1 \\ -1 & \text{otherwise} \end{cases}$$

In addition, the recursive approach will likely be less accurate. For example, consider the interval $A = ([-2, 2], [-2, 2])$. Recursively we would determine a bound for $g(x)$ over A and then use this value within $f(x)$. However, the bound generated for $g(x)$ will contain a large excess region relative to the $g(x)$ component of $f(x)$ and will thus, in general, decrease the

accuracy of the final bound generated for $f(x)$. Therefore it is usually better to propagate the logical connectives to the top level.

Thus we define the *propagating bounding function extension* method as applied to a function $f : R^n \rightarrow R$ over an interval A to determine a bound on the range of f over A , $F(A)$, as follows:

- 1 · Propagate all logical connectives to the top level to generate the propagated version of f , f_p .
- 2 · Identify the domain components, d_i , and sub-functions, f_i , of the propagated function, f_p .
- 3 · Generate the sets, $SI_i(A)$, for each d_i .
- 4 · For each set, $SI_i(A)$, determine $F(SI_i)$.
- 5 · $F(A) = \bigcup_i F(SI_i)$.

6.1.2 Krawczyk Centered Form

Krawczyk's centered forms are applicable for rational functions which include unary functions as well as Lipschitz continuous functions. We extend them to functions with logical connectives by a method similar to the method of the previous section, using Krawczyk's centered form to bound the domain components determined by the logical connectives. Once this has been done, the slope can be bound by using the appropriate upper and lower bounds of the domain components which span the region to be bounded and the bounds that define the region.

As these slopes are determined from the point at which the Krawczyk centered form is to be expanded it is desirable that the expansion point is not on the 'border' of the domain components, as otherwise an infinite slope will be obtained. The process is further complicated as the calculation of the Krawczyk centered forms for each of the domain components is required, and as the expansion point is only an element of one of these domain components. For reasons of accuracy it is desirable that expansion points are identified for each of the other domain components. This can be done by taking the midpoint of the region which is used to determine the bounds for a domain component.

6.1.3 Example

For example, consider the following function:

$$f(x) = (4x + 3)h(x)$$

where

$$h(x) = \begin{cases} 2x & \text{if } x < -1 & d_1 \\ x^2 & \text{if } -1 \leq x \leq 2 & d_2 \\ 3x - 2 & \text{if } 2 < x & d_3 \end{cases}$$

Let a function procedure for $f(x)$ be provided by the following sequence of functions, u_i . We calculate Krawczyk's Centered Form with interval slope functions, G_i and inclusion functions F_i as follows (See Section 2.4.2):

i	f_i	u_i	G_i	F_i
1	x	x	1	X
2	$h(x)$	$h(x)$	G_h	$H(X)$
3	3	3	0	$[3, 3]$
4	4	4	0	$[4, 4]$
5	$4x$	$u_4 * u_1$	$4 * G_1 = 4$	$f_5(c) + G_5(X - c)$
6	$4x + 3$	$u_5 + u_3$	$G_5 + G_3 = 4$	$f_6(c) + G_6(X - c)$
7	$(4x + 3)h(x)$	$u_6 * u_2$	$G_6 * F_2 + G_2 * f_5(c)$	$f_7(c) + G_7(X - c)$

Consider the interval $X = [-2, 3]$. Thus $c = 0.5$ and $X - c = [-2.5, 2.5]$.

We need to determine $G_2 = G_h$ and $F_2 = H(X)$. We do so by using the method of Section 6.1.1 as follows:

$$H_1(X) = 2X_{h_1}$$

$$= 2[-2, -1]$$

$$H_2(X) = c_{h_2}^2 + G_{H_2}(X_{h_2} - c_{h_2})$$

$$= 0.5^2 + (0.5 + [-1, 2])([-1, 2] - 0.5)$$

$$= 0.25 + ([-0.5, 2.5])([-1.5, 1.5])$$

$$H_3(X) = 3X_{h_3} - 2$$

$$\begin{aligned}
&= 3[2, 3] - 2 \\
F_2 &= 2[-2, -1] \cup 0.5^2 + (0.5 + [-1, 2])([-1, 2] - 0.5) \cup 3[2, 3] - 2 \\
&= [-4, -2] \cup [-3.5, 4] \cup [4, 7] \\
&= [-4, 7] \\
G_2 &= [lb(G_2), ub(G_2)] \\
lb(G_2) &= \min\left(\frac{ub(H_1(X)) - h_2(c)^2}{lb(X_{h_1}) - c}, -0.5, \frac{lb(H_3(X)) - h_2(c)^2}{ub(X_{h_3}) - c}\right) \\
&= \min\left(\frac{-2 - 0.25}{-2 - 0.5}, -0.5, \frac{4 - 0.25}{3 - 0.5}\right) \\
&= \min(0.9, -0.5, 15.1) \\
&= -0.5 \\
ub(G_2) &= \max\left(\frac{lb(H_1(X)) - h_2(c)^2}{ub(X_{h_1}) - c}, 2.5, \frac{ub(H_3(X)) - h_2(c)^2}{lb(X_{h_3}) - c}\right) \\
&= \max\left(\frac{-4 - 0.25}{-1 - 0.5}, 2.5, \frac{7 - 0.25}{2 - 0.5}\right) \\
&= \max(2.834, 2.5, 1.5) \\
&= 2.834
\end{aligned}$$

where X_{h_i} indicates the portion of X in the i^{th} domain component of $h(x)$. The midpoint of each X_{h_i} , c_{h_i} is used as the midpoint for the Krawczyk centered form for that domain component. As $c \in X_{h_2}$ we determine the slopes by the appropriate choice of rise and run using the range of the domain components and the distance from c to the appropriate boundary of X for each domain component for which c is not an element of, in this case, d_1 and d_3 . The bound for the slope in the domain component for which c is an element is already determined by Krawczyk's centered form. By taking the minimum or maximum of the appropriate slopes, one can determine bounds for the slope function for $h(x)$ for a given X .

Now we can calculate the bound on $f(x)$ by plugging in the appropriate values to get a bound of $[-125, 127.5]$. If we had used the propagated bounding function extension method of Section 6.1.1, using Krawczyk's centered form to determine a bounding function for each sub-function, we get a result of $[-12, 67]$.

The method can be extended to functions with more than one variable, although the determination of the slope function becomes complicated for complex domain components.

6.1.4 Conclusion

While this is a valid method to determine the range of a function over some region, it is probably better to use the propagated bounding function extension method of Section 6.1.1 using a Krawczyk centered form on each domain component. This follows as the union of the domain component values is used to determine the overall slope and will determine in general a larger slope than is generated individually for each domain component. This larger overall slope will then be propagated through to the final expression whereas if the logical connectives are propagated to the top level, then the smaller slopes will be used for each domain component and they will thus, in general, provide a more accurate bound.

6.2 Modified Krawczyk Centered Form

Krawczyk's Centered Form [65] is based on parsing a function into a function procedure and then building the function up again using interval arithmetic techniques to obtain a bound for the original function.

We recall the definition of Krawczyk's Centered Form (see Section 2.4.2). Let $e_i \in I^n$ denote the i^{th} unit vector, $0 \in R^n$ denote the null vector and $c = m(X)$ (or any point in X). Let S be some given function procedure. Then the interval slope $G = G_s \in I^n$ of S over X is recursively defined by

- $G_i = e_i$ for $i = 1, \dots, n$
- $G_i = 0$ for $i = n + 1, \dots, n + t$
- and for $i = n + t + 1, \dots, s$ one has

$$\begin{aligned} G_i &= G_j \pm G_k \text{ if } u_i := u_j \pm u_k \\ G_i &= G_j F_k(X) + G_k f_j(c) \text{ if } u_i := u_j \times u_k \\ G_i &= \frac{G_j}{F_k(X)} - \frac{G_k f_j(c)}{F_k(X) f_k(c)} \text{ if } u_i := u_j \div u_k \end{aligned}$$

Let f be a function with a function procedure S , G the interval slope of S over $X \in I^m$ and let $c \in X$. Then the interval $f_K(X) = f(c) + G \cdot (X - c)$ is called Krawczyk's Centered Form.

A simple modification to Krawczyk's algorithm comes about if for a given u_i an exact bound for F_i , F_i^e is known and/or an exact bound for G_i , G_i^e , is known. The modification is to use the exact expression (F_i^e or G_i^e) in place of the one generated by Krawczyk's algorithm. The bound for the final expression of the modified Krawczyk algorithm is contained within the bound of the final expression for the unmodified Krawczyk algorithm (for identical function procedures, etc.). The modification for the F_i 's is hinted at in [65].

This modification is used primarily for functions of the form $u_i = u_j \times u_k$, or for linear functions such as $u_j + u_k$ or $u_j * u_k$ where u_k is a constant.

Validity of the Modified Krawczyk Centered Form

Consider a function procedure, S , with a function, u_i , for which an exact bound, F_i^e , is known for F_i . Let F_i^K be the bound determined by the Krawczyk centered form method. Let F_K be the bound for the final expression that is generated by the Krawczyk centered form method. Let the exact F_i be substituted in place of the one generated by the Krawczyk centered form method and let F_{MK} be the final expression that is generated by the modified Krawczyk centered form method.

The only conditions that the Krawczyk centered form method places on the bounds that it generates for each function, u_i , of the function procedure is that the bounds are correct. Thus the use of the exact bound does not invalidate the final expression. As $F_i^e \subseteq F_i^K$ and as Krawczyk's centered form is *inclusion isotonic* (as it uses the interval arithmetic operations and/or unary functions which are inclusion isotonic) it follows that $F_{MK} \subseteq F_K$.

Thus, the modified Krawczyk centered form can only improve upon that of the Krawczyk centered form. A similar argument holds for an exact bound of the slope function, G_i , and is easily extended to the case for which more than one exact bound is known.

6.2.1 Evaluation

Example 6.2.1-1

Consider the following function:

$$f(x) = 4x^8 + 6x^4 - 12x - 10$$

This is the objective function from the problem Floudas and Pardalos 4.7 - SUB (see Appendix B.1.1). We determine a bound for this function in two ways, the first using

Krawczyk's centered form method and the second using the above modification, which we label as the modified Krawczyk centered form method.

We calculate Krawczyk's centered form as follows:

i	f_i	u_i	G_i	F_i
1	x	x	1	X
2	4	4	0	[4, 4]
3	6	6	0	[6, 6]
4	-10	-10	0	[-10, -10]
5	-12	-12	0	[-12, -12]
6	$x * x$	$u_1 * u_1$	$G_1 * F_1 + G_1 * f_1(c)$	$f_6(c) + g_6 * (X - c)$
7	$x^2 * x^2$	$u_6 * u_6$	$G_6 * F_6 + G_6 * f_6(c)$	$f_7(c) + g_7 * (X - c)$
8	$x^4 * x^4$	$u_7 * u_7$	$G_7 * F_7 + G_7 * f_7(c)$	
9	$-12 * x$	$u_5 * u_1$	$G_1 * f_5(c)$	
10	$6 * x^4$	$u_3 * u_7$	$G_7 * f_3(c)$	
11	$4 * x^8$	$u_2 * u_8$	$G_8 * f_2(c)$	
12	$4 * x^8 + 6 * x^4$	$u_{10} + u_{11}$	$G_{10} + G_{11}$	
13	$4 * x^8 + 6 * x^4 +$ $-12 * x$	$u_{12} + u_9$	G_{12}	
14	$4 * x^8 + 6 * x^4 +$ $-12 * x + -10$	$u_{13} + u_4$	G_{13}	$f_{14}(c) + g_{14} * (X - c)$

We calculate the modified Krawczyk centered form as follows:

i	f_i	u_i	G_i	F_i
1	x	x	1	X
2	4	4	0	[4, 4]
3	6	6	0	[6, 6]
4	-10	-10	0	[-10, -10]
5	-12	-12	0	[-12, -12]
6	$x * x$	$u_1 * u_1$	$[\frac{f_6(c) - f_6(lb(X))}{c - lb(X)}, \frac{f_6(ub(X)) - f_6(c)}{ub(X) - c}]$	X^2
7	$x^2 * x^2$	$u_6 * u_6$	$[\frac{f_7(c) - f_7(lb(X))}{c - lb(X)}, \frac{f_7(ub(X)) - f_7(c)}{ub(X) - c}]$	$(X^2)^2$
8	$x^4 * x^4$	$u_7 * u_7$	$[\frac{f_8(c) - f_8(lb(X))}{c - lb(X)}, \frac{f_8(ub(X)) - f_8(c)}{ub(X) - c}]$	

i	f_i	u_i	G_i	F_i
9	$-12 * x$	$u_5 * u_1$	$[-12, -12]$	
10	$6 * x^4$	$u_3 * u_7$	$6 * G_7$	
11	$4 * x^8$	$u_2 * u_8$	$4 * G_8$	
12	$4 * x^8 + 6 * x^4$	$u_{10} + u_{11}$	$G_{10} + G_{11}$	
13	$4 * x^8 + 6 * x^4 +$ $-12 * x$	$u_{12} + u_9$	G_{12}	
14	$4 * x^8 + 6 * x^4 +$ $-12 * x + -10$	$u_{13} + u_4$	G_{13}	$f_{14}(c) + G_{14} * (X - c)$

We note that for the above example, F_8 to F_{13} are not required in order to determine F_{14} and thus need not be determined. We test the two versions over the following Interval regions :

Interval	Modified	Unmodified	% improvement
$[-10, 10]$	$[-4.0006e8, 4.0006e8]$	$[-4.0006e8, 4.0006e8]$	0
$[-2.2223, -1.1111]$	$[-1952.91, 2541.80]$	$[-1952.91, 2541.80]$	0
$[0, 1.5]$	$[-138.293, 104.891]$	$[-138.293, 104.891]$	0
$[0, 1.25]$	$[-46.473, 13.491]$	$[-48.593, 15.610]$	4.4
$[0, 1]$	$[-21.21875, -10]$	$[-26.90625, -5.3125]$	21.1
$[0.6666666, 1]$	$[-20.35242, -11.99999]$	$[-20.35242, -11.99999]$	0
$[0.3333, 0.4445]$	$[-15.130, -13.925]$	$[-15.132, -13.923]$	0.01
$[0.8148, 0.8519]$	$[-16.3988, -15.9536]$	$[-16.3988, -15.9536]$	0
$[0.666, 0.679]$	$[-16.69399, -16.65874]$	$[-16.69400, -16.65873]$	0.00006

Table 6.1: Bound test of Modified vs Unmodified Krawczyk Centered Forms : 6.2.1-1

Thus we see that, for the given problem, the modified version improves upon the bound for intervals within the range 0 to about 1.5 and the larger % improvements (in the lower bound) occur for the larger interval size.

Example 6.2.1-2

Consider the function $f(x) = x_1 * (x_1 x_2)^2$. We calculate the Krawczyk centered form and the modified Krawczyk centered form using the following function procedure:

i	1	2	3	4	5
u_i	x_1	x_2	$u_1 * u_2$	$u_3 * u_3$	$u_1 * u_4$

over the following interval vector regions:

Interval	Modified	Unmodified	% improvement
$\{[-10, 10], [-10, 10]\}$	$[-100000, 100000]$	$[-100000, 100000]$	0
$\{[0, 10], [0, 10]\}$	$[-93750, 100000]$	$[-93750, 100000]$	0
$\{[0, 1.1111], [0, 1.1111]\}$	$[-1.5877, 1.6936]$	$[-1.5877, 1.6936]$	0
$\{[-1, 0], [0.3333, 1]\}$	$[-1, 0.88889]$	$[-1, 0.88889]$	0
$\{[-1, -0.6666], [-1, -0.3333]\}$	$[-1, 0.4856]$	$[-1, 0.4856]$	0
$\{[-1, -0.9629], [-1, -0.9753]\}$	$[-1, -0.8446]$	$[-1, 0.8446]$	0

Table 6.2: Bound test of Modified vs Unmodified Krawczyk Centered Forms : 6.2.1-2

Thus we see that, for Example 6.2.1-2, the modified version does not improve upon the bounds.

Performance Evaluation

This modification has two main effects. The first is that an improved bound may be generated and the second is that the amount of computation required is altered, usually reduced.

In practice, an improved bound is found only for 'large' boxes and does not occur for all large boxes. Small boxes show no or little improvement due to the fact that Krawczyk's centered form is quadratically convergent and thus the bounds generated converge quadratically to the exact bounds.

The computational effect is dependent upon how the exact bound is determined and how this compares with the computational expense required by the Krawczyk's centered form. For example 6.2.1-1: 8 interval additions and/or subtractions and 12 interval multiplications are

required for the unmodified version and 3 interval additions and/or subtractions, 5 interval multiplications, 8 additions and/or subtractions and 6 divisions are required for the modified version. (Note: we have not included the costs for determining F_8 to F_{13} as they are not required.) Each interval addition or subtraction involves two additions or subtractions and each interval multiplication involves 4 multiplications (as well as 6 comparisons).

For example 6.2.1-2: 3 interval vector additions and/or subtractions, 9 interval vector multiplications and 4 interval additions are required for the unmodified version and 3 interval vector additions and/or subtractions, 8 interval vector multiplications, 3 interval additions and/or subtractions and 1 interval multiplication are required for the modified version. Each interval vector addition or subtraction requires 4 additions or subtractions, each interval vector multiplication requires 8 multiplications and 12 comparisons, each interval addition or subtraction requires 2 additions or subtractions and each interval multiplication requires 4 multiplications and 6 comparisons.

Assuming each subtraction is equivalent to an addition and each division is equivalent to a multiplication a rough comparison is made and the results are summarized in Table 6.3. From the results (for both examples) the modified version is computationally cheaper.

The modified version is generally computationally cheaper because the modified terms do not in general use the slope function, G , and thus do not involve interval vector manipulations. Instead they will involve Interval manipulations which are usually cheaper in cost. The actual saving will depend on the means by which the exact bound is determined.

version	additions	multiplications	comparisons
Example 6.2.1-1			
modified	14	26	30
unmodified	16	48	72
Example 6.2.1-2			
modified	18	68	102
unmodified	20	72	108

Table 6.3: Performance test of Modified vs Unmodified Krawczyk Centered Forms

6.2.2 Conclusion

The modification can only improve upon the bound produced and will in general reduce the amount of computation required. Thus, even if no improvement on the bound is made, it will usually be less expensive computationally. In addition, if an improvement is made in the bound, then it occurs for large to medium sized intervals. This is a desirable effect as in general bounding functions become less accurate as the size of the region increases. Thus it will usually be beneficial to use the modified Krawczyk centered form over that of Krawczyk's centered form

Chapter 7

Branch and Bound Based Global Optimization System

In this Chapter we examine how the issues arising from the use of an automaton can be dealt with and we present a global optimization system based on a branch and bound framework (GOS-BB). An initial outline is given followed by an examination of the procedures used to determine feasibility and boundedness, of each of the major steps of the branch and bound framework, and of the algorithms for the detection of minimally inconsistent constraint sets and for the detection of unbounded domain points. We then examine how formulation aids such as flatness of the objective function, sensitivity analysis, periodic examination of the state of the solution process, etc. can be provided. Finally, we examine how INDETERMINATE outcomes can be evaluated.

We recall that for an automaton Θ upon which the GOS-BB is implemented, that lbN_{Θ} and ubN_{Θ} are respectively the smallest and largest numbers representable on the automaton Θ . Let the objective function be denoted by $f : R^n \rightarrow R$, the set of constraints by C , the set of variable bounds (not including the bounds enforced by the automaton) by VB , and the set of user constraints by $FC = C \cup VB$.

7.1 Issues Arising from the Use of an Automaton

In this section we examine how the issues arising from the use of an automaton can be dealt with. Specifically we examine the issues of FINITENESS, DILEMMA OF EQUALITY, FUNCTION

REPRESENTATION, and RESTRICTED FUNCTION DOMAINS.

7.1.1 Finiteness

The finiteness of the number system of the automaton is dealt with by the two-sided rounding of an element of the reals to the smallest interval that contains it. That is, for each element both the next highest and lowest representable number are determined, and these are then used to construct the smallest representable interval that contains it. The element is then represented by this interval. All operators acting on such intervals are defined such that they preserve the correctness of the bounds. In this way, the error in using the finite number system is accounted for because correct bounds for a calculation are provided. One sided roundings (for example, to the nearest number representable) lack this ability to provide correct bounds and thus are not effective. Both the two-sided and one-sided rounding methods will in general fail to provide an exact value for a calculation and are thus equally poor at providing accurate results.

Furthermore, all variables are restricted to values between lbN_Θ and ubN_Θ . The values lbN_Θ and ubN_Θ are used to represent $-\infty$ and $+\infty$ respectively and are an element of the extended interval arithmetic as described in Section 2.3.2. Extended interval arithmetic may be used to allow for the division by intervals which contain a zero as well as at least one non-zero value (division by an point interval equal to zero is still undefined and any attempt to divide by one should be prevented by the inclusion of a check for such an occurrence in the division process).

If for a given global optimization problem the bounds on the variables exceed the range of the set of numbers representable by the automaton, then as it is a compact set, the problem can, by the rescaling of the variables, be reformulated so that the bounds are representable by the automaton.

The finiteness of the number of calls to procedures means that all algorithms must terminate in a finite number of iterations. As such, care must be taken with infinite procedures which are truncated to finite procedures when implemented on an automaton (e.g. truncated Taylor series approximations). Usually, a bound on the error introduced by the truncation is available and can be incorporated into the truncated procedure.

7.1.2 Dilemma of Equality

The issue of equality is usually handled by the introduction of equality error tolerances, in which two numbers are deemed to be equal if the absolute value of their difference is less than or equal to an *a priori* defined equality error tolerance. This approach works if the smallest difference between any two representable numbers is not larger than the specified equality error tolerance. However, if this is not the case then a different approach is required. For example, consider an automaton with 15 digits of accuracy in each computation and two successive representable numbers of 1.0×10^{100} and $9.99999999999999 \times 10^{99}$ (with a difference of 1.0×10^{85}). If the first number was the value obtained for an incumbent solution and the latter was a lower bound on the incumbent solution, then as the two numbers are successive, this is the smallest difference that can be determined on the automaton. Thus, an equality epsilon of at least 1.0×10^{85} would be required. We note that this effect is due to the finiteness of the set of representable numbers and that the effect limits the usefulness of the automaton to solving problems with large domains.

The approach taken is to accept two numbers as being equal if they agree to more than an *a priori* determined number of digits of accuracy. The automaton will place an upper limit on the *a priori* determined number of digits (for the above example, 15 digits of accuracy is the maximum value allowed).

The exception to the this approach will be for the case of determining if a constraint is satisfied, which will be dealt with by the use of an error tolerance. We do so for the following two reasons:

- we wish to use the error tolerance to determine the amount of relaxation of the constraint that is allowed and thus a separate parameter is desirable.
- it is the traditional method by which equality constraints have been handled.

As the equality constraints are dealt with in this manner, then inequality constraints are dealt with in like manner, as any equality constraint can be written as two inequality constraints.

Effective Search Region

As noted above, the use of the automaton limits the accuracy of calculation. This limits the effective size of the region that can be searched. For example, if $lb.N_{\Theta} = -1.0 \times 10^{100}$ and $ub.N_{\Theta} = 1.0 \times 10^{100}$ and only 15 digits of accuracy are available, then the smallest box

that can be examined near the boundary of the region representable by the automaton is of size $1.0 * 10^{85}$. This severely limits the effectiveness of any algorithm in such regions. For example, if an infeasible problem has a small region near this boundary where the problem is close to being feasible, the limits on the accuracy essentially ensure that this region can not be examined closely enough to decide that it is infeasible. Similar problems can occur, if several local optima have similar functional values and some of these are in an area close to the boundary representable by the automaton.

Thus the region that can be effectively searched by an automaton (which we call the *effective search region*) may be smaller than the region that can be represented by the automaton. The effective search region is centered on the origin and is dependent upon the number of digits of accuracy, da , available as well on the width of the smallest subregion that is to be considered, wsr . The effective search region can be thought of as the largest region centered about the origin such that the width of the smallest subregion that can be represented anywhere within the region is smaller than that of wsr .

We propose the following formula to determine the effective search region: add da to the exponent of wsr and use this value as the exponent for the bounds on the variable for the effective search region. This formula is based on the assumption that smallest distance between any two representable points increases as the distance from the two points to the origin increases. For most automaton, this is a valid assumption. Thus, the effective search region is determined by calculating the distance from the origin to the farthest set of two representable points whose separating distance is less than the 'distance' determined by the width of the smallest subregion to be considered and the number of digits of accuracy available.

For example, for 13 digits of accuracy and a wsr of 10^{-8} , we get $13 + -8 = 5 \rightarrow -1.0 * 10^5 \leq x_i \leq 1.0 * 10^5$ as the effective search region. Thus, for region $-1.0 * 10^5 \leq x_i \leq 1.0 * 10^5$, any subregion can be searched to a width of 10^{-8} .

For some problems, a region larger than the effective search region can be examined, but invariably this means that the 'solution' is localized in the effective search region and the problem is 'well behaved' in the remaining regions, allowing for the elimination of these regions from contention. For the general problem this is not necessarily true. The GOS-BB indicates this by the presence of boxes in a set labeled SB_a . Unfortunately, when this occurs, there are usually many such boxes, thus decreasing the efficiency by which the

solution process can both identify and handle such problems.

For example: for problem Unbounded 3 (see Section 5), which has a false domain unbounded point (a point where both the denominator and the numerator are zero) at $x = 2$ and is unbounded at ∞ we get the following behaviour for various values for lbN_{Θ} and ubN_{Θ} :

- With $lbN_{\Theta} = -1.0 * 10^1$ and $ubN_{\Theta} = 1.0 * 10^1$ an outcome of INDETERMINATE with an Incumbent Solution of -14.0 at a point (10) , a Lower Bound of $-\infty$, and a set of 2 boxes, $\{[1.9999999925, 2.0000000019]$, and $[9.9999999906, 1.0000000000 * 10^1]\}$ are obtained.
- With $lbN_{\Theta} = -1.0 * 10^{10}$ and $ubN_{\Theta} = 1.0 * 10^{10}$ an outcome of INDETERMINATE with an Incumbent Solution of $-1.0000000004 * 10^{10}$ at a point $(1.0 * 10^{10})$, a Lower Bound of $-\infty$, and a set of 2 boxes, $\{[1.9999999990, 2.0000000083]$, and $[9.999999999 * 10^9, 1.0000000000 * 10^{10}] \in SB_a\}$ are obtained.
- With $lbN_{\Theta} = -1.0 * 10^{100}$ and $ubN_{\Theta} = 1.0 * 10^{100}$ the solution process terminated when the maximum of 10,000 boxes occurred and gives an outcome of INDETERMINATE with an Incumbent Solution of $-9.999999999 * 10^{49}$ at a point $(1.0 * 10^{50})$, a Lower Bound of $-\infty$, and a set of 10,000 boxes of which 1 is $[1.9999999932, 2.0000000031]$, 46 are contained in the box $[1.0000000058 * 10^{50}, 1.0000000001 * 10^{75}]$ and the remaining 9953 boxes are contained in the box $[9.999999999 * 10^{49}, 1.0000000059 * 10^{50}]$ and are in the set SB_a .

Thus, while GOS-BB correctly handles these types of problems, we see that GOS-BB is ineffective if the effective search region is much smaller than the region that can be represented. For problems which do not require a detailed search in the non-effective search regions GOS-BB performs efficiently. For example, problem Six Hump Back cs11 (see Section 5) found a global minimum (with all boxes smaller than a specified amount) for an initial box, $B_1 = -1.0 * 10^{10} \leq x_i \leq 1.0 * 10^{10}$, in 384 iterations and when an initial box, $B_2 = -1.0 * 10^{100} \leq x_i \leq 1.0 * 10^{100}$, was used, in 381 iterations.

If a problem does require a search in a region other than the effective search region, then by rescaling and offsetting the problem the effective search region can be moved to a different region of R^n . However, this procedure may introduce errors and is unlikely to be effective if the region to be searched is much larger than the size of the effective search region.

7.1.3 Function Representation

The representation used for a function is usually determined by the method used to enter it into the solution process, usually by the means of a parser, etc. If such a parser is used, it can also be used to generate other possible representations and/or evaluate the 'complexity' of the function. Complex systems for the identification of the 'best' function representation based on an analysis of the function are possible, or the solution process may use several simultaneous representations, choosing the 'best' of the values generated, etc.

Another approach is to try and limit the error in a representation due to multiple occurrences of a variable (which are treated independently by interval analysis). For example, see Lodwick [70]. He uses a method which makes use of a procedure called constraint interval arithmetic, which can be used to tighten the inclusion function for functions which have variables that appear more than once. He accomplishes this by retaining the dependence between the multiple occurrence of a variable by adding in a strict equality that can be used to solve one occurrence of the variable in terms of a second occurrence.

7.1.4 Restricted Function Domains

The restricted function domains can be dealt with by examining the set of functions (objective and constraint) and identifying the restricted domains. The restricted domains can then be added to the set of constraints and the procedures to evaluate the functions modified such that an attempt to evaluate a functions restricted domain is not allowed. Of necessity, the procedures which make use of the function evaluation must also be structured so as to allow for the refusal of the evaluation procedure to evaluate the function.

While the above will deal with the evaluation of the function at a feasible point, the evaluation of a bounding function for this type of function requires a different solution if the restricted domain can not be exactly partitioned into a finite set of the sub-region types used by the bounding function. For example, the function $f(x) = \ln(x_1^2 + x_2^2 - 1)$ has a restricted domain of $(x_1^2 + x_2^2 - 1) > 0$. As this is an open set, it cannot be exactly modelled by a finite set of intervals, thus a bounding function based on intervals will run into difficulties for intervals which cross into the restricted domain.

One possible method is to modify the bounding function to allow for intervals which extend into the undefined region. For example, for the above function, consider the interval $A = ([0.8, 1.2], [0.8, 1.2])$. The points of A which lie within $(x_1^2 + x_2^2 - 1) \leq 0$ are undefined.

2.2 · Set the lower bound for the box, $B = lb(BF(B))$.

2.3 · IF $lb(BF(B)) > lb.N_{\Theta}$ THEN

2.3.1 · Set B 's boundedness status to bounded.

ELSE

2.3.2 · Set B 's boundedness status to unknown.

ELSE

2.4 · Set B 's boundedness status to unbounded.

As each bounding function used is guaranteed to include the correct range of the function, the intersection of the bounds generated from the set of bounding functions must also include the correct range of the function.

A box is unbounded only if an unbounded domain point can be found within the box. It is bounded if it has a finite lower bound. Thus if the generated bound has a value $> lb.N_{\Theta}$ then we know a finite lower bound exists. If the generated bound is $\leq lb.N_{\Theta}$ and an unbounded domain point has not been found then the existence of a finite lower bound is unknown. If a set of unbounded domain points is identified then the program's boundedness status is changed to unbounded.

The process of determining a bound by the application of a bounding function is examined in more detail in Section 7.8. Briefly it the existing bounding function is modified so as to detect candidate unbounded domain points which are subsequently analyzed by the Unbounded Point Detection Algorithm (see Section 7.8.3). If any unbounded domain points are found they are subsequently output.

7.5 Components of the Branch and Bound Framework

We now examine the components of the algorithm of Section 7.2 in more detail (A complete listing of the detailed algorithm is available in Appendix C):

However, we can determine a bound on $f(x)$ over A as $F(A) = [-\infty, \ln(1.2^2 + 1.2^2 - 1)] = [-\infty, .63128]$ as we know that $\ln(x)$ is monotonic and that $x_1^2 + x_2^2$ attains its maximum value over A at the point $[1.2, 1.2]$.

This method is to be used with caution, but can be used to determine a bound on the function if the intention is to bound the values of the function within the restricted domain. In this case, the bounding function is inadequate (as it cannot produce an exact finite model of the restricted domain) and thus this method can be used to counter the inadequacy. The set of unary functions is responsible for restricted function domains, for which a procedure must be created for both point and interval evaluation. Thus the information required for this method can likely be embedded into the procedure used to calculate the function and its bounding functions (see Section 10.3).

7.2 An Outline of GOS-BB

In Section 4.2.5 we examined the state of the art in global optimization, in terms of the types of restrictions required and the information provided by the methods. From this analysis we developed criteria for a GOS that would be more ‘useful’ from a modellers perspective. The only class of methods which can be easily modified to meet these criteria and which is applicable to the set of global optimization problems representable on an automaton was found to be the covering methods. The basic branch and bound framework of the covering methods was chosen for GOS-BB.

A branch and bound algorithm which serves as the framework for GOS-BB (see Section 7) is as follows:

- 0 · Determine representations for f and all C_i for the automaton.
- 1 · Initialize Parameters, create the Initial Box and initialize the Set of Boxes to the set {Initial Box}.
- 2 · Test Initial Box:
 - 2.1 · Test Initial Box for feasibility.
 - 2.2 · Test Initial Box for boundedness.
 - 2.3 · Check Stopping Condition.

3 · Branch:

3.1 · Create New Boxes from a subset of the Set of Boxes.

3.2 · Test New Boxes for entry into the Set of Boxes.

4 · Fathom:

4.1 · Seek an improved incumbent solution.

4.2 · Remove all elements of the Set of Boxes whose lower bounds are strictly greater than the functional value of the Incumbent Solution..

5 · Update:

5.1 · Check if Set of Boxes is empty.

5.2 · Update Lower Bound for the global minimum.

6 · Check Stopping Condition:

6.1 · IF stopping condition is false THEN

6.1.1 · goto 3.

ELSE

6.1.2 · Output the following:

* Incumbent Solution.

* Lower Bound.

* Set of Boxes.

6.1.3 - OPTIONAL · Determine (and output) flatness of the objective function, and/or other formulation aids.

6.1.4 · STOP

The algorithm's goal is to determine a global minimum within some *a priori* determined degree of accuracy for the given objective function, f , subject to a set of constraints, C . It does so by maintaining a set of sub-regions of the domain which is guaranteed to contain a global minimum (if one exists). The type of sub-region chosen is that of an N-dimensional

interval or box. The Initial Box is assumed to contain the domain of the problem to be solved and is assumed to have finite bounds such that the numerical values of the bounds can be represented on the automaton being used, i.e., lie between lbN_{Θ} and ubN_{Θ} .

A box is *feasible* if it contains only feasible points and it is *infeasible* if it does not contain any feasible points. A box which cannot be shown to be feasible or infeasible is of *unknown* feasibility. A box is *bounded* from below if a lower bound for the functional values of the objective function is greater than lbN_{Θ} . It is *unbounded* if it can be shown that an unbounded domain point (see Section 7.8) exists within the box. Otherwise the box is of *unknown* boundedness.

The Initial Box is tested for its feasibility and boundedness. The incumbent solution is then set to the best feasible point known and the Lower Bound (for the global minimum) is set to the lower bound of f over the Initial Box.

At each iteration of the algorithm (Steps 3 to 6): a subset of boxes is chosen for subdivision into a set of smaller boxes. The chosen boxes are then removed from the Set of Boxes and each is subdivided into a set of smaller boxes to generate a set of New Boxes. Each New Box is then tested for its feasibility and boundedness. Each non-infeasible New Box whose lower bound is less than or equal to the incumbent solution's functional value is entered into the Set of Boxes. A search for an improved incumbent solution may then be undertaken, followed by a fathoming, or pruning, of the Set of Boxes. The Set of Boxes is fathomed to eliminate those boxes which are known not to contain a global minimum, thereby reducing the size of the domain to be searched. A box is fathomed if it is infeasible or if its lower bound is greater than the functional value of the incumbent solution. If the Set of Boxes does not satisfy the *a priori* determined stopping conditions, then another iteration of the algorithm is carried out.

The algorithm will always stop if an INFEASIBLE or DOMAIN UNBOUNDED outcome has been detected or if the cardinality of the Set of Boxes exceeds that of an *a priori* determined maximum number of boxes. The value of the maximum number of boxes is determined by the amount of storage space available on the automaton. These conditions make up the *basic stopping conditions*.

Each box in the Set of Boxes has associated to it a feasibility status as well as a boundedness status (in the form of a lower bound on the values of the objective function over the box). The Set of Boxes is an ordered set, usually by the lower bound, but other possible

orderings are by the width of the box, by the ‘volume’ of the box, etc.

When implemented on an automaton, a limit exists on the smallest size a box can attain, due to the finiteness of the set of numbers that the automaton can represent. Thus, when a box reaches this size, or some other *a priori* determined *minimum box size*, the box will no longer be subdivided. As the smallest size a box can obtain varies over the domain (see Section 7.1.2), each box is checked to see if it can be subdivided. If it can no longer be subdivided, it is placed in a subset of the Set of Boxes where it can be fathomed, but not further subdivided. If the Set of Boxes does not contain any subdividable boxes, the GOS-BB terminates. This implies that the subdivision process can only occur a finite number of times and the algorithm will terminate after a finite number of iterations.

When the algorithm terminates, the values of the current Incumbent Solution, the Lower Bound and the Set of Boxes are output (from which the outcome can be determined). If the outcome is not INFEASIBLE or DOMAIN UNBOUNDED, the Set of Boxes can then be analyzed to determine the flatness of the objective function about the Incumbent Solution as well as any other desired formulation aids.

7.3 Feasibility Test for a Box

In this section we examine the algorithm used to test a box, B , for its feasibility relative to the set of constraints, C .

Let the constraint function for constraint C_i be denoted by $C_i(x)$, a bound on the range of the values of $C_i(x)$ over B be denoted by $C_i(B)$ and $\delta_{equality} > 0$ be an equality error tolerance used to account for the DILEMMA OF EQUALITY. Let P_{inf} be the set of constraints which have been used to determine previous Boxes as infeasible. A box’s feasibility relative to a set of constraints, C is determined as follows:

- 0 · Initialize B ’s feasibility status as feasible.
- 1 · For each constraint, $C_i \in P_{inf}$:
 - 1.1 · Determine the feasibility of B relative to C_i .
 - 1.2 · IF B is infeasible THEN
 - 1.2.1 · Set B ’s feasibility status to infeasible.

1.2.2 · STOP.

ELSE IF B is of unknown feasibility THEN

1.2.3 · Set B 's feasibility status to unknown.

2 · For each constraint, $C_i \notin P_{inf}$:

2.1 · Determine the feasibility of B relative to C_i .

2.2 · IF B is infeasible THEN

2.2.1 · Set B 's feasibility status to infeasible.

2.2.2 · Add C_i to P_{inf} .

2.2.3 · STOP.

ELSE IF B is of unknown feasibility THEN

2.2.4 · Set B 's feasibility status to unknown.

A box B 's feasibility relative to a single constraint, C_i , is determined as follows:

- For a constraint of the form $C_i(x) \leq 0$:
 - The box is feasible if $C_i(B) \leq \delta_{equality}$.
 - The box is infeasible if $C_i(B) > \delta_{equality}$.
 - Otherwise the box is of unknown feasibility.
- For a constraint of the form $C_i(x) = 0$:
 - The box is feasible if $-\delta_{equality} \leq C_i(B) \leq \delta_{equality}$.
 - The box is infeasible if $C_i(B) < -\delta_{equality}$ or $C_i(B) > \delta_{equality}$.
 - Otherwise the box is of unknown feasibility.

A box can have three possible feasibility states: feasible, infeasible and unknown. In order for a box to be feasible, it must be feasible relative to each constraint. In order for a box to be infeasible, it need only be infeasible relative to one constraint. If a box cannot be shown to be infeasible or feasible relative to a constraint it is of unknown feasibility.

Thus each box is tested against each constraint. If it is infeasible against one constraint the algorithm stops. If it is unknown against one constraint, then it can no longer be feasible, but it may still be infeasible; thus we continue to test the box against the remaining constraints in the hope of determining an infeasible state.

We partition the set of constraints into two disjoint sets. The first set, P_{inf} , contains the constraints which have been used at least once to determine that a box (previously tested) is infeasible. The other set is the complement of P_{inf} . The algorithm checks the constraints in P_{inf} before that of its complement in the hopes that this will result in a smaller P_{inf} in the case of an infeasible outcome.

THEOREM 7.3

If the GOS-BB algorithm terminates with an INFEASIBLE outcome, then the set of constraints P_{inf} is an $HISU_{Initial\ Box}$, an inconsistent set of constraints relative to the Initial Box.

PROOF:

The GOS-BB algorithm returns an INFEASIBLE outcome if no boxes are left in the Set of Boxes and no feasible point has been found. In such a case, a box is only removed if it is infeasible. Therefore an INFEASIBLE outcome means that the region defined by the Initial Box is an infeasible region. Therefore, it follows that the set of constraints that were used to eliminate the boxes is inconsistent relative to the Initial Box. Q.E.D.

7.4 Application of a Bounding Function Strategy

In this section we examine the algorithm used to test a box, B , for its boundedness. The Bounding Function Strategy will use a set of bounding functions, BF . We denote a bound on the range of the values of f over B determined by the i^{th} Bounding Function, BF_i , by $BF_i(B)$.

1 · FOR each $BF_i \in BFS$:

1.1 · Determine the bound, $BF_i(B)$.

2 · IF an unbounded domain point is not found in 1.1 THEN

2.1 · Determine the final bound, $BF(B) = \cap_i BF_i(B)$.

2.2 · Set the lower bound for the box, $B = lb(BF(B))$.

2.3 · IF $lb(BF(B)) > lbN_{\Theta}$ THEN

2.3.1 · Set B 's boundedness status to bounded.

ELSE

2.3.2 · Set B 's boundedness status to unknown.

ELSE

2.4 · Set B 's boundedness status to unbounded.

As each bounding function used is guaranteed to include the correct range of the function, the intersection of the bounds generated from the set of bounding functions must also include the correct range of the function.

A box is unbounded only if an unbounded domain point can be found within the box. It is bounded if it has a finite lower bound. Thus if the generated bound has a value $> lbN_{\Theta}$ then we know a finite lower bound exists. If the generated bound is $\leq lbN_{\Theta}$ and an unbounded domain point has not been found then the existence of a finite lower bound is unknown. If a set of unbounded domain points is identified then the program's boundedness status is changed to unbounded.

The process of determining a bound by the application of a bounding function is examined in more detail in Section 7.8. Briefly it the existing bounding function is modified so as to detect candidate unbounded domain points which are subsequently analyzed by the Unbounded Point Detection Algorithm (see Section 7.8.3). If any unbounded domain points are found they are subsequently output.

7.5 Components of the Branch and Bound Framework

We now examine the components of the algorithm of Section 7.2 in more detail (A complete listing of the detailed algorithm is available in Appendix C):

7.5.1 STEP 1: Initialize

In Step 1, the parameters that control the solution process are initialized.

- 1.1 · Choose a stopping condition.
- 1.2 · Determine a bounding function strategy, *BFS*.
- 1.3 · Determine a branching (subdivision) strategy, *BSS*.
- 1.4 · Determine an initial inclusion, the Initial Box B , of the feasible region and initialize the Set of Boxes, $SB = \{B\}$.
- 1.5 · Initialize the Incumbent Solution (default: $+\infty$).
- 1.6 · Initialize the Lower Bound to $-\infty$.
- 1.7 · Initialize program status:
 - 1.7.1 · Set problem feasibility status (default: unknown).
 - 1.7.2 · Set problem boundedness status (default: unknown).
- 1.8 · Initialize the set P_{inf} (default: $\emptyset$).

The initial inclusion is either determined from the variable bounds of the problem or by the default values equal to the largest positive and negative numbers representable on the automaton.

The bounding and branching strategies determine what method(s) are to be used to generate the bounds on the objective function and/or constraint functions as well as the type(s) of subdivision to be used. See Section 8.2 for details on subdivision strategies and Sections 8.1.1 and 8.1.3 for details on bounding function strategies.

The stopping condition determines how the algorithm will terminate, in addition to the basic stopping conditions (see Section 7.2). Some possible stopping conditions are as follows:

1. Stop if the difference between the Lower Bound and the incumbent solution is less than an *a priori* determined value.
2. Stop if all boxes are less than an *a priori* determined size or if the difference between the Lower Bound and the incumbent solution is less than an *a priori* determined value.

3. Stop if all boxes are less than an *a priori* determined size and if the difference between the Lower Bound and the incumbent solution is less than an *a priori* determined value.
4. Stop if a feasible point is found.

Stopping conditions 1 to 3 are used if the goal is to find a global minimum. Condition 1 centers on the accuracy of the value of the incumbent solution, while conditions 2 and 3 are also concerned with the minimum size of the box to be used and thus are used to limit the depth of the search. Stopping Condition 4 is used if only the feasibility of the problem is to be determined.

7.5.2 STEP 2: Test Initial Box

In Step 2, the Initial Box is tested for feasibility and boundedness.

2.1 · Test B for feasibility.

2.2 · IF B is feasible THEN

2.2.1 · Check the lower point, the upper point and the mid-point of B as candidate incumbent solutions.

ELSE IF B is infeasible THEN

2.2.2 · Change problem feasibility status to infeasible.

2.2.3 · Apply the deletion filter $HISU_{\Theta}$ detection algorithm to the set
 $FC_{inf} = P_{inf} \cup VB$.

2.2.4 · STOP.

2.3 · Apply BFS over B .

2.4 · IF B is bounded THEN

2.4.1 · Change problem boundedness status to bounded.

2.4.2 · Update Lower Bound = lower bound for B .

ELSE IF B is unbounded THEN

2.4.3 · Change problem boundedness status to unbounded.

2.4.4 · STOP.

2.5 · IF stopping condition is true THEN

2.5.1 · goto 6.1.2.

2.6 · OPTIONAL · Seek an improved incumbent solution.

2.7 · IF an improved incumbent solution has been found in 2.2.1 or 2.6 THEN

2.7.1 · Change problem feasibility status to feasible.

2.7.2 · Update Incumbent Solution.

The Initial Box is analyzed separately for its feasibility and boundedness as the results of these tests affect the subsequent set of boxes in a global manner. For example, if the Initial Box is bounded, then all of the subsequent boxes are also bounded (albeit with different lower bound values). Finally, a check to see if the stopping conditions are true is made before the algorithm continues.

When to search for an incumbent solution and possible means by which such as search could be carried out are discussed in Section 7.6.

7.5.3 STEP 3: Branch

In Step 3, the creation and subsequent testing of the set of New Boxes is carried out and 'valid' New Boxes are added to the Set of Boxes.

3.0 · Initialize Set of New Boxes, SNB , to $\emptyset$.

3.1 · Create New Boxes:

3.1.1 · From BSS , determine the set of boxes to Subdivide, SSB .

3.1.2 · For each box, $B \in SSB$:

3.1.2.1 · Subdivide B and enter each New Box into SNB .

3.1.2.2 · Delete B from SSB and SB .

3.2 · Evaluation of New Boxes: For each $NB \in S.NB$:

3.2.1 · Apply *BFS* over NB .

3.2.2 · IF NB is unbounded THEN

3.2.2.1 · Change problem boundedness status to unbounded.

3.2.2.3 · STOP

ELSE IF the lower bound for $NB >$ Incumbent Solution THEN

3.2.2.3 · Goto 3.2

3.2.3 · IF the parent of NB is of unknown feasibility THEN

3.2.3.1 · Test NB for feasibility.

3.2.3.2 · IF NB is infeasible THEN

3.2.3.2.1 · Goto 3.2

ELSE IF NB is feasible AND the problem is of unknown feasibility THEN

3.2.3.2.2 · Change problem feasibility status to feasible.

3.2.4 · Enter NB into SB .

3.2.5 · If NB is feasible THEN

3.2.5.1 · Check the lower point, the upper point and the mid-point of NB as candidate incumbent solutions.

3.2.6 - OPTIONAL · Seek an improved incumbent solution

3.2.7 · IF an improved incumbent solution is found in 3.2.5.1 OR 3.2.6 THEN

3.2.7.1 · Update Incumbent Solution

3.2.7.2 · IF problem feasibility status is unknown THEN

3.2.6.2.1 · Change problem feasibility status to feasible

Each New Box has its boundedness and feasibility determined. If an unbounded domain point is identified by the Bounding Function Strategy then the algorithm terminates with a DOMAIN UNBOUNDED outcome and a set of domain unbounded points are output. If the parent box (the box whose subdivision created the New Box) was feasible then the box is

known to be feasible. New Boxes which are found to be infeasible or which have a lower bound greater than the incumbent solution are not entered into the Set of Boxes.

When to search for an incumbent solution and possible means by which such as search could be carried out are discussed in Section 7.6.

7.5.4 STEP 4: Fathom

In Step 4, the Set of Boxes are pruned of boxes which cannot contain a global minimum.

4.1 - OPTIONAL - Seek an improved incumbent solution

4.2 - IF an improved incumbent solution has been found in 3.2.5.1 OR 3.2.6 OR 4.1 THEN

4.2.1 - Remove all boxes in SB whose lower bound $>$ Incumbent Solution.

When to search for an incumbent solution and possible means by which such as search could be carried out are discussed in Section 7.6.

If an improved incumbent solution has been identified during the current iteration, then the Set of Boxes is fathomed, or pruned of all boxes whose lower bound is strictly greater than that of the improved incumbent solution. Infeasible boxes are fathomed implicitly when we only allow non-infeasible boxes to enter the Set of Boxes in Step 3.2.4.

7.5.5 STEP 5: Update

In Step 5, the problem feasibility and boundedness statuses are updated.

5.1 - IF $SB = \emptyset$ THEN

5.1.1 - Change problem feasibility status to infeasible.

5.1.2 - Apply the deletion filter $HISU_{\emptyset}$ detection algorithm to the set

$$FC_{inf} = P_{mf} \cup VB.$$

5.1.3 - STOP.

ELSE

5.1.4 - Lower Bound = least lower bound of the boxes in SB .

5.1.5 · If Lower Bound is $> lb_{N_0}$ and problem boundedness status is unknown
THEN

5.1.5.1 · Change problem boundedness status to bounded

If the Set of Boxes is empty, then the problem is infeasible. The set of constraints in P_{inf} is inconsistent and is then further pruned by the deletion filter HST_0 detection algorithm (see Section 7.7) to identify a minimally inconsistent set of constraints (relative to the Initial Box). The minimally inconsistent set of constraints is then output as well as an INFEASIBLE outcome.

Otherwise, the Lower Bound is update to the least lower bound of the boxes in the Set of Boxes.

7.6 Searching for an Incumbent Solution

A good incumbent solution is desirable as it increases the probability that a box which does not contain a global minimum will be fathomed in Step 4. In this section we examine conditions for initiating a search for an improved incumbent solution as well as methods to be used in the search.

Three Steps (2.6, 3.2.6 and 4.1), have optional searches for an incumbent solution. In Step 2.6, the Initial Box may be searched, in step 3.2.6 a non-infeasible New Box may be searched and in step 4.1, the complete Set of Boxes (or a subset such as the ‘best’ m boxes may be searched.

In general, if no finite incumbent solution has been determined by the algorithm then it is usually advisable to try to locate one as soon as possible. The added computational expense of the search may reduce computation later as the earlier (and thus generally larger) boxes may be fathomed, thus reducing the size of the region that needs to be considered.

If the Set of Boxes is growing faster than a specified rate then a search for an improved incumbent solution may be advisable. We note that even with a good incumbent solution, the Set of Boxes can grow quickly, perhaps due to inaccurate bounding functions or to a flat non-linear objective function (which will limit the accuracy of the bounding function), etc. However, as the Set of Boxes is continually subdivided, a search for an incumbent solution for each New Box (step 3.2.6) may result in repeated attempts for the same region, so a

search at step 3.2.6 should be limited, perhaps by keeping track of the regions for which a search has already been conducted.

Another factor is the size of the box to be searched. As the box size decreases, the likelihood that a search will greatly improve the incumbent solution decreases. Thus a general rule of thumb would be to search the larger boxes more intensely than the smaller boxes, as a larger gain is more likely for a large box than for a small one. Therefore, if the cost of the search is 'acceptable' then one should always apply a search at step 2.6, but for step 3.2.6 only if the New Box is of a 'large' enough size and at step 4.1 only for 'large best' boxes.

Several methods are possible for searching for an improved incumbent solution. The simplest is to generate a set of seed points (from a grid or randomly) and then evaluate each of these points. More complicated methods involve using local optimization methods in conjunction with the generation of a set of seed points. A cheap and effective method is desired. In general, if the box is feasible, then a stochastic global optimization method is applicable and likely to be the best option as it is usually quick and reasonably effective (see Section 4.2.2), whereas if the box is of unknown feasibility, then a constrained local optimization method is likely to be a good choice if the functions (objective and constraint) are differentiable, otherwise a method which uses the interval branch and bound line search method (see Appendix D) may be a good choice.

7.7 Deletion Filter $IISU_{\Theta}$ Detection Algorithm

This method is a modified version of the deletion filtering algorithm developed by Chinneck and Dravnieks [23, 13] to identify a minimally inconsistent set for infeasible linear programming problems. Given an input set of inconsistent functional constraints, FC_{mf} , it will determine an $IISU_{\Theta}$, that is a minimal inconsistent set of functional constraints relative to the automaton Θ (which enforces the non-functional constraints of the form $lbN_{\Theta} \leq x_i$ and $x_i \leq ubN_{\Theta}$) (see Sections 3.3 and 2.2.1). The box, H_{Θ} , is the box defined by the variable bounds $lbN_{\Theta} \leq x_i$ and $x_i \leq ubN_{\Theta}$.

- 0 · Replace f with the following 'Phase I'-like objective function (i.e. the sum of the constraint violations):

$$f_{Phase I} = \sum_i \max(g_i(x), 0) + \sum_j |h_j(x)|$$

where $g_i(x)$ and $h_j(x)$ are the constraint functions of the constraints (of the appropriate type) of FC_{inf} .

1 · For each constraint $C_i \in FC_{inf}$ DO:

1.1 · Remove the constraint from FC_{inf} . Apply a GOS-BB to the new problem, stopping when a feasible point has been detected or the set of constraints has been shown to be infeasible. If the result is an infeasible set, label this set FC_{inf_i} .

1.2 · IF a feasible point is found THEN

1.2.1 · Return the constraint C_i to FC_{inf} .

ELSE IF FC_{inf_i} is a proper sub-set of FC_{inf} THEN

1.2.3 · Replace FC_{inf} by FC_{inf_i} .

When the algorithm terminates, the set FC_{inf} is an $IISU_{\Theta}$ (see Section 3.3), a minimal inconsistent set of constraints relative to the initial inclusion of the problem.

The problem is initially modified so as to remove any constraints which are not in the set FC_{inf} . Each constraint, C_i , of FC_{inf} is removed in turn from the modified problem. If the removal results in an infeasible problem, then the constraint is removed from FC_{inf} as the set $FC_{inf} \setminus C_i$ is inconsistent. If the removal results in a feasible problem, then the constraint remains in FC_{inf} as the set $FC_{inf} \setminus C_i$ is not inconsistent. If at any time an inconsistent set FC_{inf_i} which is a proper subset of the current FC_{inf} is found, replace FC_{inf} by FC_{inf_i} as it is of smaller cardinality.

A 'Phase I'-like objective function is one of many possible options for the objective function. Since it is a measure of the feasibility of a point, it may possibly guide the GOS-BB towards a feasible point. An analysis of the effectiveness of the 'Phase I' objective function is carried out in Section 8.2.5.

THEOREM 7.7

The Deletion Filter $IISU_{\Theta}$ Detection Algorithm identifies an $IISU_{\Theta}$ if the initial input set, FC_{inf} , is inconsistent relative to the box, H_{Θ} .

PROOF:

As a constraint, C_i , is removed from FC_{inf} only if the set $FC_{inf} \setminus C_i$ is inconsistent relative to H_Θ , the set FC_{inf} at the end of the algorithm is inconsistent. Each constraint, C_i remains in FC_{inf} only if $FC_{inf} \setminus C_i$ is consistent. If $FC_{inf} \setminus C_i$ is not consistent, then the removal of any other constraint from $FC_{inf} \setminus C_i$ will also result in a non-inconsistent set. Thus, for the final FC_{inf} , the removal of any constraint results in a non-inconsistent set. Thus the final FC_{inf} is a minimally inconsistent set of functional constraints relative to H_Θ and is thus an $IISU_\Theta$. Q.E.D

7.7.1 Evaluation

Consider the following Constraint Set (see Figure 7.1):

$$\begin{aligned}
 -2x_1^4 + 8x_1^3 - 8x_1^2 + x_2 - 2 &\leq 0 \equiv a \\
 4x_1^4 - 32x_1^3 + 88x_1^2 - 96x_1 - x_2 + 36 &\leq 0 \equiv b \\
 5x_1^2 + x_2 - 10 &\leq 0 \equiv c \\
 4x_1^3 + 6x_1^2 - 3x_1 - x_2 + 2.5 &\leq 0 \equiv d \\
 0 \leq x_1 &\equiv l_1 \\
 x_1 \leq 3 &\equiv u_1 \\
 -10 \leq x_2 &\equiv l_2 \\
 x_2 \leq 20 &\equiv u_2
 \end{aligned}$$

with an objective function of $-x_1 - x_2$. The constraint set $\{a, b, d\}$ form an $IISU_{VB}$. We examine the case if the GOS-BB determines an inconsistent set P_{inf} of $\{a, b, c, d\}$. The set, $FC_{inf} = P_{inf} \cup \{l_1, u_1, l_2, u_2\}$ would be input into the Deletion Filter $IISU_\Theta$ Algorithm.

At each iteration the following would occur:

Figure 7.1: Constraint Set 6

FC_{inf}	feasibility
$b, c, d, l_1, u_1, l_2, u_2$	feasible
$a, c, d, l_1, u_1, l_2, u_2$	feasible
$a, b, d, l_1, u_1, l_2, u_2$	infeasible
a, b, l_1, u_1, l_2, u_2	feasible
a, b, d, u_1, l_2, u_2	infeasible
a, b, d, l_2, u_2	infeasible
a, b, d, u_2	infeasible
a, b, d	feasible

and the set $\{a, b, d, u_2\}$ would be identified as an $HISU_{\emptyset}$.

7.8 Unbounded Point Detection

For a real valued function, $f : D \subseteq \mathbb{R}^n \rightarrow \mathbb{R}$ which is continuous at all points except where it is undefined. If the set of points at which it is undefined is 'isolated', that is, for each undefined point, there exists a sequence of points (for which f is defined) that converges to the undefined point. If such a sequence, x_k , exists and if $\lim_{k \rightarrow \infty} f(x_k) = -\infty$ then the limit

point of the sequence is an unbounded from below point. If such a point is a feasible point relative to a set of constraints, then we label it as an unbounded domain point.

This suggests the following approach for detecting unbounded domain points within a box: identify the feasible points at which the function is undefined and check these points to see if they are unbounded domain points by applying a set of verification tests. If the tests fail then return to the 'normal' operation of branching and bounding, in the hopes that further subdivision will provide either a domain unbounded point or a finite lower bound greater than lbN_{Θ} . This follows as smaller boxes will result in more accurate bounds from the bounding function strategy and/or improve the localization of the unbounded domain point and thus improve the identification of unbounded domain points as well as the possibility of proving the existence of the required subsequence.

This approach is not robust and can fail if the global minimum value is less than lbN_{Θ} or if the verification tests fail to verify the domain unboundedness. In the former case, the problem needs to be rescaled, a procedure which will, in general, benefit from any localization that has occurred. In the latter case, the set of verification tests must be either expanded or revised to improve their accuracy. We examine first how a set of candidate unbounded points can be identified and then examine some verification tests which can be used to verify the existence of domain unboundedness.

7.8.1 Identification of Candidate Unbounded Domain Points

As noted above, we can identify candidate unbounded domain points by identifying the feasible points at which a function is undefined. We begin by examining the set of representable functions. A representable function is a finite sequence of operations (arithmetic and/or logical) upon a set of variables and/or unary functions. Thus a representable function will contain sub-functions which are rational functions, connected by the use of logical connectives and/or unary functions.

We examine the case for rational functions (we need only consider finite orders due to the finiteness enforced by the use of the automaton). A rational function can be unbounded from below either at a domain point or at infinity. We need only consider domain points due to the finiteness of the automaton. Except for the points at which they are undefined, rational functions are continuous.

The representations for a rational function are of the form

$$f(x) = \sum_i \frac{p_i(x)}{q_i(x)}$$

where $p_i(x)$ and $q_i(x)$ are either rational or polynomial functions. The use of the automaton enforces this sum to be finite. Thus the function will be undefined wherever a division by zero can occur (where the denominator of a sub-rational function of $f(x)$ is zero). For example, the function representation

$$f(x) = \frac{\frac{x_2-2}{x_1-3} + x_1^2 + \frac{x_2+x_1-3}{x_2-3}}{x_2 + x_1}$$

is undefined when $x_1 = -x_2$, $x_1 = 3$ and $x_2 = 3$.

Thus, candidate unbounded domain points can be identified whenever a function evaluation fails due to a division by zero, or when a bounding function evaluation fails due to a division by a zero. Due to the DILEMMA OF EQUALITY, there are two cases which cause a division by a zero. The first occurs if a box, B , is found such that $-\delta_{\text{unbounded}} \leq Q(B) \leq \delta_{\text{unbounded}}$, and the second occurs if a point, b , is found such that $-\delta_{\text{unbounded}} \leq q(b) \leq \delta_{\text{unbounded}}$ (where $\delta_{\text{unbounded}}$ is an equality error tolerance and q is a bounding function for q). For any other point or box, division is defined (as we use extended interval arithmetic to handle division by a box which contains 0 as well as at least one non-zero point). Thus, for a rational function, each division is monitored for a division by a zero, and any such occurrences are sent for verification of the existence of an unbounded domain point. We note that a division by zero is usually handled by using an exception handler to catch such occurrences and signal the appropriate action to be taken. Thus the implementation on an automaton requires only minor modification to the existing exception handler.

The effect of the presence of logical operators is that the domain of the function is separated into a set of domain components (see Section 6.1.1). Logical operators can be eliminated from consideration by determining the domain components and handling each domain component as a separate case. Each domain component will in turn consist of a set of connected open and/or closed sets of domain points. If these can be identified and the closure of each of these sets determined then the above method is applicable to each connected region.

Unary functions are in effect, special collections of finite sequences which have been labelled for ease of use and due to the familiarity of the function they represent. For example,

$\cos(x)$ is usually represented as a truncated Taylor expansion of rational functions. We consider only finite sequences due to the restrictions for finiteness imposed by the automaton. In a theoretical argument we can identify any logical operators hidden in the unary functions and using the propagation method of Section 6.1.1 construct a set of domain components with sub-functions consisting of rational functions. Practically, we can make use of the 'information' that is associated with the unary function in order to aid in the search for candidate unbounded domain points. For example, $\cos(x) = 0$ at $x = \pi/2$ and $\tan(x)$ is known to be unbounded for $x = \pi/2$, etc.

A point to note is that a Unary function may have an implied restricted function domain. For example, $\sqrt{x}$, carries with it the implied constraint of $x \geq 0$. See Section 7.1.4 for a more detailed examination of how to deal with restricted function domains.

The presence of unary functions requires that code exists by which the unary function and associated bounding function can be evaluated. Thus, a prerequisite for the use of a unary function would be that this code has been modified to allow for the flagging of those points or boxes for which the unary function or associated bounding function is undefined.

Thus the task of identifying candidate domain unbounded points is replaced by the task of identifying zeros of the functions (polynomials or unary) located in the denominator of rational functions or sub-functions and/or by using *a priori* acquired knowledge of unbounded points of unary functions.

7.8.2 Verification of Unbounded Domain Points

Once a candidate unbounded domain point has been identified, the existence of the required sequence needs to be verified. We limit our discussion to rational functions of the form $f(x) = \frac{p(x)}{q(x)}$ where the $p(x)$ and $q(x)$ are polynomial functions. The more general case of rational functions as well as for unary functions has been left for future research (see Section 10.3).

Let $\delta_{unbounded} > 0$ be an equality error tolerance used to account for the DILEMMA OF EQUALITY and $M \in R$. A rational function, $f(x) = p(x)/q(x)$, $x \in R^n$, with bounding functions $Q(X)$ for $q(x)$ and $P(X)$ for $p(x)$, over a box, A , is an unbounded domain point (within $\delta_{unbounded}$) at a point, $x_{ubp} \in A$, if x_{ubp} is a feasible point and one of the following is true:

$$\forall:1 \quad i) \quad 0 \leq Q(A) \leq \delta_{unbounded} \Rightarrow q(x_{ubp}) = 0 \text{ and } q(x) \geq 0 \quad \forall x \in A.$$

$$\text{ii) } M < P(A) < -\delta_{\text{unbounded}} \Rightarrow p(x) < 0 \forall x \in A.$$

$$V:2 \quad \text{i) } -\delta_{\text{unbounded}} \leq Q(A) \leq 0 \Rightarrow q(x_{ubp}) = 0 \text{ and } q(x) \leq 0 \forall x \in A.$$

$$\text{ii) } M > P(A) > \delta_{\text{unbounded}} \Rightarrow p(x) > 0 \forall x \in A.$$

Since p and q are polynomials they are continuous at all points. Since f is a ratio of two polynomials, it is continuous at all but a set of points where it is not defined (where $q(x)$ is zero). Equating each factor of the polynomial $q(x)$ to 0 will determine the set of points at which $q(x)$ can be zero. Each factor will define a set of points that form a surface in R^n of dimensionality at most $n - 1$. Since we only consider polynomials of finite order, only a finite set of these surfaces will exist and since each surface is a polynomial it is not space filling (that is, the surface does not fill a region of R^n). Thus the set of points for which $q(x)$ is zero is a set of finite non-space filling surfaces of R^n of dimension at most $n - 1$ and this implies that the set of points for which $q(x)$ is zero is also not space filling in R^n . Thus if $q(x_{ubp} = 0)$ then there exists a neighbourhood, $N(x_{ubp}, r > 0)$, such that there exists a sequence of points $x_k \rightarrow x_{ubp}$ for which $f(x_k)$ is defined for all $k \in R$ and $q(x_k)$ is non-zero for all $k \in R$.

We examine the case if $V : 1$ holds. In this case $q(x) \geq 0$ and $p(x) < 0 \forall x \in A$. Furthermore, as $0 < P(X) < M \in R$, $p(x_k)$ is a strictly positive finite number for all $k \in R$. Therefore as $f(x) = p(x)/q(x)$, $f(x_k) < 0$ for all $k \in R$ and as $\lim_{k \rightarrow \infty} f(x_k) = -\infty$. Therefore f is unbounded from below at x_{ubp} and as x_{ubp} is feasible, x_{ubp} is an unbounded domain point. The case for $V : 2$ is similar.

If a point, a , is a candidate unbounded domain point, then a small box can be placed about a and tested as above. The easiest fashion by which such a box can be found is to find the next highest and lowest representable numbers about a and use these to generate a set of boxes.

We note that if the function f is not in reduced form, that is, p and q have common roots, then if f is domain unbounded at one of these common roots the above method will fail as p will also be zero. Thus, it is desirable for any functions to be in reduced form. One possible means by which a rational function can be reduced is by the use of symbolic manipulation (see Section 10.3).

Thus we can verify that a feasible candidate unbounded domain point is a domain unbounded point by finding an interval which satisfies the above verification conditions for a

class of rational functions. The given method is not robust, as it depends on the representation used for the given rational function. Further research is required to extend this to the general class of codeable functions.

7.8.3 Unbounded Point Detection Algorithm

We present a methodology for an unbounded domain point detection algorithm for rational functions. Apply the bounding function, BF_i to the box B with the following modifications:

1 · IF a division by a box A such that $-\delta_{unbounded} \leq Q(A) \leq \delta_{unbounded}$ is to occur THEN

1.1 · Determine A 's feasibility or the existence of a feasible point, $a \in A$.

1.2 · IF A is feasible THEN

1.2.1 · Apply the unbounded point detection algorithm
with arguments A and $m(A)$.

1.2.2 · IF an unbounded domain point is found THEN

1.2.2.1 · Set B 's boundedness to unbounded.

ELSE

1.2.2.2 · goto 3.

ELSE IF A contains a feasible point, a .

1.2.3 · Apply the unbounded point detection algorithm
with arguments A and a .

1.2.4 · IF an unbounded domain point is found THEN

1.2.4.1 · Set B 's boundedness to unbounded.

ELSE

1.2.4.2 · goto 3.

2 · IF a division by a point a such that $-\delta_{unbounded} \leq Q(a) \leq \delta_{unbounded}$ is to occur THEN

2.1 · Determine a 's feasibility.

2.2 · IF a is feasible THEN

- 2.2.1 · Determine $\triangleleft(a_i)$ and $\triangleright(a_i)$, the next smallest and largest representable point from a_i , the i^{th} component of the point a .
- 2.2.2 · Use the $\triangleleft(a_i)$'s and $\triangleright(a_i)$'s to generate a box,

$$A = [(\triangleleft(a_1), \dots, \triangleleft(a_n), (\triangleright(a_1), \dots, \triangleright(a_n))].$$
- 2.2.3 · Apply the unbounded point detection algorithm with arguments A and a .
- 2.2.4 · IF an unbounded domain point is found THEN
 - 2.2.4.1 · Set B 's boundedness to unbounded.
 - ELSE
 - 2.2.4.2 · goto 3.

3 · Indicate an arithmetic exception has occurred, division by zero and apply the appropriate error handler to maintain continuation of 'normal' GOS-BB operation by providing a correct bound.

The unbounded point detection algorithm, UPDA, is as follows:

- 1 · Use A and x_{ubp} , the input parameters to determine if $V : 1$ or $V : 2$ are true.
- 2 · IF either $V : 1$ or $V : 2$ are true THEN
 - 2.1 · x_{ubp} is an unbounded domain point.
 - 2.2 · Output x_{ubp} as an unbounded domain point.

7.8.4 Evaluation

Consider the problem (see Figure 7.2):

$$\begin{aligned}
 f(x) &= \frac{p(x)}{q(x)} = \frac{-1}{(x_1 - 2)^2 + (x_2 - 4)^2} \\
 s.t. \quad &0 \leq x_i \leq 10 \text{ for } i = 1, 2
 \end{aligned}$$

For this problem whenever a box or point whose range of functional values lies within $-\delta_{\text{unbounded}}$ to $\delta_{\text{unbounded}}$ occurs, this box or point becomes a candidate unbounded domain point. Subsequent testing of the box or point will determine if an unbounded domain point exists. For example, if a box, B , is found such that $0 \leq Q(B) \leq \delta_{\text{unbounded}}$ then $Q(B)$ is a small box about the point $(2, 4)$. As $P(B) < 0 \forall B$, the box B can then be shown to contain an unbounded domain point (within the accuracy determined by $\delta_{\text{unbounded}}$).

Figure 7.2: $f(x) = \frac{-1}{(x_1-2)^2 + (x_2-4)^2}$

7.9 Formulation Aids

In this section we examine how the flatness of the objective function about the global minimum, a form of sensitivity analysis, and snap shot evaluations can be provided.

7.9.1 Flatness of the Objective Function

If a GLOBAL MINIMUM outcome occurs, then a bound on the 'flatness' of the objective function about the global minimum is desirable. By the *flatness* of a point we mean the relative flatness of a function in a region centered about that point. The *flatness factor* of a point centered on a region is the ratio of the range of the functional values over the width of the region. The larger the value of the flatness factor, the less 'flat' is the function at that point.

The flatness is related to the derivative of the function at the point, but differs as the derivative is a local measure, while the flatness is an area measure. If we take the range of the values of the derivative over the region and divide by the width of the region to get a 'flatness' measure, we do not necessarily get an indication of the flatness of the surface. For example, the function may have a lot of small hills with sharp slopes, for which the measure using the range of the derivative would produce a large flatness factor, whereas the actual range of the function does not vary greatly over the region.

We formally define the *flatness factor* of a function at a point, x , as the range of the objective function over a n -cube (a box with all sides of equal length) centered at x , divided by the length of the edge of the n -cube. By determining a bound on the flatness factor for a point using n -cubes of several sizes, information about the behaviour of the function can be obtained. For example, for a function with small hills of sharp slopes, a small box will give a large flatness factor and as the size of the box increases the flatness factor will decrease. On the other hand, if a small flatness factor occurs for a small box and if as the size of the box increases so does the flatness factor, then this indicates that the region about the point is flat in a small region but over a larger region it is not.

By default, when the GOS-BB terminates with a feasible Incumbent Solution, three boxes of widths 2.0×10^{-6} , 2.0×10^{-3} , and 2.0 centered on the Incumbent Solution are used in order to determine bounds on the flatness factor. The accuracy of the bound on the range of the function is dependent upon the methods used to bound the range of the function and on the size of the region. Thus the bound on the flatness factor will be more accurate as the size of the box decreases.

Variations on Flatness Measures

The flatness factor can be used to explore the 'shape' of the function by simply applying it to several regions about an area of interest. Thus the procedure could be used as the basis of an exploratory tool, allowing the modeller to examine specified regions. The Set of Boxes which remain after the termination of the application of GOS-BB may possibly be used to help identify regions of interest.

Other possible variations are to determine a bound on the flatness factor for the largest feasible box centered about the Incumbent Solution or for the largest feasible box which contains the Incumbent Solution. Both of the above variations can yield information on the relative size and shape of the feasible region about the Incumbent Solution (or any other point).

Example of Flatness Factors

For example: consider the following problem:

$$\min \quad 4x_1^2 - 2.1x_1^4 + (1/3)x_1^6 + (x_1x_2) - 4x_2^2 + 4x_2^4$$

$$\begin{aligned}
 \text{s.t.} \quad & -2x_1^4 + 2 - x_2 \leq 0 \\
 & 2 - (x_2 + 2)^2 + x_1^2 \leq 0 \\
 & \cos(x_1 + 1) + (x_1 - 1)^2 - x_2 - 6 \leq 0
 \end{aligned}$$

At the end of the solution process, for the boxes which contain the incumbent solution, a value of $2.17 * 10^{-7}$ is obtained for the flatness factor, for the largest feasible box that can be found centered on the incumbent solution (width of 1.316) a value of 13.7 is obtained for the flatness factor, and for a set of 3 boxes centered on the Incumbent Solution the following flatness factors were obtained:

width of box	$2.0 * 10^{-6}$	$2.0 * 10^{-3}$	2.0
flatness factor	8.59	0.017	22.15

indicating that the function has a 'medium' sized relatively flat region (with small hills) about the Incumbent Solution within a larger less flat region.

7.9.2 Sensitivity Analysis

What is desired in a sensitivity analysis is the set of global minimum values (and associated points) that are possible if a parameter, c , is allowed to vary by some specified amount, that is, $c \in C$. A lower bound on the set of global minimum values (and the least global minimum) can be found by letting c become a variable, bounded to stay within C . The difficulty is in determining an upper bound as well as the values of any other global minimum values that may exist.

A method by which bounds on the sensitivity of the problem to a parameter change can be obtained is to replace the parameter by an interval which spans the variability of the change. The modified problem is then solved and the results are compared to the results of a non-modified problem. Several parameters can be modified if a group effect is desired. However, problems arise since interval analysis in general provides poor bounds for large intervals, thus restricting the intervals used to small sizes in order to ensure that the results adequately represent the sensitivity of the problem to a parameter (see [97]). Thus this method will produce correct bounds, but not likely sharp results. As they are correct, the bounds produced can be used to reduce the problem space from which a search for sharper bounds can be undertaken.

A method has been developed (see [43, 42, 97, 19, 20, 16]) which attempts to determine tighter bounds by solving a set of sub-problems. However, it is only valid if certain conditions are true and thus may not be applicable. Other attempts at providing sensitivity analysis for Global Optimization Problems are similar to 'ESTAC' [118], which requires repeated applications of slightly perturbed variations and is of limited use. Thus, a form of sensitivity analysis is possible, although more research is needed in order to assure sharp bounds (see Section 10.3).

7.9.3 Snap Shot Evaluations

Some problems may require a large amount of time, either due to the size of the problem or to the particular nature of the problem (for example, a flat function which forces a large number of boxes to be generated, or an infinite number of global minima that 'span' a large non-linear region, etc.). Examining the state of the current solution at regular intervals via a 'snap shot' evaluation may yield useful information as formulation aids can be applied to the current solution to increase the modeller's knowledge about the characteristics of the problem, and/or to evaluate the solution process with the goal of improving its performance.

For example, if no feasible point has been determined, the maximum cardinality of the set of boxes remains relatively constant at a small number, and the size of the smallest box is not 'too' small, then if this status does not change over a large period of time this indicates that the region defined by the set of boxes is slowly being removed as being infeasible. Furthermore, the lack of any really small boxes indicates that once boxes are subdivided into a small enough size, then their infeasibility can be verified and they are removed. This suggests that either the problem is infeasible or that the objective function is not directing the search for a feasible point effectively. This indicates that a change to a different objective function, such as a Phase I objective function, may be desirable.

In addition, the information gleaned from the application of formulation aids at periodic intervals may be useful as input to a control system that guides the solution process. More research in this area is needed to determine the usefulness of a control system and to determine what information would be required by the system (see Section 10.3).

In addition to the flatness factor for the region about the current Incumbent Solution, additional snap shot information can be useful from a modellers perspective:

- the percentage of the final 'volume' of the Set of Boxes relative to the 'volume' of the

Initial Box. The ‘volume’ of a box is determined by the product of the widths of the edges of the box. If a box has an edge of width zero, then this measure of volume fails and alternative measures are then necessary.

- the values of the Incumbent Solution, Lower Bound.
- the number and/or volume of infeasible boxes removed.
- the number and/or volume of boxes fathomed.
- the number of times a local optimization method was applied and the number of times an improvement in the Incumbent Solution occurred.

For a specified set of boxes, B , the following items are also useful:

- the smallest containing box, that is, the smallest interval that can be generated which contains the set B .
- the percentage of the ‘volume’ of the set B relative to the ‘volume’ of the containing box for the set.
- the maximum and minimum widths and/or volumes of the boxes of the set B .
- the cardinality of B .
- the smallest and largest boxes (by width and/or volume) that are removed due to infeasibility or by fathoming by the Incumbent Solution.

where the set B could be one of:

- the Set of Boxes. In the actual implementation of GOS-BB, the Set of Boxes is comprised of three sets, $SB_{\geq}$, $SB_{<}$, and SB_a (see Section 9.1.1) which correspond to the boxes which are greater than or equal to an *a priori* determined minimum box size, those that are less than the minimum box size and those that lie outside of the effective search region (see Section 7.1.2) and are smaller than the minimum box size representable.
- the boxes which include the Incumbent Solution.

- the boxes which form a maximal set of connected boxes (a connected component). Two boxes are connected if they share at least one point. Thus, a set of boxes can be partitioned into sets of connected components.
- the boxes which have a finite (infinite) lower bound.
- the feasible (infeasible) boxes or the boxes with unknown feasibility.
- the ‘best’ n boxes from the Set of Boxes, where ‘best’ can be determined by the least lower bound, by least value of a phase I objective function (for infeasible problems), etc.

These items can be obtained by analyzing the set *a posteriori* or by maintaining the necessary values, counters, etc. within the solution process. Thus, the above formulation aids can be applied (when applicable) to any one or a combination of the above sets.

For example, the connected components of $SB_{\geq}$ can be obtained by applying the appropriate algorithm to the set. We note that it can only be done *a posteriori* to the Set of Boxes as this set can lose elements, but it can be done during the solution process for the set of infeasible boxes, fathomed boxes, etc., thus removing any need to store these sets of boxes.

As stated, the above aids can be used to determine the shape of the region defined by a set of boxes. For example, if the final volume is very small relative to the volume of the containing box for a large set of small boxes, then the region is characterized by small disjoint regions or by a long thin feasible region. By determining the connected components of the Set of Boxes more detail on the shape of the feasible region can be determined.

7.10 Indeterminate Outcomes

If GOS-BB finishes before an outcome of INFEASIBLE. DOMAIN UNBOUNDED or GLOBAL MINIMUM can be determined, then an INDETERMINATE outcome is indicated. In this event, the current Set of Boxes at termination can be output, along with the value of the incumbent solution and Lower Bound. The ‘best’ of these boxes can then be output as a set of promising sub-regions. ‘Best’ could be defined in terms of the value of the lower bound, or a weighted measure of the lower bound and the difference from feasibility (the amounts that the constraints must be relaxed to make the box feasible) if no feasible point has been identified, etc.

In addition, formulation aids (see Section 7.9) can be applied to the final Set of Boxes to determine the shape of the Set of Boxes and to further help to localize the 'cause' of the indeterminacy by determining subsets of boxes which do not have a finite lower bound on the range of the functional values, connected components, etc.

Chapter 8

Branching and Bounding Strategies for GOS-BB

In this Chapter we examine bounding and branching strategies to be used within GOS-BB.

8.1 Bounding Functions

In this section we examine a set of bounding functions and develop a bounding function strategy, adaptive bounding, to be used within GOS-BB.

8.1.1 Basic Set

We use two types of bounding functions, the natural interval extension (NIE), and the modified Krawczyk centered method (MK) in conjunction with two function representation schemes, a Horner-like scheme (H) and a *power* scheme (P) as described below. In all, four bounding functions are determined for each function. We label these as MK-H, MK-P, NIE-H and NIE-P for ease of reference.

Representation schemes

We use two representation schemes. The first, a Horner-like scheme, attempts to produce a representation which has minimal computational expense (in terms of arithmetic operators), and the second, the power scheme, attempts to produce a representation which minimizes

the occurrences of the variables and maximizes the occurrence of terms with exponents > 1 . These representations are only informally defined.

The Horner-like scheme is developed as follows. Initially a Horner expression for the variable of highest degree is determined, followed by the attachment of as many of the other variable terms as possible to this expression. If any terms are left over, the process repeats itself (on the remaining terms) until all terms are attached. The rationale is that since the Horner scheme (for a polynomial of one variable) provides a representation which minimizes the computational expense, an extension to polynomials of more than one variable will also tend to provide a representation of smaller computational expense.

For example: consider the function

$$4x_1^2 - 2.1x_1^4 + (1/3)x_1^6 + (x_1x_2) - 4x_2^2 + 4x_2^4$$

the variable x_1 attains the highest degree and a Horner scheme for x_1 would be

$$x_1 * (x_1 * (x_1^2 * (x_1^2/3 - 2.1) + 4))$$

attaching any x_2 terms results in

$$x_1 * (x_1 * (x_1^2 * (x_1^2/3 - 2.1) + 4) + x_2)$$

with the left over terms of $-4x_2^2 + 4x_2^4$; for which a Horner scheme can be found to result in the final representation

$$x_1 * (x_1 * (x_1^2 * (x_1^2/3 - 2.1) + 4) + x_2) + 4 * x_2^2 * (x_2^2 - 1)$$

The power scheme aims to minimize the occurrence of a variable as well as maximize the higher exponents of terms in the expression (so as to take advantage of the modifications used within the modified Krawczyk centered form (see Section 6.2)). It does so by attempting to complete squares, cubics, etc. as this will minimize the occurrence of a variable as well as provide terms with exponent > 1 . If no squares, etc., can be completed, then the representation essentially becomes a sum of the exponential terms.

For example: consider the function

$$\begin{aligned} & x_3 * (x_3 * (x_3^2 + 1.0) - 2.0) + x_2 * (x_2 * (x_2^2 + 1.0) - 2.0) + 3.0 + \\ & x_1 * (x_1 * (x_1 * (x_1 - 2.0) + 4.0) - 2.0 * (1.0 + x_2^2 + x_3^2)) \end{aligned}$$

completing the squares produces the following representation

$$\sum_{i=1}^3 [(x_i - x_i^2)^2 + (x_i - 1)^2]$$

Both representations are designed primarily with polynomials in mind. Any rational functions are first put into rational form, that is $f = p/q$ where p and q are polynomial functions. Alternatively, if this is too difficult, a sum of rational functions, $f = \sum_i p_i/q_i$ is used. This is followed by an attempt at the reduction of roots common to both p_i 's and q_i 's. Finally the scheme is applied to each of the p_i 's and q_i 's to determine the representation.

The two representations are used in an attempt to provide opposite approaches. Other representations are possible as well as procedures by which a representation can be evaluated to minimize the effect of multiple occurrences of variables (see Lodwick [70]).

8.1.2 Evaluation

We evaluate all four methods by examining the relative order of the lower bounds produced by the bounding function methods over a large set of boxes. The boxes have been produced by the GOS-BB algorithm as it solves a given problem. Each box produced is tested against each of the four methods and the relative performance is evaluated against the size of the boxes. The smallest inclusion for the four methods is used as the resultant bound for the range (see Section 7.4).

Let the triple (a, b, c) represent the number of times the bounding function produced a lower bound (equal, greater, less) than the specified bounding function. We tested against a large set of problems. A representative sample of the Six Hump Back and Schwefel No. 3.1 problems (see Section 5) is given in Table 8.1.

An analysis of the complete set of result shows the following pattern. For large boxes, no single method outperforms the other methods. As the box size decreases, one (or more) methods dominate the other methods. No method dominates all of the other methods for all of the test problems.

8.1.3 Adaptive Bounding Methodology

The analysis of the four bounding methods suggests the following strategy. For a set of bounding functions, *SBF*, and a given problem, determine a cutoff box size, *cbs*, a box size

Six Hump Back Camel - boxes of width < 0.1			
Bounding Function	Specified Bounding Function		
	NIE-P	MK-H	MK-P
NIE-H	(248, 1392, 0)	(248, 0, 1392)	(248, 0, 1392)
NIE-P		(172, 0, 1468)	(172, 0, 1468)
MK-H			(1176, 464, 0)
Six Hump Back Camel - boxes of width ≥ 0.1			
Bounding Function	Specified Bounding Function		
	NIE-P	MK-H	MK-P
NIE-H	(0, 466, 1)	(0, 173, 294)	(227, 240)
NIE-P		(0, 79, 388)	(0, 91, 376)
MK-H			(1, 428, 38)
Schwefel No. 3.1 - boxes of width < 0.1			
Bounding Function	Specified Bounding Function		
	NIE-P	MK-H	MK-P
NIE-H	(12, 0, 132)	(12, 0, 132)	(12, 0, 132)
NIE-P		(114, 30, 0)	(115, 29, 0)
MK-H			(120, 0, 24)
Schwefel No. 3.1 - boxes of width ≥ 0.1			
Bounding Function	Specified Bounding Function		
	NIE-P	MK-H	MK-P
NIE-H	(0, 0, 101)	(1, 57, 43)	(1, 55, 45)
NIE-P		(0, 101, 0)	(0, 101, 0)
MK-H			(3, 0, 98)

Table 8.1: Bound Comparissons for the 4 basic bounding set

such that all boxes smaller than the cutoff are dominated by at least one or more methods. This cutoff box size is determined on the fly by the evaluation of the boxes generated by the GOS-BB. Once a cutoff has been determined, use a dominant method to evaluate any box less than the cutoff size and use all of the bounding methods for boxes which are larger than the cutoff size. If the cutoff box size is correctly determined then the bound determined

for every box will be the best possible for SBF and will require less computational expense than applying all of the methods to all of the boxes (which would also determine the best bound for every box).

Adaptive Bounding Algorithm

The adaptive bounding algorithm maintains a set of cutoff boxes, SCB until the cutoff box size, cbs , has been determined for a function, f . Each new box is tested to see if it can enter the set SCB . If the entry succeeds then a second test is performed to see if a valid cutoff exists. If this test is successful, the cutoff size is set to the size of the largest box in SCB and the best bounding function, BBF , is chosen as the dominant bounding function of SCB of least computational cost. If the entry fails then the set SCB is updated to maintain the following conditions:

- If a box is an element of SCB then all other boxes which have been produced by GOS-BB which are of the same size or of a lesser size must also be in the set.
- At least one bounding function produces the best lower bound for all of the boxes in the set.

The size of a box, B , is determined by its width, $w(B)$. The cardinality of SCB is denoted by $card(SCB)$ and the difference in the width of the largest and smallest boxes of SCB is denoted by $dw(SCB)$. A heuristic test is used to determine if a valid cutoff exists. Let mc and mdw be *a priori* determined values representing a minimum cardinality and minimum difference in width. If $card(SCB) > mc$ and $dw(SCB) > mdw$, then a valid cutoff is said to exist.

Let SBF denote the set of bounding functions available for the function, f , (with an ordering of the bounding functions by computational expense), and $SBBF_\Phi$ denote the set of best bounding functions for a set of boxes Φ . The actual box, B , is not stored in SCB as all that is required is $w(B)$ and $SBBF_B$.

Initialize the parameters as follows:

- Set $SCB = \emptyset$.
- Set $SBBF_{SCB} = \emptyset$
- Set the largest width (the width of the largest box in SCB), $lw = \infty$, .

- Set the best bounding function, $BBF = \emptyset$.

For each box, B , for which a bound is to be determined:

1 · Determine width of B , $w(B)$.

2 · IF $BBF = \emptyset$

2.1 · Apply each bounding function of SBF to B .

2.2 · The intersection of the bounds produced in 2.1 is output as a bound on f over B .

2.3 · Apply the determine cutoff procedure to SCB with B and the set of bounds produced in 2.1.

ELSE IF $w(B) \leq lw$

2.4 · Apply BBF to B and output the bound as a bound on f over B .

2.5 - OPTIONAL · Test:

2.5.1 · apply each bounding function of SBF to B .

2.5.2 · determine the set of best bounding functions, $SBBF_B$.

2.5.3 · IF $BBF \notin SBBF_B$ THEN

2.5.3.1 · Set $BBF = \emptyset$.

2.5.3.2 · Update set SCB .

2.5.3.3 · goto 2.1.

ELSE

2.5.3.4 · $SBBF_{SCB} = SBBF_B \cap SBBF_{SCB}$

ELSE

2.6 · Apply each bounding function of SBF to B .

2.7 · The intersection of the bounds produced in 2.6 is output as a bound on f over B .

If a cutoff box size has been determined (indicated by $BBF \neq \emptyset$) then the current value of lw is used as the cutoff box size. If, in addition, the width of a box is less than or equal to lw then a bound is determined by applying BBF to the box. Otherwise a bound is determined by applying each of the bounding functions available to the box and taking the intersection of the bounds as the resultant bound. If a cutoff box size has not been determined then if the width of the box is less than or equal to lw the determine cutoff procedure (see below) is applied to update SCB and to check if a valid cutoff can be determined.

The optional test can be implemented to improve confidence in the determination of a good cutoff value. The test checks to see if the current BBF is still valid by determining SBF_B . If BBF is still valid, then the current set $SBBF_{SCB}$ is updated to reflect the information gained from the new box, if not, then the set SCB is updated by removing the minimum number of boxes required such that all of the boxes remaining in SCB are dominated by at least one bounding function method. This condition is relaxed from the previous conditions on SCB , necessitated by the fact that some boxes may have been applied only to the BBF and cannot be reapplied to the complete set SBF as this box may no longer exist.

Possible conditions as to when to test are after an *a priori* number of boxes less than the cutoff have been processed or when boxes of width smaller than a percentage of the cutoff value are processed (this value can then be updated depending upon the results of the test). However, as the accuracy of the bounding functions increases as the size gets smaller, a bad choice of best bounding function will likely result in only a small increase in the associated error. In addition, any method that has been chosen has already performed as well if not better than the other methods and thus is unlikely to be the poorest method for these smaller boxes. Thus the optional test is likely to be used more as a means to test the adaptive bounding algorithm, without resorting to checking every box.

DETERMINE CUTOFF PROCEDURE

This algorithm maintains the set SCB by using a procedure called determine entry (see below) to see if a box, B , can be entered into SCB . If the entry fails, then all boxes in SCB which are greater in width than B are removed and another attempt at entering B is made. If this also fails then all boxes in SCB which are greater than or equal in width to $w(B)$ are removed. When boxes are removed from SCB the set $SBBF_{SCB}$ is updated to determine the maximum number of dominating bounding function methods for all of the

remaining boxes in SCB (by using the set of best bounding functions associated to each box in SCB). When an entry succeeds, the determine valid cutoff procedure (see below) is called to determine if SCB meets the requirements for the existence of a valid cutoff box size and after this test the procedure stops. The current largest width, lw , is an upper bound on a valid cutoff box size and is updated as appropriate.

1 · Determine entry for B into SCB .

2 · IF entry succeeds THEN

2.1 · Determine valid cutoff procedure.

2.2 · Return.

ELSE

2.3 · Remove all boxes in SCB whose size is $> w(B)$.

2.4 · $lw = w(B)$.

2.5 · Update $SBBF_{SCB}$.

2.6 · Determine entry for B into SCB .

2.7 · IF entry succeeds THEN

· 2.7.1 · Determine valid cutoff procedure.

2.7.2 · Return.

ELSE

2.7.3 · Remove all boxes in SCB whose width is $\geq w(B)$.

2.7.4 · $lw = \triangleleft(w(B))$, the next smallest representable number on the automaton.

2.7.5 · Update $SBBF_{SCB}$.

DETERMINE ENTRY PROCEDURE

This procedure attempts to enter a box B into SCB , while maintaining the conditions on the set. It does so by finding the intersection of the current set of best bounding functions for SCB , $SBBF_{SCB}$ and the the best bounding functions for B , $SBBF_B$. If none are in

common the box is refused entry. If at least one best bounding function is found in common, the box is entered and the current set of best bounding functions for SCB is updated to the set of best bounding functions in common.

- 1 · IF $card(SCB) = 0$ AND $w(B) \leq lw$
 - 1.1 · Enter B into SCB .
 - 1.2 · Determine the set $SBBF_B$.
 - 1.3 · $SBBF_{SCB} = SBBF_B$.
- 2 · ELSE IF $w(B) \leq lw$
 - 2.1 · Determine the set $SBBF_B$.
 - 2.2 · IF $SBBF_{SCB} \cap SBBF_B \neq \emptyset$
 - 2.2.1 · $SBBF_{SCB} = SBBF_B \cap SBBF_{SCB}$
 - 2.2.2 · Enter B into SCB .
 - ELSE
 - 2.2.3 · Entry fails.

The set $SBBF_B$ is determined by finding the set of bounding functions with the highest lower bound for the box B . As each bounding function provides correct bounds, this value is the best lower bound that can be determined from the set of bounding functions.

DETERMINE VALID CUTOFF PROCEDURE

This procedure checks to see if the conditions for a valid cutoff have been met. Although the application of such a procedure is not required at step 2.7.1 as the cardinality has been reduced, it is included in the case that different validation criteria are used than a minimum cardinality and minimum difference in width.

- 1 · IF cardinality of $SCB \geq mc$ AND the $dw(SCB) \geq mdw$ THEN
 - 1.1 · Set BBF to be the bounding function of $SBBF_{SCB}$ of least computational cost.

As the set BBF has been ordered by its computational cost, 1.1 can be carried out by using this order. In practice, the relative computational expense of a set of bounding functions is difficult to determine and as such a rough estimate must be used.

Evaluation

The adaptive bounding algorithm (ABA) is tested in two phases. In the first phase, the ABA is used to select the bounding function, but all of the bounds are calculated for all of the boxes, in order to determine whether ABA has chosen the correct bounding function. The following data is recorded for each test: a) the total number of times ABA did not generate the best bound, b) the number of boxes that the ABA determined best bounding function was applied to, and c) the total number of boxes generated, and given by the triple (not best, tested, total). Testing was carried out with $mc = 200$ and $mbw = 1.0 * 10^{-3}$. The results are summarized in Tables 8.2 and 8.3.

Problem Name	number of boxes			Problem Name	number of boxes		
	not best	tested	total		not best	tested	total
FP 3.2, S1	0	36,510	36,711	FP 3.2, S2 10%	0	77,434	77,635
FP 3.2, S2	0	60,646	60,847	FP 3.2, S2 30%	0	58,899	59,100
FP 3.2, S3	0	58,684	58,885	FP 3.2, S2:50%	0	44,019	44,220
FP 4.7, S1	0	14,586	14,787	FP 4.7, S2 10%	0	8,814	9,015
FP 4.7, S2	0	20,844	21,045	FP 4.7, S2 30%	0	11,599	11,800
FP 4.7, S3	0	20,844	21,045	FP 4.7, S2:50%	0	11,720	11,921
THC, S1	0	39	739	THC, S2:10%	0	960	2,723
THC, S3	0	0	753	THC, S2:30%	0	18	893
THC, S4	0	0	735	THC, S2:50%	0	38	816
SHB, S1	0	36,510	36,711	SHB, S2:10%	0	77,434	76,635
SHB, S3	0	60,646	60,847	SHB, S2:30%	0	58,899	59,100
SHB, S4	0	58,684	58,885	SHB, S2:50%	0	44,019	44,220
UB 8, S1	0	19,761	19,999	UB 8, S2:10%	0	19,764	19,998
UB 8, S3	0	19,407	19,999	UB 8, S2:30%	0	19,764	19,998
UB 8, S4	0	14,772	15,000	UB 8, S2:50%	0	19,764	19,998

where the S's refer to the subdivision method used (see Section 8.2).

Table 8.2: Validity tests for ABA

Problem	No. of variables	No. of constraints	<i>BBF</i> found	approx. <i>cbs</i>
FP 3.2	5	6	NIE-H	3
FP 4.7	2	1	MK-H	21
THC	2	0	NIE-H	$5.0 * 10^{-3}$
SHB	2	0	MK-H	$2.0 * 10^{-1}$
UB 8	5	0	NIE-H	$6.6 * 10^{99}$

where FP denotes the Floudas and Pardalos test set, SHB and THC are the Six Hump Back and Three Hump Camel problems, and UB 8 is the Unbounded 8 problem (see Section 5).

Table 8.3: *BBF*'s and *cbs*'s for ABA

The above results show that the determination of a correct cutoff box size is possible and that the cutoff box size determined varies from function to function. It is likely that the values for *mc* and *mdw* will depend on the set of bounding functions used and on the dimension of the problem. More testing and use is required to determine how to properly choose these values (see Section 10.3).

In the second phase, the ABA is tested against the four bounding functions (NIE-H, NIE-P, MK-H, MK-P). The results are summarized in Tables 8.4 and 8.5. A description of the test problems used can be found in Section 5.

From the results we see that the ABA performs well against the individual bounding functions, being, in general, about the second best method for a given problem. When compared against all of the test problems, its performance is better than the NIE-H, MK-H, and MK-P methods and is roughly equal to that of the NIE-P method. The importance of using a good bounding function is shown by the cases in which an improper method fails to terminate.

Thus the two phases of testing indicate that the general principle of the adaptive bounding methodology can improve the accuracy of the bounds as well as improve the performance of a GOS-BB when compared to a single bounding function. It performs better than a set of bounding functions (applied to all boxes), as fewer function evaluations are required once a best bounding function has been determined. Therefore, as a means to determine *a priori* a

Problem Name	Bounding Function	No. of Iterations	No. of function Evaluations	Max. No. Boxes	Time (sec)
SCH 3.1	NIE - H †	112,887	1,077,373	1,000,000	1261.37
SCH 3.1	NIE - P	72	487	1	0.44
SCH 3.1	MK - H	9,761	58,853	364	269.26
SCH 3.1	MK - P	9,106	54,875	364	203.57
SCH 3.1	ABA	72	792	1	2.22
THC cs12	NIE - H	121	25,118	42	19.11
THC cs12	NIE - P	1,778	97,564	785	43.90
THC cs12	MK - H	1,721	96,553	759	55.18
THC cs12	MK - P	1,721	96,553	759	53.76
THC cs12	ABA	194	27,776	76	22.67
UB 8 c21	NIE - H †	10,154	61,109	10,000	121.85
UB 8 c21	NIE - P	112	776	100	1.09
UB 8 c21	MK - H †	9,999	60,125	10,000	306.69
UB 8 c21	MK - P †	9,999	60,125	10,000	285.79
UB 8 c21	ABA	112	1,383	100	5.05
UB 2 c25	NIE - H †	68,101	434,111	30,000	398.12
UB 2 c25	NIE - P	11,489	41,568	2,803	44.35
UB 2 c25	MK - H	11,515	41,711	2,808	113.32
UB 2 c25	MK - P	11,515	41,711	2,808	89.04
UB 2 c25	ABA	11,489	42,171	2,803	45.50

† - For this bounding function, the method terminated after the preset maximum number of boxes was achieved.

Table 8.4: Performance tests for ABA

good bounding function for a given function is not known, the use of the ABA in conjunction with a set of bounding functions is a good strategy to use with GOS-BB.

Problem Name	Bounding Function	No. of Iterations	No. of function Evaluations	Max. No. Boxes	Time (sec)
SHB	NIE - H	22,871	200,347	15,778	195.98
SHB	NIE - P	93,248	811,893	63,106	906.62
SHB	MK - H	579	3,627	62	8.41
SHB	MK - P	662	4,193	84	9.49
SHB	ABA	389	3,867	54	8.20
UB 7	NIE - H	56	63	48	0.35
UB 7	NIE - P	56	63	48	0.33
UB 7	MK - H	56	624	48	0.72
UB 7	MK - P	56	624	48	0.59
UB 7	ABA	56	964	48	1.08
UB 2 cs2	NIE - H	71	1,023	7	0.31
UB 2 cs2	NIE - P	44	628	13	0.20
UB 2 cs2	MK - H	71	1,152	7	0.76
UB 2 cs2	MK - P	71	1,152	7	0.69
UB 2 cs2	ABA	44	1,073	13	0.90

Table 8.5: Performance tests for ABA

8.2 Subdivision Strategies

In this section we develop four methods by which the subdivision of a box can be undertaken. The four methods are:

- S1 Classic division into two equal parts.
- S2 Division by narrow band about a local minimum.
- S3 Division into thirds, by functional value.
- S4 Division into thirds, by functional and slope values.

These four methods divide the box by dividing the variable that corresponds to the longest edge of the box (hereafter called the *splitting variable*) into two or three parts. Thus,

the width of the boxes will become smaller as the algorithm progresses. Ties are broken by choosing the first variable (from x_1 to x_n).

8.2.1 S1: Classic Division into Two Equal Parts

In this method, the box is divided into two equal parts, by partitioning the splitting variable in half. For example, the box $([-12, 3], [0, 1], [0, 15])$ is split into the two boxes $([-12, -4.5], [0, 1], [0, 15])$ and $([-4.5, 3], [0, 1], [0, 15])$.

8.2.2 S2: Division by Narrow Band about a Local Minimum

In this method a search is made for a local minimum. If one is found, then a narrow band of *a priori* determined width is created about the local minimum by partitioning the splitting variable equidistant about the corresponding local minimum variable value. If this band exceeds the space of the original box, the band is truncated so that it fits. If no local minimum can be found, then the box is split by the classic division into two equal parts. Thus, either two boxes or three boxes are created.

For example, consider an *a priori* determined width of 2% and a local minimum found at the point $(-1.4, 0.4, 13.13)$ then the width of the narrow box is determined to be 2% of $15 = 0.30$ and the box is split into the three boxes $([-12, -1.55], [0, 1], [0, 15])$, $([-1.55, -1.25], [0, 1], [0, 15])$ and $([-1.25, 3], [0, 1], [0, 15])$. If a local minimum was found at the point $(-11.9, 0.4, 13.13)$ for the box $([-12, 3], [0, 1], [0, 15])$ then the box would have been split into the two boxes $([-11.75, 3], [0, 1], [0, 15])$ and $([-12, -11.75], [0, 1], [0, 15])$.

8.2.3 S3: Division into Thirds, by Functional Value

In this method, two points are evaluated for their functional values. The two points are determined as follows. Partition the splitting variable into three equal parts. Take the midpoints of all of the non-splitting variables along with the two center points obtained from the partitioning of the splitting variable to generate two points labelled a and b ($a < b$ with no loss of generality). The point which has the least functional value is used to partition the splitting variable into two parts. Ties are arbitrarily decided in favour of the point b .

For example, consider the box $([-12, 3], [0, 1], [0, 15])$. The two points are determined as $a = (-7, 0.5, 7.5)$ and $b = (-2, 0.5, 7.5)$. If the former has a lower functional value then the

box is split into the two boxes $([-12, -7], [0, 1], [0, 15])$ and $([-7, 3], [0, 1], [0, 15])$.

8.2.4 S4: Division into Thirds, by Functional and Slope Values

In this method, two points are evaluated for their functional and slope values. The two points are determined as in the above method (division into thirds, by functional value method). Label the two points a and b ($a < b$), and the slopes by $f'(a)$ and $f'(b)$. This method differs from the above in that the partition is in some cases into three parts. Thus the partitions are made at a , or at b , or at a and b .

The choice of which point(s) (a , b , or a and b) to use to partition the splitting variable is made using the following rules:

case	point(s) chosen
$f'(a), f'(b) \geq 0$ and $f(a) < f(b)$	a
$f'(a), f'(b) \leq 0$ and $f(a) > f(b)$	b
otherwise	a and b

For example, consider the box $([-12, 3], [0, 1], [0, 15])$. The two points are determined as $a = (-7, 0.5, 7.5)$ and $b = (-2, 0.5, 7.5)$. If $f'(a) > 0$, $f'(b) = 0$ and $f(a) < f(b)$ then the box is split into the two boxes $([-12, -7], [0, 1], [0, 15])$ and $([-7, 3], [0, 1], [0, 15])$. If $f'(a) > 0$, $f'(b) = 0$ and $f(a) = f(b)$ then the box is split into the three boxes $([-12, -7], [0, 1], [0, 15])$, $([-7, -2], [0, 1], [0, 15])$ and $([-2, 3], [0, 1], [0, 15])$.

8.2.5 Evaluation

We note that deletion filtering involves a different objective function (a phase I) and a different goal (obtain a feasible point) than the main problem. As such, we examine the different subdivision methods first for 'normal' operation and secondly for use within deletion filtering.

Normal GOS-BB Operation

We use the following set of problems:

Problem	No. of		Outcome
	variables	constraints	
FP 3.2	5	6	GLOBAL MINIMUM
FP 4.7	2	1	GLOBAL MINIMUM
Ex. 1	1	2	GLOBAL MINIMUM
SHB	2	0	GLOBAL MINIMUM
UB 2	2	0	DOMAIN UNBOUNDED
UB 2 c11	2	1	DOMAIN UNBOUNDED
UB 8 cs8	5	2	INFEASIBLE
THC cs12	2	18	INFEASIBLE

to evaluate the subdivision methods, using the subdivision methods of $S1$, $S1 - M$, $S2$, 30%, $S2$, 50%, $S2$, 70%, $S3$, $S3 - M$, $S4$, and $S4 - M$, where M indicates that for each new box entered into the Set of Boxes, the midpoint is used as a starting point for a local optimization method algorithm. As the $S2$ method uses an application of a local optimization method to find a local minimum point when each box is split, applying a local optimization method as described for the methods $S1$, $S2$ and $S3$ will roughly give these methods the same amount of local optimization method usage. FP refers to the Floudas and Pardalos Test Set and Ex. 1 is the Example 1 problem, SHB is the Six Hump Back Camel problem, UB 2 is the Unbounded 2 problem and UB 2 c11 is the Unbounded 2 c11 problem, UB 8 cs8 is the Unbounded 8 cs8 problem, and THC cs12 is the Three Hump Camel cs12 problem (see Appendix B).

The results of the testing is summarized in Tables 8.6, 8.7, and 8.8, where for problems with a GLOBAL MINIMUM outcome, in parenthesis after the number of iterations is a value that indicates when and by what means the incumbent solution was obtained. The letter before the dash indicates the means (l indicates by a local optimization method, t by the evaluation of the lower, mid and upper point) and the number following the dash indicates the iteration at which the incumbent solution was obtained. The infeasible problems were only evaluated for the initial pre-Deletion Filtering $IISU_0$ Algorithm phase. Values in bold type are the best for that given problem. A local optimization method application is made at the midpoint of the Initial Box for all subdivision methods and the stopping condition used was to stop when the Incumbent Solution and the Lower Bound agreed to within 5 digits of accuracy.

Problem : Subdivison	No. Iterations	No. function Evaluations	Max. Boxes	Time (sec)	No. of Splits	
					2	3
FP 3.2 : S1	10,105 (t-10,105)	407,607	1,809	115.42	10105	0
FP 3.2 : S1-M	9,111 (l-2,384)	923,636	434	293.33	9,111	0
FP 3.2 : S2, 30%	16,029 (l-12,501)	1,657,792	871	515.10	13,689	2,340
FP 3.2 : S2, 50%	12,011 (l-3,081)	393,618	709	206.66	10,529	1,482
FP 3.2 : S2, 70%	12,082 (l-8,890)	1,252,940	865	387.95	10,501	1,581
FP 3.2 : S3	15,714 (t-15,085)	725,727	2,573	205.78	15,714	0
FP 3.2 : S3-M	13,981 (l- 1,321)	1,439,306	804	454.49	13,981	0
FP 3.2 : S4	16,764 (t-16,073)	845,052	3,125	231.25	13,837	2,927
FP 3.2 : S4-M	14,989 (l-1,852)	1,591,967	1,108	493.07	12,389	2,600
FP 4.7 : S1	1,593 (t-1,501)	22,853	565	16.78	1,593	0
FP 4.7 : S1-M	1,530 (l-621)	53,550	353	27.33	1,530	0
FP 4.7 : S2, 30%	1,676 (t-914)	53,546	416	29.44	1,543	133
FP 4.7 : S2, 50%	1,565 (t-1,334)	51,102	392	27.55	1,395	170
FP 4.7 : S2, 70%	1,683 (l- 251)	54,923	383	29.75	1,494	189
FP 4.7 : S3	1,808 (t-1,680)	32,551	514	21.02	1,808	0
FP 4.7 : S3-M	1,795 (l-1,273)	69,342	388	12.52	1,795	0
FP 4.7 : S4	1,808 (t-1,680)	36,167	524	21.31	1,808	0
FP 4.7 : S4-M	1,795 (l-1,273)	72,932	388	34.73	1,795	0
Ex. 1 : S1	17 (l-0)	264	4	0.22	17	0
Ex. 1 : S1-M	17 (l-0)	528	4	0.30	17	0
Ex. 1 : S2, 30%	15 (l-0)	511	6	0.28	10	5
Ex. 1 : S2, 50%	8 (l-0)	335	1	0.16	0	8
Ex. 1 : S2, 70%	15 (l-0)	595	1	0.32	0	15
Ex. 1 : S3	16 (l-0)	313	5	0.22	16	0
Ex. 1 : S3-M	16 (l-0)	582	5	0.28	16	0
Ex. 1 : S4	14 (l-0)	352	4	0.21	12	2
Ex. 1 : S4-M	14 (l-0)	593	4	0.26	12	2

Table 8.6: Subdivision tests for normal GOS-BB operation

Problem : Subdivision	No. Iterations	No. function Evaluations	Max. Boxes	Time (sec)	No. of Splits	
					2	3
SHB : S1 †	10,171 (t-0)	22,713	10,000	140.33	10,171	0
SHB : S1-M	515 (l-224)	14,616	48	13.79	515	0
SHB : S2, 30%	422 (l-118)	11,091	54	12.52	275	147
SHB : S2, 50%	349 (l-126)	10,721	67	12.50	118	231
SHB : S2, 70%	495 (t-297)	14,671	63	17.01	104	391
SHB : S3 †	10,165 (t-0)	63,331	10,000	163.32	10,165	0
SHB : S3-M	508 (l-120)	18,366	47	15.09	508	0
SHB : S4 †	9,989 (t-0)	82,157	10,000	163.67	9,791	198
SHB : S4-M	439 (l-193)	14,620	54	13.09	349	90
UB 2 : S1	1	140	2	0.05	1	0
UB 2 : S1-M	1	140	2	0.06	1	0
UB 2 : S2, 30%	1	456	3	0.13	0	1
UB 2 : S2, 50%	1	456	3	0.14	0	1
UB 2 : S2, 70%	1	456	3	0.13	0	1
UB 2 : S3	1	144	2	0.05	1	0
UB 2 : S3-M	1	144	2	0.15	1	0
UB 2 : S4	1	150	3	0.08	0	1
UB 2 : S4-M	1	150	3	0.06	0	1
UB 2 c11 : S1	26	453	23	0.44	26	0
UB 2 c11 : S1-M	2	585	2	0.18	2	0
UB 2 c11 : S2, 30%	3	646	4	0.21	2	1
UB 2 c11 : S2, 50%	3	642	3	0.20	3	0
UB 2 c11 : S2, 70%	3	642	3	0.27	3	0
UB 2 c11 : S3	20	435	19	0.34	20	0
UB 2 c11 : S3-M	2	422	2	0.15	2	0
UB 2 c11 : S4	16	437	24	0.35	8	8
UB 2 c11 : S4-M	2	530	3	0.21	1	1

† - Terminated after the maximum number of boxes was exceeded.

Table 8.7: Subdivision tests for normal GOS-BB operation

Problem : Subdivision	No. Iterations	No. function Evaluations	Max. Boxes	Time (sec)	No. of Splits	
					2	3
UB 8 cs8 : S1	1,535	22,571	31	15.41	1,535	0
UB 8 cs8 : S1-M	1,535	148,083	31	48.65	1,535	0
UB 8 cs8 : S2, 30%	1,535	148,110	31	48.84	1,535	0
UB 8 cs8 : S2, 50%	1,535	148,110	31	48.39	1,535	0
UB 8 cs8 : S2, 70%	1,535	148,110	31	48.10	1,535	0
UB 8 cs8 : S3	244	15,466	6	6.59	244	0
UB 8 cs8 : S3-M	244	31,870	6	13.51	244	0
UB 8 cs8 : S4	223	5,996	6	6.34	212	11
UB 8 cs8 : S4-M	223	26,966	6	11.92	212	11
THC cs12 : S1	34	1,980	2	1.12	34	0
THC cs12 : S1-M	34	9,599	2	8.58	34	0
THC cs12 : S2, 30%	34	9,695	2	9.08	34	0
THC cs12 : S2, 50%	34	9,695	2	9.14	34	0
THC cs12 : S2, 70%	34	9,695	2	9.01	34	0
THC cs12 : S3	53	3,176	1	1.68	53	0
THC cs12 : S3-M	53	29,844	1	25.30	53	0
THC cs12 : S4	40	2,788	1	1.75	0	40
THC cs12 : S4-M	40	6,532	1	4.79	0	40

Table 8.8: Subdivision tests for normal GOS-BB operation

As we see, no single subdivision method is better than the rest. $S1 - M$ seems to be the best overall in terms of the maximum number of boxes generated (having 4 first place finishes and 4 second place finishes), $S1$ seems to be the best overall in terms of time, $S1 - M$ the best overall in terms of number of iterations required, and $S1$ the best overall in terms of the number of function calls. As for large problems, space is usually the primary reason for a termination, we shall use $S1 - M$ as the default subdivision method. We recognize that this method likely overuses the local optimization method and that a more efficient use of the local optimization method is desirable (see Section 10.3).

In problem, SUB, $S1$, $S2$, and $S3$ failed to terminate as too many boxes were created. As these methods are the only methods which do not employ the use of the local optimization method (except for the Initial Box) this indicates that the usage of local optimization method during the execution is desirable). For the INFEASIBLE problems, we see that the usage of the local optimization method is superfluous (as no feasible point exists) while for the DOMAIN UNBOUNDED problems, the usage of the local optimization method improves the performance significantly. Future research in this area is needed (see Section 10.3).

Deletion Filtering Operation

Consider a function $f : R^n \rightarrow R$ with a set of equality constraints, $h_{i_h}(x) = 0$, and a set of inequality constraints, $g_{i_g}(x) \leq 0$ with bounding functions $F(X)$, $H(X)$, and $G(X)$ respectively. The deletion filtering objective function for a box, B (where $m(B)$ is the midpoint of the Interval B) is:

$$\sum_{i_h} \max(|lb(H_{i_h}(B))|, |ub(H_{i_h}(B))|) + \sum_{i_g} \max(ub(G_{i_g}(B), 0))$$

and is the sum of the maximum amounts by which the constraint function bounds over a box, B , are infeasible. The boxes which are the least infeasible are searched first, thus the objective function's goal is to find a feasible point.

Evaluation

We use the following inconsistent constraint sets (see Section B.2.4)

Constraint Set	No. of		card($IISU_\Theta$) found
	variables	constraints	
cs4	5	4	2
cs6	2	8	4
cs9	2	6	2
cs13	2	3	3
cs14	2	4	2
cs15	6	5	3

to evaluate the subdivision methods, using the subdivision methods of $S1$, $S1 - M$, $S2$, 30%, $S2$, 50%, $S2$, 70%, $S3$, $S3 - M$, $S4$, and $S4 - M$. The results of the testing are summarized in Tables 8.9 and 8.10.

Problem : Subdivision	No. Iterations	No. function Evaluations	Max. Boxes	Time (sec)	No. of Splits	
					2	3
cs4 : S1	3,988	65,478	158	25.91	3,988	0
cs4 : S1-M	3,071	1,384,141	6	85.40	3,071	0
cs4 : S2, 30%	3,072	1,383,039	6	86.53	3,071	1
cs4 : S2, 50%	3,072	1,383,039	6	86.38	3,071	1
cs4 : S2, 70%	3,072	1,383,039	6	86.39	3,071	1
cs4 : S3	1,328	23,872	336	12.74	1,328	0
cs4 : S3-M	367	229,708	1	13.89	367	0
cs4 : S4	227	5,975	6	2.46	0	227
cs4 : S4-M	227	37,074	6	4.06	0	227
cs6 : S1	434,608	8,507,282	25	4,052.67	434,608	0
cs6 : S1-M	1,081	356,520	16	29.18	1,081	0
cs6 : S2, 30%	1,082	358,398	16	29.69	1,082	0
cs6 : S2, 50%	1,082	358,398	16	29.93	1,082	0
cs6 : S2, 70%	1,082	358,398	16	29.87	1,082	0
cs6 : S3 ‡	2,000,850	49,049,152	30	26,768.49	2,000,850	0
cs6 : S3-M ‡	2,000,850		30		2,000,850	0
cs6 : S4 ‡	2,000,714	60,023,106	29	30,731.94	0	2,000,714
cs6 : S4-M ‡	2,000,714	209,077,620	29	41,823.78	0	2,000,714
cs9 : S1	174	4,626	3	1.32	174	0
cs9 : S1-M	174	217,706	3	9.69	174	0
cs9 : S2, 30%	174	218,014	3	9.71	174	0
cs9 : S2, 50%	174	218,012	3	9.75	174	0
cs9 : S2, 70%	174	218,012	3	9.70	174	0
cs9 : S3	179	4,746	5	2.86	179	0
cs9 : S3-M	179	317,220	5	13.98	179	0
cs9 : S4	112	4,288	4	1.39	0	112
cs9 : S4-M	112	26,608	4	2.82	0	112

‡ - terminated after one of the trials took more than 2,000,000 iterations.

Table 8.9: Subdivision tests for Deletion Filtering operation

Problem : Subdivision	No. Iterations	No. function Evaluations	Max. Boxes	Time (sec)	No. of Splits	
					2	3
cs13 : S1	531	9,074	25	4.02	531	0
cs13 : S1-M	62	50,220	9	3.92	62	0
cs13 : S2, 30%	267	99,948	12	8.92	265	2
cs13 : S2, 50%	267	99,948	12	8.90	265	2
cs13 : S2, 70%	267	99,948	12	8.94	265	2
cs13 : S3	476	9,316	24	4.70	476	0
cs13 : S3-M	324	286,736	21	20.46	324	0
cs13 : S4	391	10,125	32	4.81	0	391
cs13 : S4-M	218	49,086	15	5.99	0	218
cs14 : S1	83	1,819	4	0.62	83	0
cs14 : S1-M	83	51,163	4	3.15	83	0
cs14 : S2, 30%	83	51,295	4	3.15	83	0
cs14 : S2, 50%	83	51,295	4	3.18	83	0
cs14 : S2, 70%	83	51,295	4	3.22	83	0
cs14 : S3	85	1,772	3	0.76	85	0
cs14 : S3-M	85	92,065	3	5.18	85	0
cs14 : S4	44	1,421	2	0.49	0	44
cs14 : S4-M	44	7,181	2	0.95	0	44
cs15 : S1	1,532	61,831	40	12.59	1,532	0
cs15 : S1-M	1,344	16,098,590	7	601.38	1,344	0
cs15 : S2, 30%	1,345	16,115,308	7	598.69	1,345	0
cs15 : S2, 50%	1,345	16,115,310	7	598.31	1,344	1
cs15 : S2, 70%	1,345	16,115,310	7	597.30	1,344	1
cs15 : S3	755	48,087	67	9.68	755	0
cs15 : S3-M	368	3,058,573	10	109.89	368	0
cs15 : S4	336	40,404	89	6.03	0	336
cs15 : S4-M	296	1,915,130	22	63.01	0	296

Table 8.10: Subdivision tests for Deletion Filtering operation

From the results we see that $S1 - M$ seems to be the best overall method; as the $S3$'s and $S4$'s fail to provide an ISU_{θ} for cs6, the $S2$'s are slightly outperformed by $S1 - M$ and as $S1$ is greatly outperformed by $S1 - M$ for cs6 and for the remaining constraint sets, $S1$ and $S1 - M$ are roughly equal in performance. As such we shall use it as the default subdivision method when deletion filtering.

Chapter 9

An Integrated Global Optimization Formulation and Solution Methodology

In this Chapter we examine a prototype for a GOS-BB, called GOPS (a global optimization problem solver) and then use it to evaluate GOS-BB against the problem test set developed in Section 5.

9.1 GOPS: A Software Prototype for a GOS-BB

In this section we give a brief overview of an implementation of a GOS-BB, called GOPS (a global optimization problem solver), coded using GNU-C++ on a Sparc 10-30c workstation.

9.1.1 Set of Boxes

Let mbs be a parameter representing the minimum box size to be examined. Its default value is set at a width of $1.0 * 10^{-8}$. Three sets of boxes are maintained which comprise the Set of Boxes. They are the set of boxes whose widths are $\geq mbs$, $SB_{\geq}$, those whose widths are $< mbs$, $SB_{<}$, and those whose widths are smaller than the minimum box size representable (see Section 7.1.2), SB_a . Once a box is placed in $SB_{<}$ or SB_a it can no longer be subdivided, but it can be fathomed. The sets are ordered by the lower bound of their functional value over the box and at each iteration the box with the least functional value in

$SB_{\geq}$ is chosen to be subdivided. Each set of boxes is maintained as an AVL tree (a balanced binary search tree) [1] with each node holding the set of boxes with the same functional lower bound (maintained in a stack associated with the node).

9.1.2 Bounding and Branching Methods

The objective and constraint functions are coded explicitly along with four bounding functions, $NIE - H$, $NIE - P$, $MK - H$, and $MK - P$ (see Section 8.1.1) for the objective function and two bounding functions, $NIE - H$ and $NIE - P$ for each constraint function. The bounding function strategy used for a constraint function was to determine both $NIE - H$ and $NIE - P$ bounds and use the intersection of these bounds to determine the bound on the constraint function. Future versions envision an automated means by which a set of bounding functions can be used for any function as well as the availability of the adaptive bounding methodology for constraint functions. The default bounding method is the adaptive bounding method for the objective function and the best of $NIE - H$ and $NIE - P$ for the constraint functions.

The box with the lowest functional bound in the set $SB_{\geq}$ is chosen for subdivision. The subdivision methods, $S1$, $S1 - M$, $S2$, $S3$, $S3 - M$, $S4$, and $S4 - M$ (see Section 8.2) were implemented. The default subdivision method is $S1 - M$.

9.1.3 Stopping Conditions

The four stopping conditions:

1. Stop if the difference between the Lower Bound and the Incumbent Solution is less than an *a priori* determined value.
2. Stop if all boxes are less than an *a priori* determined size or if the difference between the Lower Bound and the Incumbent Solution is less than an *a priori* determined value.
3. Stop if all boxes are less than an *a priori* determined size and if the difference between the Lower Bound and the Incumbent Solution is less than an *a priori* determined value.
4. Stop if a feasible point is found.

were implemented.

The default value for the difference between the Lower Bound and the Incumbent Solution is set at at least 5 digits of accuracy.

9.1.4 Search for Incumbent Solution

The search for an incumbent solution was implemented as follows. For each new box generated, the lower, mid, and upper points were checked as candidate incumbent solutions. A mechanism was employed to ensure that each point was only checked once. In addition, an optional number of random points can be generated as candidate incumbent solutions for each box that enters the Set of Boxes.

A local optimization package, *LSGRG2* [112], was used as a local optimization procedure. It is used to search for a local minimum (using the midpoint of the box) for the subdivision method *S2* and can be used optionally after the creation of the initial box as well as with each random point generated as described above. The default setting is to use *LSGRG2* after the creation of the initial box.

9.1.5 Algorithms

The deletion filtering *IISU₀* algorithm was not automated. Each subproblem is run separately. Future versions envision the automation of this algorithm. The unbounded point detection algorithm was automated for rational functions only.

9.1.6 Statistics Kept

Various statistics were kept for the system and procedures, including the total and maximum times spent in a procedure, the number of times they were called, etc. as well as the number of function calls, etc. A mechanism to examine the state of the parameters as well as the current statistics was implemented, to monitor how well the algorithm is progressing. In addition, the iteration at which the Incumbent Solution was identified is maintained as well as the means by which it was found (either by *LSGRG2* or by the examination of the lower, mid, and upper point).

Snap Shot Evaluations

By default, after a time span of roughly 10 minutes, a snap shot of the state of the solution is made. In addition, a snap shot is made every 10,000 iterations. The 10 minute snap shot evaluation is made in order to examine how much information has been gathered after roughly 10 minutes of operation. The snap shots after each 10,000 iterations are used to determine patterns in the progress of the solution process for problems that are not quickly solved.

9.1.7 Defaults

Unless otherwise stated, default values of $1.0 * 10^{-6}$ for the *constraint- ϵ* and for $\delta_{unbounded}$ are used. By default, 13 digits of accuracy are available for any calculation and the smallest and largest numbers representable have been arbitrarily set at $lbN_{\Theta} = -1.0 * 10^{100}$ and $ubN_{\Theta} = 1.0 * 10^{100}$. In addition, a default Initial Box defined by $-1.0 * 10^{10} \leq x_i \leq 1.0 * 10^{10}$ is used to define a region of width $2 * 10^{10}$ which will extend somewhat beyond the effective search region (see Section 7.1.2).

9.2 Evaluation

In this section we examine three classes of problems aimed at testing the three major outcome classes of INFEASIBLE, DOMAIN UNBOUNDED, and GLOBAL MINIMUM. Each class of problems is designed (see Section 5) to evaluate GOS-BB to see if it can determine the proper outcome, results, etc. Then the flatness factor is examined to see if produces correct results and three case studies are presented to help evaluate the performance of GOS-BB from a modelling perspective. This is followed by an examination of how INDETERMINATE outcomes have been dealt with. Finally, an overall evaluation of GOS-BB is made.

9.2.1 Infeasible Problems

In this section we examine how well the deletion filtering *ISSU $_{\Theta}$* detection algorithm performs against the problems summarized in Table 9.1, where UB denotes an Unbounded problem (see Section 5), FP denotes a problem from the Floudas and Pardalos test set, and SHB and THC are the Six Hump Back and Three Hump Camel problems.

Problem	No. of			card(<i>ISUe</i>)	No. of Iterations	No. of function Evaluations	Max. No. Boxes	Time (sec)
	variables	constraints	variable bounds					
UB 2 cs2 mod	2	3	1	3	0	64	1	0.13
UB 8 cs8	5	2	0	2	1,536	148,248	31	48.75
FP 3.2 cs5	5	5	0	3	1,457	2,618,416	38	89.21
SHB cs10 †	2	6	0	3	2,272,105	376,422,635	28	38,419.68
THC cs12	2	18	0	2	119	73,849	5	12.92

† - terminated after one of the trials took more than 2,000,000 iterations.

Table 9.1: INFEASIBLE problem tests

Snap Shot Evaluation of SHB cs10

For SHB cs10 the final iteration of the deletion filtering $IISU_{\Theta}$ algorithm terminated as too many iterations had occurred. A snap shot after 33,712 iterations (600.01 seconds) revealed that no feasible point had been found, that a maximum of 24 boxes had been used, that the minimum box size in the list was of width 39.5 and roughly 25% of the volume of the Initial Box had been eliminated as being infeasible. Snap shots taken at regular intervals up to the termination after 2,000,000 iterations indicate that this state has not changed (at 2,000,000 iterations, 34,202.36 seconds, a maximum of 28 boxes, a minimum box size of 316, and still only about 25% of the volume of the Initial Box had been eliminated). This suggests that the objective function used is inefficient at directing the search for a feasible point or for identifying infeasible regions.

As a Phase I objective function was used and as only 2 out of 6 constraints are present for the final iteration, the existence of a feasible point is suggested. Thus an alternative method, which carries out an initial breadth first search until all boxes have a width of $< 1.0 * 10^4$ and then continues with the normal depth first search, was used to determine the outcome for the final iteration (which was that a feasible point existed).

Thus, we see that the snap shot evaluation has provided useful information, in this case to evaluate the performance of the default Phase I objective function. This shows that it may be beneficial to develop a control/expert system to control the operation of the solution process by altering objective functions, etc. (see Section 10.3). In addition, this shows that the phase I objective function used by the deletion filtering $IISU_{\Theta}$ algorithm is not optimal. Thus, more research is needed to determine the best objective function(s) to use with the deletion filtering $IISU_{\Theta}$ algorithm (see Section 10.3).

Conclusion

From the above results (as well as from the results of Section 8.2.5), we see that the deletion filtering $IISU_{\Theta}$ algorithm can effectively identify an $IISU_{\Theta}$ under various conditions. Problem SHB cs10 had one sub-problem which required about 270,000 iterations to determine an infeasible outcome. The need to eliminate all of the boxes to determine infeasibility requires that the whole space be searched, thus limiting the size of the problems that can be handled. This limit on size is more so for the number of dimensions and not on the number of constraints involved. More research is needed to extend the method to larger problems

or to develop new methods which can be applied to larger problems (see Section 10.3). In addition, the use of a local optimization procedure to look for a feasible point within the deletion filtering $IISU_{\Theta}$ algorithm is recommended. The frequency of use of such a procedure remains to be determined.

Thus, we conclude that the deletion filtering $IISU_{\Theta}$ algorithm can identify $IISU$'s (subject to the degree of accuracy specified by the *constraint- ϵ*).

9.2.2 Domain Unbounded Problems

In this section we examine how well the unbounded point detection algorithm performs against a set of problems, some of which have DOMAIN UNBOUNDED outcomes and some of which have GLOBAL MINIMUM outcomes but have 'false' unbounded domain points (points for which both the denominator and the numerator have a common root).

Testing was carried out against the problems summarized in Table 9.2, where UB denotes an Unbounded problem (see Section 5). In all cases subdivision was by $S1 - M$ (see Section 8.2). A maximum of 10,000 boxes was allowed. Max. Boxes refers to the maximum number of boxes in the Set of Boxes at any time.

From the results we see that in all problems with zero constraints, an accurate outcome is determined. For problem UB 6 the algorithm terminated when more than 10,000 boxes were present. GOS-BB had problems with Unbounded 3, where a false unbounded domain point exists (the denominator and numerator have a common root of the same order), but did not report an inaccurate outcome. Unbounded 4 is Unbounded 3, with the common root removed and this problem is effectively solved. This shows the importance of reducing any functions (see Section 10.3). We note that for Unbounded 7, which contains a common root of $x_1 - 3$ of order 1 in the numerator and of order 3 in the denominator that a correct unbounded domain point was identified, thus indicating that some non-reduced problems (likely those which do contain unbounded domain points) do not require the reduction of common roots.

For constrained problems, with the exception of Unbounded 2 mod, the correct outcome was identified, with the correct domain unbounded point or global minimum supplied as appropriate. Problems Unbounded 2 mod and Unbounded 2 mod2 both have GLOBAL MINIMUM outcomes, with an infeasible unbounded from below point, x_{ub} , near the feasible region. In Unbounded 2 mod x_{ub} is within 0.0001 units of the feasible region while in Un-

Problem Name	variables	No. of constraints	variable bounds	solution outcome	No. of Iterations	No. of function Evaluations	Max. No. Boxes	Time (sec)
UB 1	1	0	0	DOMAIN UNBOUNDED	4	849	2	0.24
UB 2	2	0	0	DOMAIN UNBOUNDED	1	140	2	0.13
UB 3	1	0	0	INDETERMINE †	146	17,946	67	6.41
UB 4	1	0	0	GLOBAL MINIMUM	44	854	1	0.45
UB 5	2	0	0	DOMAIN UNBOUNDED	58	2,313	25	1.28
UB 6	2	0	0	INDETERMINE	39,888	428,408	10,000	371.56
UB 6 /B*	2	0	0	DOMAIN UNBOUNDED	46	2,110	15	1.20
UB 7	2	0	0	DOMAIN UNBOUNDED	1	210	2	0.20
UB 8	5	0	0	DOMAIN UNBOUNDED	1	1,494	2	0.41
UB 2 cs2	2	3	0	DOMAIN UNBOUNDED	14	7,405	2	0.57
UB 2 mod	2	0	1	DOMAIN UNBOUNDED †	19	559	4	0.41
UB 2 mod2	2	0	1	GLOBAL MINIMUM	1,639	35,257	519	15.58
UB 8 con21	5	1	0	DOMAIN UNBOUNDED	72	81,156	51	15.32

† - the correct outcome is GLOBAL MINIMUM.

where /B* indicates an Initial Box with $-1,000.0 \leq x_i \leq 1,000.0$ was used.

Table 9.2: DOMAIN UNBOUNDED problem tests

bounded 2 mod2 x_{ub} is within 0.001 units of the feasible region. In Unbounded 2 mod the *constraint- ϵ* allowed x_{ub} to be a feasible point, while in Unbounded 2 mod2 it did not and thus for Unbounded 2 mod a false outcome of DOMAIN UNBOUNDED was returned while for Unbounded 2 mod2 the correct outcome was returned. In Section 9.2.1, we see that for problem Unbounded 2 cs2 mod, which has an infeasible unbounded from below point, the correct outcome is obtained.

Conclusion

The unbounded point detection algorithm can be used to detect unbounded domain points under both unconstrained and constrained conditions. Single unbounded domain points can be identified as well as at least one of an infinite number of unbounded domain points. To date only rational functions have been tested. More research is needed to extend the algorithm to more general functions (see Section 10.3). Thus, we conclude that GOS-BB can identify unbounded domain points, subject to the degree of accuracy specified by $\delta_{unbounded}$ and by the *constraint- ϵ* .

9.2.3 Global Minimum Problems

In this section we examine how well GOPS performs against problems with GLOBAL MINIMUM outcomes. Testing was carried out against the problems summarized in Table 9.3, where FP denotes a problem from the Floudas and Pardalos test set, SHB and THC are the Six Hump Back and Three Hump Camel problems, VR is the Variation on Rosenbrock problem and SCH denotes a problem from the Schwefel test set. In parenthesis after the number of iterations is a value that indicates when and by what means the incumbent solution was obtained. The letter before the dash indicates the means (l indicates by a local optimization method, t by the evaluation of the lower, mid and upper point) and the number following the dash indicates the iteration at which the incumbent solution was obtained.

All of the test problems resulted in a correct outcome of GLOBAL MINIMUM. For FP 6.2, though the best known solution is found, the problem is that the Lower Bound is not yet accurate, while for FP 3.1, neither the best known solution or a good Lower Bound has been identified, although both problems have been shown to have a GLOBAL MINIMUM outcome. For FP 3.1 the Incumbent Solution found was within 3 digits of accuracy of the best known solution. In all other problems, the proper global minimum and Lower Bound

Problem Name	No. of			No. of Iterations	No. of function Evaluations	Max. Boxes	Time (sec)
variables	constraints	variable bounds					
FP 3.1 †	9	6	16	561,833 (1-4,268)	1,156,875,190	30,000	30,926.86
FP 3.2	5	6	10	9,111 (1-2,384)	923,636	434	293.33
FP 4.6	2	2	4	58,611 (1-41,760)	6,867,059	267	903.23
FP 4.7	2	1	4	1,530 (1-621)	53,550	353	27.33
FP 6.2 †*	9	6	2	77,001 (1-1)	278,844,709	30,000	6,750.70
SHB	2	0	0	461 (1-260)	12,965	52	12.38
THC	2	0	0	183 (t-0)	5,894	36	5.03
SCN 2.1	2	0	0	1 (1-0)	971	2	0.24
SCN 3.1	3	0	0	1 (1-0)	83	1	0.17
SHB cs1	2	3	0	265 (1-111)	57,248	29	9.53
SHB cs11	2	3	0	254 (1-214)	146,886	19	13.80
THC cs3	2	2	2	223 (t-223)	24,339	13	5.45
THC cst	2	3	2	21,490 (1-18,196)	6,956,212	29	724.16
Zhang 1	2	2	4	1,233 (1-438)	137,609	234	32.13
VR †**	50	0	100	4,233 (1-3)	6,609,015	2,000	3,561.2
VR cs16 †	50	25	100	11,383 (1-5)	6,986,797,078	2,000	172,434.24
VR cs17 †**	50	50	100	11,383 (1-5)	14,594,752,839	2,000	255,524.58

† - did not identify the global minimum after the preset maximum number of boxes was exceeded.

†* - Terminated after the preset maximum number of boxes was achieved and due to a problem with LSGRG2, required a smaller initial box than specified in Section 5.

†** - terminated after the preset maximum number of boxes was exceeded and global minimum is unknown.

Table 9.3: GLOBAL MINIMUM problem tests

accurate to at least 5 digits is found. FP 6.2 contains equality constraints while FP 3.1 does not, suggesting that the problem may not be with the equality constraints, but with the size of the problem and/or the 'flatness' of the objective function (as both FP 3.1 and FP 6.2 have linear objective functions of relatively small slopes).

From the results of VR, VR cs16, and VR cs17 we see that all three of these produce 'limited' results in that the difference between the Incumbent Solution and the Lower Bound is large (of the order of 760 with an Incumbent Solution functional value of about -260). Examining the Set of Boxes after termination gives a smallest box of size of width 10. As the Initial Box was of width 20 (along all edges), this indicates that the size of problem is likely to be the limiting factor. We note that the addition of constraints has the effect of increasing the number of iterations until the maximum number of boxes is exceeded, due to the elimination of some infeasible boxes, however, for the constraint sets chosen, this effect is not large enough to provide greater accuracy for the Incumbent Solution. Thus, a limiting factor on the GOS-BB is the number of dimensions.

Comparison to existing methods is difficult due to a scarcity of methods that provide guaranteed results. For problem FP 3.2, Zhang and Sha [121] report a result of 34,203 iterations and a maximum cardinality for the set of boxes of 4,851 for obtaining a result accurate to 6 digits for the incumbent solution and with no boxes with width greater than 10^{-4} . We obtain similar results using $S1-M$ in 18,626 iterations and a maximum cardinality of 473 boxes. For Zhang No. 1, they report a result of 1719 iterations and a maximum cardinality for the set of boxes as 125 for a result accurate to 4 digits and all boxes with width less than 10^{-5} , whereas we obtain the same results in 1441 iterations with a maximum cardinality of 224 boxes. Thus we see a general improvement over their method.

However, interval Newton based methods generally perform better than our method as the following example shows, due to the additional power available from using knowledge gained from the derivative. We have avoided using such knowledge as we do not wish to require assumptions about the differentiability of the functions used. In Hansen [46], on a Cyber 175, the Three Hump Camel problem took 0.38 seconds and 407 function evaluations to get an accurate result to well under 16 digits of accuracy and the Beale (Schwefel 2.1) problem took 2.1 seconds and 2,040 function evaluations to get a result accurate to 26 digits and a minimum box size of width less than $1.0 * 10^{-11}$ while using $S2, 50\%$ for the Three Hump Camel problem took 12.83 seconds and 11,266 function evaluations to get a result

accurate to 16 digits and for the Beale problem it took 20.53 seconds and 20,758 function calls to get a result accurate to 24 digits and a minimum box size of 1.0×10^{-11} .

Snap Shot Evaluations

A snap shot evaluation after 10 minutes is made for problems FP 3.1, FP 6.2, FP 1.6, VR, VR cs16 and VR cs17 in order to examine the amount of information provided by the GOS-BB. As all of these problems took more than 10 minutes, we will examine if the results after 10 minutes of execution can be used to predict the final results.

FP 3.1

After, 2,005 iterations, 600.25 seconds, an Incumbent Solution of 7,321 (at iteration 1,083) and a Lower Bound of 2,100 were found, 3,841 applications of a Local Optimizer (improving the Incumbent Solution only once and finding only 1 feasible point) and a maximum of 1,836 boxes were used, and only 0.74% of the volume of the Initial Box has been eliminated. Currently there are 1,836 boxes (all of unknown feasibility) with a width of 500 and 9,000 for the smallest and largest box respectively (the Initial Box has a width of 9,900). 0 boxes have been fathomed and 170 boxes have been removed due to infeasibility. The 10 best boxes (by lower bound) do not contain the best known solution.

As the objective function is linear, we know that the bounds produced by the interval analysis are accurate. Thus the Lower Bound is an accurate bound for the current Set of Boxes. As only about 0.5% of the boxes have been determined as being infeasible (occupying 0.74% of the volume of the Initial Box), we see that no large infeasible regions have been detected. This can be due to either the lack of any such regions, or due to the size of the regions producing poor bounds for the constraint functions. As no boxes have been fathomed by the Incumbent Solution, this indicates that the Incumbent Solution is likely to be in a high region of the objective function (this is also supported by the fact that the best boxes do not contain the Incumbent Solution). The last two points suggest that there are either small regions of feasibility and/or near-feasibility throughout the region defined by the Initial Box. The fact that 3,841 applications of the local optimizer (centered about the lower regions of the objective function as these are the boxes that are chosen for further subdivision) have not found a better Incumbent Solution, indicates that latter is the more likely occurrence. This suggests that the Incumbent Solution is likely to be a fairly good solution and a large portion of the region defined by the Initial Box is likely to be infeasible.

At termination, an Incumbent Solution of 7,054.6 and a Lower Bound of 3,787.5 were obtained, 591,833 applications of the local optimizer (improving the Incumbent Solution 3 times and finding 9 feasible points) and a maximum of 30,000 boxes were used, 99.4% of the volume of the Initial Box has been eliminated, and the width of the smallest and largest box were 61.8 and 495 respectively. 953 boxes have been fathomed and 530,881 boxes have been removed due to infeasibility. Currently, 30,000 boxes (all of unknown feasibility) exist, contained within a box of width 900. About 90% of the boxes created have been removed due to infeasibility, while less than 0.2% have been fathomed by the Incumbent Solution.

Thus we see a small change in the Incumbent Solution and a larger change in the Lower Bound. The Incumbent Solution is within 3 digits of accuracy with the best known solution, but is at a different point (about a distance of 130 between the two points). From the failure of the large number of applications of the local optimizer to improve greatly upon the Incumbent Solution from the 10 minute snap shot (only about a 4% improvement is made), we can conclude that the value for the Incumbent Solution is a good solution. In addition, as a large number of infeasible boxes have been identified, the analysis carried out from the information gained at the 10 minute snap shot is likely to be correct.

FP 6.2

After 4,552 iterations, 600.24 seconds, an Incumbent Solution of value 400 (at iteration 1 by LSGRG2) and an Upper Bound (as it is a maximization problem) of 2,650 were found, 49.1% of the volume of the Initial Box has been eliminated, and 6,496 applications of the local optimizer were made (improving the Incumbent Solution once and finding 9 feasible points). A maximum of 1,944 boxes were required. Currently 1,944 boxes (all of unknown feasibility) exist, the width of the smallest and largest box are 15.6 and 500 respectively (the width of the Initial Box is 500). The Set of Boxes are contained within a box equal to the Initial Box. The 10 best (by lower bound) boxes did not contain the Incumbent Solution. 1 box was fathomed by the Incumbent Solution and 2,608 boxes were removed due to infeasibility.

As the objective function is linear and except for the fact that about 50% of the volume of the Initial Box has been eliminated, the results are similar to that of FP 3.1 analyzed above. The effect of the larger removal is balanced by the fact that the smallest containing box is still the Initial Box. Thus, a similar analysis is valid, except that it is possible that the regions of small near-feasibility are only near the lower and upper point of the containing

box. Thus, the Incumbent Solution is likely to be a fairly good solution and a large portion of the region defined by the Initial Box is likely to be infeasible.

At termination 97.8% of the volume of the Initial Box has been eliminated, the Incumbent Solution has not changed, an Upper Bound of 2,193.75 was found, and the width of the smallest and largest box are 3.91 and 250 respectively. The width of the box which contains the Set of Boxes is 100. There are currently 30,000 boxes (all of unknown feasibility). 4 boxes have been fathomed and 46,998 boxes have been eliminated due to infeasibility. The local optimizer was applied 107,001 times (improving the Incumbent Solution once (at iteration 1) and finding 23 feasible points).

Thus we see no change in the Incumbent Solution and some change in the Upper Bound. The Incumbent Solution is equal to the best known solution. From the failure of the large number of applications of the local optimizer to improve upon the Incumbent Solution, we can conclude that the value for the Incumbent Solution is a good solution. In addition, as a large number of infeasible boxes have been identified, the analysis carried out from the information gained at the 10 minute snap shot is likely to be correct.

FP 4.6

After 42,518 iterations, 600 seconds, a global minimum has been identified (at iteration 41,760 by LSGRG2) to an accuracy of 3 digits. A maximum of 243 boxes occurred of which there are currently 218 boxes, contained within a box of width about 0.1, and 99.998% of the volume of the Initial Box has been eliminated. As the results are close to a global minimum and as the number of boxes is small and are contained within a small region, it is likely that only one global minimum exists, or that if more than one global minima exists, they are clumped closely together.

At termination (903.23 seconds), 99.9999% of the volume of the Initial Box has been eliminated and the global minimum was identified to an accuracy of 5 digits. There remain only 249 boxes (all of unknown feasibility), with the width of the smallest and largest being 1.4×10^{-6} and 9.8×10^{-4} respectively, and they are contained within a box of width 0.03. Thus, the global minimum (or minima) are contained within a small box. Thus the analysis made after the 10 minute snap shot is reasonably accurate.

VR

After 244 iterations, 601.58 seconds, a maximum of 144 boxes occurred, an Incumbent Solution of -238.94 and a Lower Bound of -1000 have been found and only $1.0 \times 10^{-14}\%$ of

the volume of the Initial Box has been eliminated. The width of the smallest and largest box are 10 and 20 respectively (the Initial Box has a width of 20). The local optimizer was called 388 times (improving the Incumbent Solution 3 times and finding 364 feasible points). 101 boxes had been fathomed by the Incumbent Solution (about 30% of the boxes produced). The flatness factors for boxes of width 2.0×10^{-6} , 2.0×10^{-3} , and 2.0 are 3.8, 17.2, and 45.300 respectively.

The values of the Incumbent Solution and that of the Lower Bound are of the same magnitude. The Incumbent Solution is not on the boundary of the Initial Box and as the flatness factors go from small to large, this indicates that it lies on the bottom of some valley. As the objective function is quite complex, it is likely that the bounds produced for the boxes are inaccurate. Thus it is likely that the Lower Bound is actually higher than stated. This suggests that the value attained for the Incumbent Solution is likely to be fairly good.

At termination, 4,233 iterations, a maximum of 2,000 boxes occurred, the Lower Bound was still at -1000, the Incumbent Solution has remained the same, the width of the smallest and largest box are 10 and 20 respectively, and $1.0 \times 10^{-13}\%$ of the volume of the Initial Box has been eliminated. Thus we see that GOS-BB is slowly eliminating regions of the Initial Box, but as it does so it requires that a large number of boxes need be generated. The local optimizer was called 6,233 times (improving the Incumbent Solution 3 times and finding 4,756 feasible points). As the Incumbent Solution has not been improved upon, this adds some support to the conclusion made at the 10 minute snap shot is correct.

VR cs16

After 16 iterations, 613.09 seconds, a maximum of 17 boxes occurred (all of unknown feasibility). An Incumbent Solution of -238.85 and a Lower Bound of -1000 were found, and no boxes have been eliminated. All boxes are of width 20, equal to that of the Initial Box. The local optimizer was called 33 times (improving the Incumbent Solution 3 times and finding 10 feasible points). The flatness factors for boxes of width 2.0×10^{-6} , 2.0×10^{-3} , and 2.0 are 4.8, 17.8, and 46,200 respectively, and the two smaller boxes are feasible.

From the values of the flatness factor we see that in a small neighbourhood about the Incumbent Solution the objective function is fairly flat. As the area about the Incumbent Solution is feasible and the Incumbent Solution is not on the boundary of the Initial Box it is likely that the Incumbent Solution lies at the bottom of a valley. As the objective function is quite complex, it is likely that the bounds produced for the boxes are inaccurate. Thus

it is likely that the Lower Bound is actually higher than stated. These two points suggest that the value attained for the Incumbent Solution is likely to be fairly good. As only about one third of the applications of the local optimizer result in a feasible point and as the local optimizer uses a phase I objective function to try to find a feasible point, this suggests that for a majority of the regions of infeasibility are reasonably complex.

At termination, 11,383 iterations, a maximum of 2,000 boxes have occurred, the Lower Bound was still at -1000 and about $1.0 \times 10^{-13}\%$ of the volume of the Initial Box has been eliminated and the Incumbent Solution and the Lower Bound had not changed. The local optimizer was called 13,383 times (improving the Incumbent Solution 3 times and finding 35 feasible points). 1,564 boxes were fathomed and 7,820 boxes were eliminated due to infeasibility. There exist 2,000 boxes contained in a box equal to the Initial Box. The width of the smallest and largest box are 10 and 20 respectively.

As the Incumbent Solution has not been improved upon and as the very few feasible points have been found by the Local Optimizer applications, the analysis made at the 10 minute snap shot is likely to be correct.

VR cs17

After 13 iterations, 630.47 seconds, a maximum of 14 boxes were used. An Incumbent Solution of -238.87 and a Lower Bound of -1000 were found, and no boxes have been eliminated. All boxes are of width 20, equal to that of the Initial Box. The local optimizer was called 27 times (improving the Incumbent Solution 3 times and finding 8 feasible points). The flatness factors for boxes of width 2.0×10^{-6} , 2.0×10^{-3} , and 2.0 are 5.0, 17.0, and 46,300 respectively, and all of these boxes are of unknown feasibility.

From the values of the flatness factor we see that in a small neighbourhood about the Incumbent Solution the objective function is fairly flat. As the area about the Incumbent Solution is of unknown feasibility and the Incumbent Solution is not on the boundary of the Initial Box it is likely that the Incumbent Solution lies near the bottom of a valley. As the objective function is quite complex, it is likely that the bounds produced for the boxes are inaccurate. Thus it is likely that the Lower Bound is actually higher than stated. These two points suggest that the value attained for the Incumbent Solution is likely to be fairly good. As only about one third of the applications of the Local Optimizer result in a feasible point and as the local optimizer uses a phase I objective function to try to find a feasible point, this suggests that for a majority of the regions of infeasibility are reasonably complex.

At termination, 11,383 iterations, the Lower Bound was still at -1000 and about $1.0 \times 10^{-13}\%$ of the volume of the Initial Box has been eliminated and the Incumbent Solution and the Lower Bound had not changed. The local optimizer was called 13,383 times (improving the Incumbent Solution 3 times and finding 29 feasible points). 1,564 boxes were fathomed and 7,820 boxes were eliminated due to infeasibility. There exist 2,000 boxes contained in a box equal to the Initial Box. The width of the smallest and largest box are 10 and 20 respectively.

As the Incumbent Solution has not been improved upon and as the very few feasible points have been found by the local optimizer applications, the analysis made at the 10 minute snap shot is likely to be correct.

Conclusions

From the snap shot evaluations for the larger problems, we see that a good solution is usually found within the first 10 minutes. The difficulty with these problems is in eliminating and reducing the Set of Boxes to obtain an accurate Lower Bound. That is, a good Incumbent Solution is usually found within the first 10 minutes for problems that have GLOBAL MINIMUM outcomes, but for the larger problems, a large amount of time and space is required to prove that the Incumbent Solution is in fact a global minimum.

For the VR problems, the size of the problem does limit the amount of information that is useful after 10 minutes of processing, due to the fewer number of iterations of the algorithm that have occurred. For the FP problems, formulation information is more plentiful and a reasonably good picture of the problem is developed. For all of the problems the analysis at 10 minutes is indicative of what the analysis at a latter time produces.

This suggests that an examination of the solution process after an initial 'short' period of time is appropriate. The value of short is dependent on the size of the problem, with the larger the problem the larger the value of 'short'.

It also suggests that any control system that is developed to control the solution process should undertake an initial detailed evaluation after each significant change (such as change in subdivision strategy, objective functions, etc.) that it has imposed upon the solution process (see Section 10.3).

Conclusions

The GOS-BB can identify GLOBAL MINIMUM outcomes and provide an Incumbent Solution and Lower Bound to within specified accuracies at about the same performance level of existing comparable methods. Like these methods it can, in general, only do this only for small problems. Furthermore, GOS-BB usually determines a good Incumbent Solution in a relatively short period of time and that the major portion of the time (and space) required is spent on providing a proof (by way of an accurate Lower Bound) of the accuracy of the Incumbent Solution.

9.2.4 Flatness Factors

In Table 9.4, flatness factors about the Incumbent Solution are given for a subset of the test problems in Table 9.3. For all of the test problems, the flatness factors are consistent with the known shape of the objective function. FP 4.6 has a linear objective function and we see that the flatness factor remains constant. FP 3.2, FP 4.7 and SCH 3.1 are quadratic functions and this is reflected in the flatness factors for FP 4.7 and SCH 3.1, whereas for FP 3.2 a more linear region about the Incumbent Solution is indicated. SHB cs1, SHB cs11 and THC cs3 have objective functions which have multiple hills. The flatness factors for SHB cs11 and THC cs3 indicate that the Incumbent Solution is in a large valley, whereas for SHB cs1, the flatness factors indicate that the Incumbent Solution is in a small valley.

Problem Name	width of box		
	$2.0 * 10^{-6}$	$2.0 * 10^{-3}$	2.0
FP 3.2	454.6	454.6	454.6
FP 4.6	2.0	2.0	2.0
FP 4.7	16.06	16.06	17.06
SCH 3.1	$1.1 * 10^{-5}$	$1.1 * 10^{-2}$	19.50
SHB cs1	8.59	0.017	22.15
SHB cs11	12.40	12.44	101.6
THC cs3	1149.0	1149.0	1876.5

Table 9.4: Flatness Factors

9.2.5 Case Studies

In this section we examine three problems in greater detail in order to examine how well GOS-BB performs from a modelling perspective.

Spring Design Problem

This problem (see Appendix B.1.3) deals with the optimal design of a high duty helical compression spring.

An application of GOS-BB determines an Incumbent Solution of:

$$\begin{aligned} f(x^*) &= 0.025715 \\ x^* &= (4, .50000, 0.13572, 0.43214) \end{aligned}$$

and a Lower Bound of 0.025710 (an accuracy of 4 digits) and terminates after 20,000 boxes are in the Set of Boxes.

In [60] the best result is given as

$$\begin{aligned} f(x^*) &= 0.10563 \\ x^* &= (5, .76122, 0.19939, 0.50000) \end{aligned}$$

so an improvement upon the previous results have been obtained. The flatness factors are determined as:

width of box		
$2.0 * 10^{-6}$	$2.0 * 10^{-3}$	2.0
0.437	0.437	4.502

and the smallest containing box of the Set of Boxes is:

variable	Interval
x_1	[4.00000, 4.00003]
x_2	[0.50000, 0.50023]
x_3	[0.13570, 0.13573]
x_4	[0.30000, 0.43219]

which has a volume of $2.736 * 10^{-14}$. The Set of Boxes consists of 20,000 boxes, and a net volume of $7.566 * 10^{-16}$. The Initial Box has a volume of 0.5675. The volume of the Set of

Boxes is about 2.7% of the volume of the smallest containing box, and about $4.8 \times 10^{-12}\%$ of the volume of the Initial Box.

From the smallest containing box, we see that the boxes of the set of boxes differ only in their value of x_4 . This is due to the fact that once x_1, x_2 , and x_3 have been determined, for constraint c_4 any value of $x_4 \leq 0.4321$ will satisfy the constraint. Thus a large number of boxes are generated, varying essentially only in their value of x_4 . The flatness factors indicate that the function is relatively flat near the Incumbent Solution. Thus, one would conclude that the values obtained for the design variables are appropriate, due to the flatness of the Incumbent Solution and due to the fact that except for x_4 all of the other design variables have been determined to a reasonable degree of accuracy.

Thus the information on the flatness and on the 'shape' of the Set of Boxes is useful in determining the appropriateness of the Incumbent Solution.

Floudas and Pardalos 11.1 - Modified

This problem (see Appendix B.1.1) deals with the design of a hollow cylindrical pressure shell of length 50 feet and 5 feet diameter. The design variables are $x_3, x_{15}, x_{20}, x_{37}$, and x_{38} . The minimum values for the design variables are 0.0625. The design variables correspond to the thickness of the head, shell, saddle ring, vacuum ring and wear plate. The original problem had fewer variables, some additional variables were created by taking some of the parameters and allowing these parameters to vary by a small amount.

Part I - Initial Problem

An application of GOS-BB to the problem resulted in an INDETERMINATE outcome (bounded but of unknown feasibility) after 20,000 iterations, 116,329.6 seconds, using a maximum of 36 boxes. The smallest box in the list was of width 1.58×10^5 . As this type of behaviour (few boxes and no small boxes) is typical of INFEASIBLE problems, the problem was rerun with a Phase I objective function. Similar results were obtained.

Part II - First Reformulation

The problem was then modified. Originally, the design variables were allowed to go from 0.0 to $+\infty$. As the difficulty was perceived to be with too large a search space, a more restrictive assumption was made for the design variables. The decision was made to allow for a maximum of 3 inches of thickness for each design variable and to enforce the minimum thickness allowed. Thus each design variable was allowed to vary between 0.0625

and 3.0. The stated solution for the original problem [26] lies within these bounds, and thus we have not eliminated the ‘best known solution’. Using local propagation (a constraint programming system method [12]), the bounds on some of the other variables were reduced to obtain a new set of bounds (see the alternative set of bounds for Floudas and Pardalos 11.1 in Appendix B.1.1).

When this new problem was tested an $IISU_{\theta} = \{c_2, l_1, l_2, u_2, l_3, u_3, l_4, l_9, u_9, u_{11}, l_{43}, u_{43}\}$ was found requiring 68 iterations, 36,356 function calls, a maximum of 35 boxes and 8.37 seconds. Examining these bounds and variables, x_1, x_2, x_{11} , and x_{43} are eliminated as sources of the cause of the infeasibility, as they are parameters that were allowed to vary by a small amount. This leaves only the variables x_3, x_4 and x_9 . As x_3 is a design variable we can eliminate it. The bounds generated for x_4 and x_9 from the new restrictions on the design variables were double checked and verified. This left c_2 as the only possible source of the infeasibility.

Part III - Second Reformulation

An assumption was made that this constraint contained a typographical error and the relation of the constraint was changed from $\leq$ to $\geq$. With this change the problem was rerun using $S1$ as the subdivision method (i.e., without LSGRG2 due to difficulties with numerical error exceptions with LSGRG2 for this problem) which resulted in an INDETERMINATE outcome (bounded but of unknown feasibility) after 2,000,000 iterations, after 74,454.78 seconds, using a maximum of 67 boxes. Only about $1.0 * 10^{-10}\%$ of the ‘volume’ of the Initial Box has been removed as infeasible. At this point the limitations of the GOS-BB have been met, as the problem space is still too large to effectively identify the correct outcome (although the results do point towards an INFEASIBLE outcome).

Thus, GOS-BB has identified one $IISU_{\theta}$, but fails to determine if other $IISU_{\theta}$ ’s exist (but information available seems to indicate that there are other $IISU_{\theta}$ ’s present). Thus, information (the $IISU_{\theta}$, the pattern of a typical infeasible problem, etc.) has been provided which seems to indicate that the model is faulty. This information would not have been available if a multistart stochastic method (multiple applications of a local optimization method) was applied by using a penalty function to bypass the difficulty with the constraints, were the only information that would be present is that no feasible point has been found. Thus, a re-examination of the original problem is required, starting with the constraints and variables in the $IISU_{\theta}$ that was identified.

Melville Shipping Ltd. Problem

This problem (see Appendix B.2.3) deals with ship motion in the surge direction, which is described by the following equation:

$$M_e \ddot{u} = A_2 u^2 (1 - w)^2 (1 - t) \rho D^2 + A_1 u (1 - w) (1 - t) n \rho D^3 + A_0 (1 - t) n^2 \rho D^4 - 0.5 C_r \rho S u^2 \quad (9.1)$$

The ship's steady state speed performance was desired, that is, when the acceleration, $\ddot{u} = 0$. The variables are: t the thrust deduction fraction, w the wake friction, and C_r the open water resistance coefficient. Thus, the values for which Equation 9.1 has a left hand side equal to zero are sought.

The problem was formulated as a global optimization problem by squaring the right hand side and looking for a global minimum equal to zero. The problem was tested with the following values for the parameters:

case	u	n	A_0	A_1	A_2
a1	5.5694	2.1667	0.239580	-0.308760	-0.107579
b1	6.6195	2.1667	0.334962	-0.354097	-0.070406
b10	8.0511	2.1667	0.511790	-0.403934	-0.043294

and $S = 2036$, $D = 4.8$, and $\rho = 1025$.

The results are tabulated in Table 9.5. For each parameter set, a large number of small boxes are generated (we have stopped after 30,000 boxes have been created). Of these boxes, about 29,975 of them are of width less than 1.0×10^{-4} and have functional values which include zero. Testing points within these boxes (see Table 9.6) produces alternate global minimum (to within at least 8 digits of accuracy for Equation 9.1). This suggests that there are an infinite number of global minima, likely forming a continuous sub-region. Flatness factors about the global minimum (adjusted to account for the squaring of Equation 9.1) are tabulated in Table 9.7 and indicate that the region about the global minima is unstable, that is, a small random perturbation to the incumbent solution will likely result in a large change in the value of the objective function.

Thus, information derived from analysis of the Set of Boxes plus that of the flatness factors, provides information on the number, and probable 'shape', of global minima as well as on the general flatness of the problem (testing the flatness of each additional global

case	global minimum (t, w, C_r)	functional value
a1	$(3.66311 * 10^{-3}, -2.16319 * 10^{-2}, 3.02415 * 10^{-3})$	$1.90582 * 10^{-19}$
b1	$(5.94704 * 10^{-1}, 2.050548 * 10^{-1}, 3.00100 * 10^{-3})$	$1.66018 * 10^{-19}$
b10	$(5.14944 * 10^{-1}, 6.047277 * 10^{-3}, 3.21220 * 10^{-3})$	$5.72594 * 10^{-19}$

Table 9.5: Melville Shipping Ltd. Problem - global minimum

case	global minimum (t, w, C_r)	functional value
a1	$(9.99769 * 10^{-3}, -2.04964 * 10^{-2}, 3.02518 * 10^{-3})$	$1.09775 * 10^{-18}$
	$(1.01077 * 10^{-2}, -2.08172 * 10^{-2}, 3.09123 * 10^{-3})$	$3.22084 * 10^{-18}$
b1	$(3.88284 * 10^{-3}, -6.68223 * 10^{-2}, 3.19896 * 10^{-3})$	$8.13999 * 10^{-19}$
	$(3.58987 * 10^{-3}, -6.66026 * 10^{-2}, 3.20338 * 10^{-3})$	$9.79170 * 10^{-19}$
b10	$(1.54174 * 10^{-2}, -9.42871 * 10^{-2}, 5.15146 * 10^{-3})$	$2.46995 * 10^{-18}$
	$(1.52711 * 10^{-2}, -9.42508 * 10^{-2}, 5.15273 * 10^{-3})$	$2.11758 * 10^{-18}$

Table 9.6: Melville Shipping Ltd. Problem - alternate global minima

minimum found can be used to determine if the flatness is likely to be poor for all of the global minima).

case	Flatness Factors for widths of:		
	$2.0 * 10^{-6}$	$2.0 * 10^{-3}$	2.0
a1	$1.65 * 10^7$	$1.65 * 10^7$	$1.67 * 10^7$
b1	$2.37 * 10^7$	$2.37 * 10^7$	$2.40 * 10^7$
b10	$3.43 * 10^7$	$3.43 * 10^7$	$3.47 * 10^7$

Table 9.7: Melville Shipping Ltd. Problem - Flatness Factors

9.2.6 Indeterminate Outcomes

For INDETERMINATE outcomes such as Unbounded 3 and the final sub-problem of Six Hump Back cs10, a set of boxes is produced that can be used to determine the proper outcome. For example, for Unbounded 3, we see that the box about the point $x = 2$ lacks a finite lower bound which is a result of the common root at this point for both the denominator and numerator for the objective function. Once this has been identified and removed, the new problem (Unbounded 4) produces an accurate result.

For the final sub-problem of Six Hump Back cs10, a set of 17 boxes remain to be tested whose volume is roughly 25% less than the volume of the Initial Box for the problem. These boxes can then be used as input to a different deletion filtering objective function to locate the feasible point and resolve the indeterminacy.

For Floudas and Pardalos 11.1 - Modified, we see that very little information is available for determining promising boxes, as the volume of the Set of Boxes is essentially equal to the volume of the Initial Box. The only useful information obtained is that the type of results provided indicate that it is likely that the problem is infeasible, and thus the set of constraints (13 out of 49) which have been used to eliminate infeasible boxes can be used as a starting point for further investigation.

9.2.7 Conclusions

From the testing we see that for small problems GOS-BB can identify HSU's, unbounded domain points for rational functions, and global minima. In addition, from the case studies we see that the formulation aids of flatness factors, % change in volume, smallest containing box, snap shot evaluations, etc. are useful from a modelling perspective. In addition, as snap shot evaluations provide periodic information, if the solution process terminates due to an unforeseen error, information is gained provided at least one snap shot has been taken. For INDETERMINATE outcomes, a small set of subregions are determined along with formulation information which can subsequently be used to determine the proper outcome of the problem. Furthermore, GOS-BB performs at roughly the same performance level of existing comparable methods.

Testing indicates that as the size of the problem increases the effectiveness of the method decreases. For problems of the size of 50 variables, formulation information is still attained, but the ability to obtain an accurate incumbent solution has been lost due to the 'curse of

dimensionality'. In addition problems with equality constraints (or feasible regions which mimic equality constraints) may also be difficult for GOS-BB to solve effectively. Furthermore, it does not provide a good sensitivity analysis.

Thus we conclude that GOS-BB meets (for small problems) the requirements for a global optimization system (see Section 4.4) except for the provision of a useful sensitivity analysis. Finally, we conclude that the global optimization system developed is useful from a modellers perspective as it provides useful information by which the appropriateness and correctness of a model can be determined.

Chapter 10

Concluding Remarks

In this chapter we provide a summary of the dissertation, including a summary of the contributions made as well as a discussion on topics for future research.

10.1 Dissertation Summary

The state of the art in global optimization methods does not address the provision of information for the evaluation of the correctness and appropriateness of a solution to a global optimization problem. As such information is useful from a modelling perspective, a methodology which addresses these issues is desirable. Thus the development of a global optimization system which will provide the modeller with formulation and modelling information for the solution of global optimization problems is an important step towards improving the state of the art.

The global optimization system, GOS-BB, developed in this dissertation has been shown to be able to meet most of the requirements for a global optimization system (specified in Section 4.4). It fails to provide an effective sensitivity analysis (see Section 7.9.2) and can only identify unbounded domain points for rational functions. In addition, it does not perform (in the sense of providing an accurate incumbent solution) well in the presence of equality constraints (or feasible regions which approximate the presence of equality constraints). It does perform well in the sense of determining the type of outcome of the given problem and in the provision of useful formulation information. Finally, testing indicates that as the size of the problem increases the effectiveness of the method decreases. For problems of the size

of 50 variables, formulation information is still attained, but the ability to obtain an accurate incumbent solution has been lost due to the 'curse of dimensionality'.

Thus, we conclude that GOS-BB can be used as a modelling tool for small problems, but requires improvements in order for it to be applicable to larger problems and to problems with equality constraints. Possible means by which these improvements may be attained are examined in Section 10.3.

In addition, the adaptive bounding methodology has been found to be a good strategy to use with GOS-BB, as it has the advantages of allowing for the usage of multiple bounding methods while, being able to identify a single dominant method for specified conditions, allowing for an efficient usage of the multiple bounding methods.

10.2 Contributions

The following contributions have been made to the area of Global Optimization:

1. A novel methodology for the formulation and solution of global optimization problems was developed. This global optimization system:

- provides for the solution of global optimization problems by verifying that exactly one of the following outcomes has occurred :
 - GLOBAL MINIMUM - The solution set consists of an incumbent solution, a lower bound for the global minimum and a set of regions guaranteed to contain at least one global minimizer.
 - INFEASIBLE - The solution set consists of a minimally inconsistent set of constraints.
 - DOMAIN UNBOUNDED - The solution set consists of a set of points and/or regions, which contain at least one point for which the objective function is domain unbounded.
 - INDETERMINATE- A set of *promising* sub-regions and/or a set of starting points for local optimization methods, etc., are identified.

For GLOBAL MINIMUM and DOMAIN UNBOUNDED outcomes, the size of the regions in the solution set is either within some *a priori* specified limits or as small as

possible subject to the time and/or space restrictions imposed by the automaton and/or user restrictions.

- It provides formulation aids such as stability measures.
- It addresses the problems that arise due to the use of the automaton which is required for the solution of Practical Global Optimization Problems, specifically the issues of: FINITENESS, DILEMMA OF EQUALITY, FUNCTION REPRESENTATION & RESTRICTED FUNCTION DOMAINS.
- It can be implemented on an automaton such that the space capacity of the automaton will not be exceeded during the execution and that it will complete the task within a reasonable amount of time.

2. Development of the adaptive bounding methodology.
3. Verification of a conjecture for the use of bounding functions for real valued functions with logical connectives.
4. The development of a test set for global optimization which includes infeasible and unbounded problems as well as feasible and bounded problems.
5. A line search method based on 1-dimensional branch and bound algorithm for constrained/unconstrained problems.
6. Modifications and improvements to Krawczyk's centered form.
7. An extension of Krawczyk's Centered Form to real valued functions of 1-dimension (and in principle to n-dimensions) with logical connectives.
8. Examination of the use of sub-division strategies within a branch and bound framework.

10.3 Future Research

Future research is proposed in the following topics:

1. Symbolic Manipulation

Manipulation of the representations of functions, aided by the use of symbolic manipulation and/or an rule based decision system, can possibly provide a means by which the following topics can either be dealt with or improved upon:

- Reduction of representations of rational functions into reduced forms by identifying common roots, rationalizing the functions, etc. This will help improve the robustness of the unbounded point detection algorithm as well as improve the accuracy of the bounding functions that make use of the representation.
- Elimination of equality constraints by identifying and solving for variables in linear terms of equality constraints and then in the substitution into the remaining functions and subsequent simplification of those functions. The potential benefit from the elimination of equality constraints is seen in the following example. Consider the following problems, Floudas and Pardalos No. 4.7 and Floudas and Pardalos 4.7 SUB (see Section 5). Solving the two problems to an accuracy of 5 digits produces the following results:

Problem Name	No. of Iterations	No. of function Evaluations	Max. No. Boxes	Time (sec)
FP 4.7	7,393	30,245	3,285	63.07
FP 4.7 SUB	44	566	4	0.8

The elimination of an equality constraint by substitution may introduce errors. If the equality constraint is linear, then the amount of error that is introduced is small. However, in general, the constraint will not be linear. If however a variable occurs exactly once in a linear term, then the constraint can be eliminated by solving for that particular variable using the above method. For example, the constraint $5x_1 + (x_2^3 x_3 / x_4^{1/2})^{1/2} = 0$ can be solved for in terms of the variable x_1 with only a small additional error to the system caused by the division by the coefficient of x_1 .

If no such linear terms occur, then solving for a variable will invariably add numerical instabilities. For example, numerical instabilities can occur if in solving for a variable, some variables are placed into the denominator of the resultant expression. When the denominator approaches zero, the expression may become unstable depending on how the expression is evaluated, etc.

2. A Control System to Control the Solution Process.

Research is desirable in developing a control system to control the types of subdivision methods, bounding strategies, objective functions, etc. to use. Information for the control system could be provided by the use of formulation aids, etc. examined at predetermined intervals, or when certain types of behaviour are detected. In this way, the information gained from the formulation aids can be fed back into the solution process, hopefully to improve upon its performance.

3. Sensitivity Analysis

Further research is required on developing an efficient and effective sensitivity analysis for a global optimization problem.

4. Restricted Function Domains

Research is required in developing effective methods by which the automated identification (and subsequent problem adaptation) of restricted function domains can be achieved.

5. Unbounded Domain Point Identification for Non-Rational Functions

Research is required into extending the unbounded point detection algorithm to functions which are not in reduced form and/or contain unary functions.

Possible means by which this can be accomplished are to devise a means by which information from the code for the unary function can be used by the unbounded point detection algorithm and by the use of symbolic manipulation to both identify the components of the function and to reduce them to a form suitable for application of the unbounded point detection algorithm.

6. Adaptive Bounding Methodology Parameter Determination

Further research is required to ascertain how to determine the parameters to be used for an arbitrary set of bounding functions. Possibly, a different set of conditions (to indicate when a best bounding function has been identified) may be required.

7. Efficient Local Optimization Usage

The use of a local optimization method within GOS-BB has been shown to be effective and beneficial. However, current strategies likely overuse the application of the local

optimization method. Thus, strategies for a more efficient usage of local optimization methods are desirable.

8. Deletion Filtering

Currently, the phase I objective function has been shown to be ineffective for certain problems. Thus there is a need for an improved objective function, or for a strategy that can recognize when the phase I is not performing well and then take appropriate action.

A possible reason for the ineffectiveness of the phase I is due to the inaccuracies that arise with the use of the bounding function over large regions. As the box size decreases, the accuracy of the bound increases and this biases the search towards small regions. The inaccuracies in the large regions may lead to a poor choice of regions to subdivide. Thus, a possible avenue of research is to explore different search strategies that aim to reduce the effect of the inaccuracies of large regions.

The phase I objective function is designed towards identifying a feasible point and is therefore unlikely to be efficient for infeasible problems. Thus research into alternate objective functions for deletion filtering which are also effective for infeasible problems may be desirable.

9. Extension to larger problems

Currently, only small problems can be solved by GOS-BB. Thus research is needed in means by which larger problems can be effectively searched. Possible avenues of research are parallel computing methods, inclusion of stochastic methods, etc.

10. Variable Interdependence & Constraint Programming Systems.

In some problems, the relationship between variables can be determined. Thus if two variables are dependent then when the bounds on one variable are reduced (indicated by the elimination of boxes) then it follows that the coupled variable will likely also have reduced bounds. One possible means of taking advantage of such a phenomena is to use knowledge of this type to guide the subdivision process. For example, if one variable has been reduced, then the subdivision process should focus on the variables that coupled to the reduced variable. Thus research into means by which such relationships can be identified and subsequently used to guide the subdivision process is desirable.

One possible direction is to determine a priority scale for the variables, and then use this scale to determine a subset of variables which are to be used for the subdivision process.

Another possible means is to use a constraint programming system [12] to reduce the feasible region by reducing bounds on variables by analyzing the constraint set.

11. Stochastic Methods within GOS-BB.

As stochastic methods provide good answers quickly they can be useful in generating a good incumbent solution quickly. The problem is that they are not practical in constrained environments. Thus research into the development of stochastic methods within a branch and bound framework is desirable.

One possible solution is to develop a feasible directions descent (fdd) optimization method as well as a clustering-like method to limit the number of applications of the fdd optimization method. Part of this solution requires a line search method applicable in a constrained environment. Such a line search is developed in Appendix D.

12. Branching Strategies

The branching strategies introduced in Section 8.2 show that the type of branching used affect the performance of GOS-BB overall. However, no single method seems to be better than any other method, suggesting that a strategy which employs various branching methods may be appropriate. Thus research is required in developing such a strategy.

Appendix A

State of the Art in Global Optimization Methods

In this Appendix we review the current state of the art in global optimization methods. We present a classification scheme for these methods a supplemental background section and then examine each class of methods in more detail.

We examine only those methods which attempt to identify a global minimum. Those methods which only attempt to identify a local minima (local optimization methods) are not examined as they do not address the problem correctly and standard texts exist which cover these methods in various levels of detail. Of these, Fletcher [25], Luenberger [71] and McCormick [73] are often cited. Examples of such methods are generalized reduced gradient methods, successive linear programming methods, sequential quadratic programming methods, Newton based methods, method of steepest descent, and so forth.

There are many ways in which one can classify the set of global optimization methods. These are discussed by Törn and Žilinskas [115], Lemar'echal [67] and Rinnooy Kan and Timmer [102] to name a few. We basically follow the classification used in [115], with some modifications, to obtain the following classification:

1. Global Boundary Optimization - These methods reformulate the problem such that the problem is reduced to finding the global minima on a boundary. Once the problem has been reformulated, methods are applied that take advantage of the localization of the global minima to the boundary of the feasible region.
2. Covering Methods - These methods attempt to cover the feasible region in some sys-

tematic manner. They invariably subdivide the feasible region and then attempt to locate (and then eliminate) those sub-regions which can be guaranteed not to contain any global minimum. The remaining regions are then subdivided and the process is repeated.

3. Generalized Descent Methods - These methods are a generalization of local optimization to the global optimization problem. The underlying principle is to systematically search the set of local minima so as either to locate all of the local minima, or in such a manner to ensure that each successive local minimum that is found is an improvement over the previous local minimum.
4. Stochastic Methods - These methods are based upon either an initial random sampling of points from the domain of the function or on a method by which a local search method can be stochastically modified so as to find another local optimum. For the former, each sample point can then be used as the initial point of a local optimization procedure, thus a set of local minima is produced, of which it is hoped that one is a global minimum. As this is inefficient, methods such as cluster analysis, etc. are used to reduce the number of local optimization procedures required. For the latter, the stochastic modification is either gradually reduced (as in simulated annealing) or is terminated after a specific stopping condition has been met (tabu search).

A.1 Background

A point x_b is on the boundary of a set S if every ϵ -neighbourhood of x_b contains at least one point of S and at least one point that is not in S . We note that a boundary point of S does not necessarily need to be an element of S .

If a real valued function $f : D \rightarrow R$ is continuous on D and has partial derivatives that are also continuous on D , then we denote this by $f \in \mathcal{C}^1$. In general, if the function has continuous partial derivatives of order p , then we denote this by $f \in \mathcal{C}^p$.

If $f \in \mathcal{C}^1$ then the function f is differentiable and we denote the gradient of f as ∇f or $f^{(1)}(x)$ with :

$$\nabla f = f^{(1)}(x) = \left(\frac{\partial f(x)}{\partial x_1}, \dots, \frac{\partial f(x)}{\partial x_n} \right)$$

A function $f : D \subseteq R^n \rightarrow R$ is called D.C. (the difference of two convex functions) on D if there exists two convex functions, $p : D \rightarrow R$ and $q : D \rightarrow R$ such that $f(x) = p(x) - q(x) \forall x \in D$.

If $h(x)$ is a convex function, then a constraint of the form $h(x) \leq 0$ is called a convex constraint, while a constraint of the form $h(x) \geq 0$ is called a reverse convex constraint. If $h(x)$ is Lipschitzian, then $h(x) \leq 0$ is called a Lipschitzian inequality.

Let x_l^* be a local minimum of a function, f in the presence of a constraint set C with constraint functions g_j and h_k . We assume that $f, g_j, h_k \in C^1$. Furthermore, let x_l^* satisfy a regularity condition (or constraint qualification) for the constraints. Then the Kuhn-Tucker Conditions state that there exists a vector $\lambda \in R^{n_h}$ and a vector $\mu \in R^{n_g}$ with $\mu > 0$ such that

$$\mu^T g(x_l^*) = 0 \text{ and } \nabla f(x_l^*) + \lambda^T \nabla h(x_l^*) + \mu^T \nabla g(x_l^*) = 0$$

If x is a point such that $h(x) = 0$ and $g(x) \leq 0$ and J is the set of indices j such that $g_j(x) = 0$, then x satisfies a regularity condition for the constraints if the gradient vectors $\nabla h_i(x)$ and $\nabla g_j(x)$, $j \in J$ are linearly independent. We note that there are other possible regularity conditions.

We also note that the Kuhn-Tucker Conditions are necessary but not sufficient. Thus a point which satisfies these conditions may not be a local minimum and that they require that the functions and constraints are differentiable.

A.2 Global Boundary Optimization

The basic idea behind these methods is to reformulate the problem into a form which guarantees that the global minima are found on the boundary of the feasible region. The boundary is then searched in a systematic manner to obtain the global minimum. The problem is still global in nature in that the boundary will in general have many local minima. The methods developed for these problems are designed to obtain the global minimum of the problem. We shall call these methods global boundary optimization methods. We now examine the conditions on the problem that are necessary for this to occur.

If the feasible region is convex and compact, and the objective function is continuous and strictly concave then the global minima are found only on the boundary of the feasible region. This type of problem is called a concave minimization problem. A subclass of this

type for which the objective function is a (non-convex) indefinite quadratic function and the feasible region is defined by a set of linear constraints is called quadratic programming. Methods such as cutting planes (where the feasible region is reduced by cutting off regions which do not contain the global minimum) and branch and bound based methods are used to solve this class of problems and are examined in [50, 53, 55, 88]. A detailed examination of Quadratic Programming can be found in [88].

Concave minimization and integer programming are related in that broad classes of integer problems can be formulated as equivalent (in terms of the optimal solutions) concave minimization problems [29, 55]. An important generalization of concave minimization is the D.C. programming problem which is defined as follows:

$$\min f(x) \text{ subject to } x \in C, g_i(x) \leq 0 \quad i = 1, \dots, l \quad (\text{A.1})$$

where C is convex and the functions f, g_i are D.C. on C (see Appendix A.1). It can be shown that every function, $f \in C^2$, is D.C. and that a real valued function defined on a compact (convex) subset D of R^n is the limit sequence of D.C. functions on D which converge uniformly in D [55].

By the addition of at most two variables, every D.C. program can be converted into an equivalent canonical D.C. program. The canonical D.C. program problem is defined as :

$$\min cx \text{ subject to } h(x) \leq 0, g(x) \geq 0 \quad (\text{A.2})$$

where h and g are convex functions and cx denotes the inner product. If we let H and G denote the regions defined by the corresponding constraints, and if we let ∂H and ∂G represent the boundary of the regions H and G , then it can be shown that the canonical d.c. program problem (for bounded H and for the case of $\min cx \mid x \in H < \min cx \mid x \in H \cap G$) that the global minimum lies on $\partial H \cap \partial G$. We note that if $\min cx \mid x \in H < \min cx \mid x \in H \cap G$ does not hold then the problem reduces to the convex minimization problem. The D.C. programming problem can then be solved by either the application of an outer approximation method, branch and bound methods or by a sequence of linear programs and line searches [55].

We now examine two methods which are used to solve global boundary optimization problems: outer approximation methods and relief indicator methods.

A.2.1 Outer Approximation Method

Outer approximation methods use successive inclusions of the feasible region to identify the global minimum. Each inclusion (or outer approximation) is defined by a set of constraints (not necessarily linear). The global minimum of the objective function over the current outer approximation, x_b , is then determined by an exhaustive search of the extreme points of the outer approximation (vertices if the outer approximation is a polyhedron). If the global minimum on the outer approximation is a feasible point of the original problem, the algorithm stops. Else a separating constraint is created that separates x_b from the outer approximation. This new constraint is added to the list of constraints to produce a new outer approximation and the procedure is iterated. The separating constraint is generated with the aid of knowledge of Lipschitz constants (or other similar bounds) for the constraints.

An indepth examination of these methods can be found in [53, 55].

A.2.2 The Relief Indicator Method

A different approach, due to Thach and Tuy [114] has been developed for the constrained global optimization problem with no equality constraints. The method defines a 'relief' function, $\Phi(\alpha, x)$ such that for an interior point of the feasible region, x , $\Phi(\alpha, x) > 0$, $= 0$ or < 0 according to $f(x) > \alpha$, $= \alpha$ or $< \alpha$. In addition, Φ is a D. C. function. One of the terms of this function is a generalization of the distance from x to a level set of the function (all the points of the function with the same functional value). This term can be constructed if the objective function and the constraints satisfy a Lipschitz constant, have bounded second derivatives, etc. Otherwise, this term is difficult to determine.

The algorithm starts from an initial feasible point, x^1 , and then at each iteration $\Phi(\alpha, x^i)$ is minimized over all feasible points. If the minimum value is zero then x^i is a global minimum. Else x^{i+1} is chosen as the point which minimizes $\Phi(\alpha, x^i)$.

In practice, an approximation to Φ is used to ensure that the solution can be obtained in a finite number of steps. This is usually solved by an application of an outer approximation method.

A.2.3 Discussion

Due to the rapid growth of the number of constraints and thus extreme points [55] that are maintained and subsequently searched, these methods are applicable only to small sized problems. While there are some constraint dropping strategies, typically, the number of constraints will be considerably larger than the number of variables. Thus these methods rely heavily on efficient methods by which the vertex sets are updated. The solution is complicated by the requirement to determine Lipschitz constants (or similar conditions). Thus these methods are restricted to problems of at best small size.

In addition, the feasible region must be full dimensional, that is no equality constraints are allowed. The requirement for Lipschitz constants usually restricts the problem to bounded problems. Thus these methods are applicable only to bounded and feasible problems. Furthermore, these methods are only applicable to those functions which are at least D.C., which effectively restricts these methods to functions which are twice differentiable.

A.3 Covering Methods

Covering methods are explicit exhaustive search methods. An underestimating cover of the objective function is created over the feasible region of the problem. The minimization of this cover leads to a lower bound on the global minimum. This cover is generated by first creating a screen of points which span the feasible region. Sub-regions of the feasible region can then be defined by using the screen points as the boundary points for the sub-regions. Each sub-region is then covered by creating a function which underestimates the objective function. The incumbent solution is taken to be the lowest functional value from the set of all feasible screen points. This value is of course an upper bound on the global minimum. Thus the difference between these two values is a bound on the amount of error in the incumbent solution.

We note that the underestimating cover for each sub-region is used to determine a lower bound on the objective function for that region. Thus the methods described in Section 2.4 can be used to determine such bounds. The spanning screen can be determined from the results of bounding the feasible region. A feasible region can be bound by a set of boxes which contain the feasible region by using the methods of Section 2.4 to bound the constraint functions and then use the resulting bounds to determine if the box contains any feasible

points [97].

The accuracy of these methods depend on the density of the screen. The finer the screen the more accurate the screen. However, a dense screen is computationally inefficient. What is desired is a screen which is dense only around the global minima and sparse elsewhere. This leads to the use of a branch and bound strategy to carry out the search for a global minimum.

A.3.1 A Branch and Bound Prototype

A prototype branch and bound procedure has been developed by Horst and Tuy [51, 52, 54, 55]. This is a formalization of the branch and bound strategy into the covering methods. A similar treatment can be found in Pinter [90].

We define a partition of a set N as follows. Let N be a closed subset of R^n and I be a finite set of indices. A set $\{N_i \mid i \in I\}$ of closed subsets of N is said to be a partition if $N = \bigcup_{i \in I} N_i$ and $N_i \cap N_j = \partial N_i \cap \partial N_j$ for $i, j \in I, i \neq j$ where ∂N_i denotes the relative boundary of N_i to N . By convention, $\inf \emptyset = \min \emptyset = \infty$.

Consider the global optimization problem of $\min\{f(x) \mid x \in D\}$ where $f : R^n \rightarrow R$ and $D \subseteq R^n$.

Step 0 Choose a box to enclose the feasible region, $N_0 \supseteq D$. Find a feasible point for that box, $S_{N_0} \in D$ and determine a lower bound for the box, $-\infty < \beta_0 \leq \min f(D)$. Set :

- the initial partition to be the box $\eta_0 = \{N_0\}$.
- the incumbent solution to be $\alpha_0 = \min\{f(S_{N_0}), \infty\}$.
- the overall lower bound to be $\beta(N_0) = \beta_0$.

If $\alpha_0 < \infty$ then choose an x_0 such that $f(x_0) = \alpha_0$. If $\alpha_0 - \beta_0 = 0$ (or $\alpha_0 - \beta_0 \leq \epsilon$) then stop, $\alpha_0 = \beta_0 = \min f(D)$ and x_0 is an optimal solution. Else go to Step 1.

Step k ($k = 1, 2, \dots$) At the beginning of the k^{th} step there exists the current partition η_{k-1} of a subset of N_0 which contains all of the global optima. In addition, for each element of the partition, $N \in \eta_{k-1}$ there exists :

- a set $S_N \subseteq N \cap D$, a lower bound $\beta(N)$ and an upper bound $\alpha(N)$ that satisfy the following :

- $\beta(N) \leq \inf f(N \cap D) \leq \alpha(N)$ if N is known to be feasible.
- $\beta(N) \leq \inf f(N)$ if N is uncertain.
- an incumbent solution, α_{k-1} and an overall lower bound β_{k-1} such that $\beta_{k-1} = \inf_{N \in \eta_{k-1}} \{\beta(N)\} \leq \min f(D) \leq \alpha_{k-1} = \inf_{N \in \eta_{k-1}} \{\alpha(N)\}$.

If $\alpha_{k-1} < \infty$ then a point $x^{k-1} \in D$ has been found for which $f(x^{k-1}) = \alpha_{k-1}$, that is, a feasible incumbent solution.

Step $k.1$ Delete all $N \in \eta_{k-1}$ which satisfy $\beta(N) \geq \alpha_{k-1}$. Let R_k be the collection of all of the remaining sets in the partition η_{k-1} .

Step $k.2$ Select a nonempty collection of sets $P_k \in R_k$ and construct a partition of every element of P_k . Let P'_k be the collection of all of the new partition elements.

Step $k.3$ Delete each $N \in P'_k$ for which it is known that $N \cap D = \emptyset$ or for which it is known that the $\min f(D)$ cannot occur. Let N'_k be the collection of all of the remaining elements of P'_k .

Step $k.4$ Assign to each $N \in \eta'_k$ a set S_N and a quantity $\beta(N)$ that satisfy the following :

- $S_N \subseteq N \cap D$.
- $\beta(N) \leq \inf f(N \cap D)$ if N is known to be feasible.
- $\beta(N) \leq \inf f(N)$ if it is uncertain.
- $S_N \supseteq N \cap S_{N'}, \beta(N) \geq \beta(N')$ whenever $N \subset N' \in \eta_{k-1}$.

Set $\alpha(N) = \min f(S_N)$.

Step $k.5$ Set $\eta_k = (R_k \setminus P_k) \cup \eta'_k$. Calculate $\alpha_k = \inf \{\alpha(N) \mid N \in \eta_k\}$ and $\beta_k = \inf \{\beta(N) \mid N \in \eta_k\}$.

Step $k.6$ If $\alpha_k - \beta_k = 0$ (or $\alpha_k - \beta_k \leq \epsilon$) then stop, $\alpha_k = \beta_k = \min f(D)$ and x^k is an optimal solution. Else go to Step $k + 1$.

The basic operation of this prototype can be summarized as follows. At the start of any iteration there exists a partition (a collection of elements) that contain the global optima. For each element of the partition a lower and upper bound for the least (feasible) functional value of the element is known as well as a set of feasible points.

From the bounds on the elements of the partition, bounds on the global minimum are determined. Once this is known, those elements which cannot contain a global minimum

(either due to the fact that its lower bound is greater than the current upper bound of the global minimum or that it is known that the element does not contain any feasible points) are eliminated.

A subset of the remaining elements is chosen for further partitioning. Within this partitioning phase, some of the new partitions may be eliminated for the same reasons as given above. For each new partition element a set of feasible points is determined as well as a lower bound on the functional values. New lower and upper bounds for the global minimum are determined from the updated set of bounds. If these are identical (or differ by an *a priori* determined amount) then the algorithm stops as a global optimum has been identified. In actual practice, it is not necessary for each element of the partition to have a known feasible set. However, if there is to be any reduction, some feasible points must be found in some of the elements of the partition.

The prototype has three basic operations which will affect its convergence and efficiency. The first is to decide upon a method by which to obtain the proper bounds on the global minimum. The lower bound is a direct bounding procedure, with possible methods being the use of Lipschitz constants, interval analysis, and convex underestimating functions (see Section 2.4).

The upper bound is determined by the existence of feasible points and their associated functional values. The algorithm can proceed without finding a feasible point. If this occurs then the current value for the upper bound on the global minimum is by default ∞ .

The second operation is how to choose which elements should be further partitioned. At each stage all or one element can be partitioned into smaller components. It may be advantageous to have small elements as, in many cases, the bounding operation will be more accurate as the size of the set decreases. However, the resultant cost may be in a large set of elements that are produced that cannot be eliminated. One approach is to determine the best candidate element, that is, the element which seems to have the best chance of containing the global minimum and partitioning this element only at each iteration. The difficulty with this approach is that the best candidate may not always be the best choice.

The third operation is to decide how to partition a set. The set may be simply split in two elements, or it may be split into many elements, of either identical size and shape or into various sizes. The only requirement is that each set that is created must be further partitionable and that the bounding operation must be applicable to each of the new sets

created.

We now examine how the prototype detects infeasible or unbounded problems. We note that domain unboundedness (see Section 3.2) is not allowed as the objective function is constrained to being defined and continuous over R^n . If the feasible region is compact, then the problem cannot be unbounded.

The prototype detects an infeasible problem when the partition set becomes the null set. The problem becomes feasible as soon as a single feasible point has been identified. Unfortunately, both the detection of a feasible or an infeasible problem may require an infinite number of steps. Thus in the worst case this prototype will asymptotically converge to a decision on the feasibility of the problem. This difficulty may further be compounded by the possibility of an unbounded problem.

The problem may be unbounded at ∞ . This will be indicated by the fact that the lower bound is at $-\infty$. This unboundedness at ∞ can only be detected by the prototype if the upper bound on the global minimum becomes $-\infty$ as well. However this is usually detected only by an infinite process as some of the variables must approach values of $\pm\infty$. In addition it requires that a feasible point has been identified, as otherwise the upper bound will be at $+\infty$ and the lower bound at $-\infty$. In the worst case, the identification of the feasible point may only occur at the point where the problem is unbounded. That is, in the limit it is determined that the problem is unbounded, whereas after any finite number of iterations, the problem could be either unbounded, infeasible or a finite solution exists.

The problem lies in the fact that the prototype was designed primarily for problems which are feasible and bounded from below. Furthermore, from a practical perspective (finite time and space requirements) the prototype has the following difficulties:

- The size of the partition may become unwieldy even for relatively small problems.
- The possibility of logical errors introduced by the need to terminate an infinite sequence. For example if the feasibility of the problem is still unknown when the sequence is terminated then reporting that the problem is infeasible may not be true.
- The generation of feasible points, lower and upper bounds depends on the introduction of external methods.

A.3.2 Covering Method Algorithms

The earliest covering methods were those developed by Pijavskij [89] and Shubert [110] which find the global minimum of a function of one variable over some finite interval. These methods assume that the objective function satisfies a Lipschitz condition, which allows for the construction of an underestimating function consisting of cones centered at the chosen screen points. A variation on this algorithm, due to Brent [11], assumed the existence of a finite second derivative and then used this to construct an underestimating function which used parabola's instead of cones.

These methods were extended to the multivariate case by Evtushenko [24] who assumed a Lipschitz condition based on the maximum norm to produce bounds on boxes (hyper-rectangles) and Mladineo [77] who used assumed a Lipschitz condition to produce multivariate cones for which a global minimum could be found by solving a system of n linear equations and one quadratic equation (n being the number of variables).

Interval Analysis Based Algorithms

Interval analytic methods for solving global optimization problems are branch and bound based algorithms and thus are specific implementations of the prototype procedure of Horst and Tuy.

Several interval analysis based branch and bound algorithms for unconstrained global optimization have been developed (see [40, 41, 56, 59, 81, 97, 99, 111]). An exception to this is the algorithm due to Ratschek and Voller [99] which makes use of an extended interval arithmetic which allows intervals to be defined with infinite widths (for example, $(-\infty, a]$). While this increases the amount of computation needed it does allow for the solution of more general unconstrained problems. That is, problems that are not restricted to a box.

In common to these methods is a branch and bound algorithm which partitions the feasible domain into a list of boxes, one of which is chosen for further subdivision (by dividing the longest edge into two equal lengths). Bounds for the range of the function over a box are found by using interval analysis. At this point the various methods diverge. Some methods order the list of boxes by size and some by age and then choose the first box on the list as the next box to analyze.

Various tests, called acceleration techniques, are used to eliminate boxes from the lists or to improve the value of the incumbent solution. The most common acceleration technique

is called the midpoint test which evaluates each box at its midpoint to generate a set of known values, with the incumbent solution being the minimum functional value of this set. The functional value of the incumbent solution is then used to eliminate any boxes with lower bounds strictly greater than the incumbent solution. We now briefly identify other acceleration devices that are commonly used.

The monotonicity test checks each feasible box to see if it is monotonically increasing or decreasing. If so, then the global minimum can only occur at one of its boundaries and thus the rest of the box can be eliminated leaving the requirement that the boundaries need to be checked.

Methods used to generate 'good' incumbent solutions vary as well, with some using interval Newton methods to find the set of critical points and others using local minimization procedures on the midpoint of each box that is generated. Hansen [40, 41] has developed a 'quadratic method' which tries to prune a box of regions for which it can be shown that the functional values are greater than the incumbent solution. This method makes use of the Taylor expansion of order 2 for the function and involves the solution of a quadratic equation.

As with all branch and bound methods, the main difficulty is in the number of sub-boxes that are required to be maintained in the list. The addition of acceleration techniques reduces this problem somewhat but it does not eliminate this problem.

An algorithm for constrained global optimization has been developed by Ratschek and Rokne [97]. This algorithm is based on an algorithm for finding an inclusion for the feasible region. If a box has been determined to contain feasible points, then a bound on the global optima can be obtained and on the incumbent solution. The algorithm branches by systematically dividing up the space R^n into smaller boxes. Testing is carried out to determine if the current box is infeasible (upon which it is subsequently eliminated) or if it is feasible. For a feasible box, a bound is calculated and a comparison is made to the incumbent solution (this may result in a new best value and the elimination of a box). Any remaining boxes (as well as any indeterminate boxes, with respect to feasibility) are slated for further subdivision in the next iteration of the algorithm. If no more boxes need to be checked or if all of the remaining boxes are smaller than an *a priori* specified width the algorithm ends.

The acceleration devices mentioned above can be used with this algorithm, with some modifications to account for the need to determine the feasibility of a point or region, etc.

3 of/de 3

PM-1 3½"x4" PHOTOGRAPHIC MICROCOPY TARGET
NBS 1010a ANSI/ISO #2 EQUIVALENT

PRECISIONSM RESOLUTION TARGETS

Ratschek and Rokne note that if equality constraints are present (or equivalently if the feasible region is of a lower dimensionality than R^n) then the algorithm may fail to determine the feasibility of the problem when implemented on an automaton. It is suggested that the use of a test of existence (see Section 3.3.2) will in practice be beneficial, although no guarantee can be given for the determination of the feasibility of the problem.

In [121] the boxes are ordered by a weighted measure of objective function values and a feasibility measure of the box.

A.3.3 Discussion

The covering methods are limited to small sized problems due to the large number of sub-regions that are generated. These methods rely heavily on the existence of methods by which to bound the range of the function over some sub-region and as such have been developed for those classes of functions for which a bounding function is available. Those methods which rely on a Lipschitz constant are limited in that in general it is difficult to determine an accurate Lipschitz constant [55] while those methods which use interval analysis are limited to the classes of functions for which interval analysis is applicable (codeable functions).

The effectiveness of these algorithms depends on the quality of the methods used to bound the range of the function as well as on the ability to identify a good incumbent solution quickly.

The branch and bound prototype for these methods allows the method to be used on both infeasible problems and on problems that are unbounded at ∞ in addition to problems with a finite global minimum.

A.4 Methods of Generalized Descent

These methods attempt to systematically search the set of all local minima, the global minimum being the least element of this set. There are three main branches of these methods: trajectory, penalty and interval Newton's methods.

A.4.1 Trajectory Methods

The trajectory methods are based on the solution of a differential equation which describes the motion of a point mass on the surface of the function. As such they have been designed primarily for unconstrained optimization (and in practice to variable-bounded optimization due to the requirements of the automaton used to solve for the trajectories). This trajectory is invariably calculated by the use of some numerical integration technique such as the Runge-Kutta routines. Of these methods, those of interest are the golf methods and Branin's method.

The golf methods solve a differential equation of the form :

$$m(t)\ddot{x}(t) - n(t)\dot{x}(t) = -\nabla f(x(t)) \quad (\text{A.3})$$

where $m(t) \geq 0$ is the mass of the golf ball, $n(t) \leq 0$ and $-n(t)\dot{x}(t)$ is a dissipative force. Various methods arise dependent upon the choice of the parameters and the initial conditions to specify the starting position and velocity of the golf ball. By varying the parameters, the trajectory can be limited to converge to local minima smaller than some specified amount [37, 57, 58]. The problem is in determining the proper specified amount. In addition, the success of these methods is not guaranteed as very steep local minima can trap a golf ball, even for good choices of the parameters.

Branin's method [10] tries to find all of the solutions to the system of equations $\nabla f(x) = 0$. The solutions to the above equation are found by using the trajectories of the following differential equations :

$$\frac{d\nabla f(x(t))}{dt} \pm \nabla f(x(t)) = 0 \quad (\text{A.4})$$

However, problems arise in that the method is not globally convergent and that not all trajectories pass through all of the local optima. This means that it is difficult to determine how many trajectories need to be made to ensure that all of the local minima have been found.

A.4.2 Penalty Methods

In this branch of methods each time a local optimum is found, the function is modified so as to remove that local optimum. The function is then minimized again and the process

is repeated until all of the local optima have been found. There are three approaches of interest. An algebraic approach, tunneling methods and filled functions.

The algebraic approach (see Goldstein and Price [34]) is applicable to algebraic functions and is based on the Taylor expansion about a local minimum to define a new function. This new function is then searched for a particular class of solutions, any of which is a 'good' starting point for finding an improved local minimum. The above procedure is then iterated until a 'good' starting point can no longer be identified. This type of method suffers from the problem that the function to be minimized tends to flatten as the algorithm progresses.

Tunneling methods are based upon the idea of alternating applications of a local optimization algorithm with the minimization of a tunneling function. The tunneling function is designed so that the previously located local optima are penalized if they are approached (see [35, 68, 119]). The method was designed for variable-bounded optimization.

The tunneling function is minimized by some method to obtain some local minimum. If the functional value of this minimum is higher than the best solution to date, then a parameter in the tunneling function is modified and the modified function is minimized. However, the effect of this modification is to flatten the tunneling function and care must be taken to minimize this effect. This is due to the fact that it is difficult to locate a local minimum of a nearly flat function.

The filled function methods [27, 28]) are based on initially finding a local minimum and then creating a 'filled' function which has a local maximum at the location of the local minimum of the original function. In addition, if the parameters are chosen appropriately, this filled function will not have any minima or saddle points which correspond to local minima of the original function with functional values greater than the incumbent solution. The minimum of the filled function is then used as the starting point of a local optimization procedure to obtain a new local minimum. If the parameters have been correctly chosen then the new minimum will improve upon (or at worst be equal to) the incumbent solution.

The next filled function is constructed by using the newly identified local minimum. A sequence of local minima is produced that converge towards the global minimum.

A.4.3 Interval Newton Based Methods

The interval Newton's method is due to Moore [81] and has been built upon by various authors, [39, 43, 44, 64, 79] to name a few. The interval Newton's method is an extension of

Newton's method to intervals and is used to solve variable-bounded optimization problems.

The basic equation that is solved is that of $f(x) = (f_1(x), \dots, f_n(x)) = 0$, by using the Mean Value Theorem as applied to intervals, this equation is transformed to $J(X_i)(x_i - Y) = f(x_i)$ where $J(X_i)$ is the natural interval extension of the Jacobian matrix, $J_f(x)$. At each iteration this equation is solved for Y or more exactly for an interval that contains the interval Y .

A set of disjoint intervals are produced whose union contains all of the solution set of $f(x) = 0$. However, each generated interval is not guaranteed to contain a solution point. Thus an additional procedure must be found to verify the existence of the solution points.

The interval Newton's method can be used to solve a constrained optimization problem as well as the variable-bounded optimization problem. All that is needed is a set of equations that describe the set of local minima. For example, consider the problem of finding the global minimum of a function $f(x)$ subject to $g_j(x) \leq 0$ for $j := 1, \dots, n_g$ and $h_i(x) = 0$ for $i := 1, \dots, n_h$ where $g(x) = (g_1(x), \dots, g_{n_g}(x))$ and $h(x) = (h_1(x), \dots, h_{n_h}(x))$. If the Kuhn-Tucker conditions A.1 (see Appendix A.1) are valid they may be used to define a function, $F(x, u, v)$ as follows:

$$f(x, \lambda, \mu) = \begin{bmatrix} (\nabla f(x) + \lambda^T \nabla h(x) + \mu^T \nabla g(x))^T \\ \mu_1 g_1(x) \\ \vdots \\ \mu_{n_g} g_{n_g}(x) \\ h_1(x) \\ \vdots \\ h_{n_h}(x) \end{bmatrix} \quad (\text{A.5})$$

If $f(x, \lambda, \mu) = 0$, $g(x) \leq 0$ and $\mu \geq 0$ then the point (x, λ, μ) is a Kuhn-Tucker point of the constrained global optimization problem [103]. The solution sets for $f(x, \lambda, \mu) = 0$ are generated by the interval Newton's method. We note that the solution sets are really bounds on the sets of solutions and thus the existence of the solution points themselves must still be proved.

Methods for proving the existence of a solution for the interval Newton's method can be found in [2, 82, 103, 109]. These methods are unfortunately not robust. That is, they do not always provide a means by which the existence (or lack thereof) of a solution point within an interval can be proved.

We examine the application of the interval Newton's method to finding all of the local minima of an unconstrained function. Consider a function $f : X \rightarrow R^n, X \in I^n$. The basic solution method is as follows:

1. Set $X_0 = X$.
2. For $n = 0, 1, 2, \dots$ Do
 - (a) Choose $x_n \in X_n$.
 - (b) Determine a superset Z_{n+1} of the solution Y_{n+1} of the linear interval equation with respect to Y

$$J(X_n) * (x_n - Y) = f(x_n) \quad (\text{A.6})$$

where $J(X_n)$ is the natural interval extension of the Jacobian matrix $J_f(x)$ to J_n .

- (c) Set $X_{n+1} = Z_{n+1} \cap X_n$

The set X_i , represents the enclosure of a possible element of the solution set. At any iteration, this set may be split into two or more intervals. When this occurs, treat each new interval as if only it existed. Thus, the method branches whenever a split occurs. The steps that are used to manage this branching have been omitted for clarity.

At each iteration an interval is either split, reduced or eliminated. A stopping condition is invoked to terminate each of the branches. What remains is a set of disjoint intervals, the union of which contains the set of points satisfying $f(x) = 0$.

In general the efficiency of this method relies on how a solution is found to equation A.6 (see Ratschek and Rokne [97]). Several procedures have been developed to solve this difficult problem, one of the most effective being the method developed by Hansen and Greenberg [43]. Basically the procedure to solve equation A.6 is based upon the following four procedures:

1. A preconditioning step in which an approximation of the inverse of $J(X)$, B , is multiplied to both sides to get $BJ(X)(x - Y) = Bf(x_n)$. We let $M = BJ(X)$ and $b = Bf(X_n)$.
2. A relaxation procedure in which the current values of X and x are updated so as to reduce their size in the case of X and to get closer to zero in the case of x . As X is made smaller it can happen that the interval is split into two or more disconnected

sub-intervals. This step is used to solve the equation $M(x - Y) = b$. If B is close to the inverse of $J(X)$ then the matrix M is close to the unity matrix, which makes this procedure easier.

3. A local iteration procedure brings the results of the previous step closer to a solution. This is carried out by using a non-interval quasi-Newton's method. This prepares the problem for the final step which requires that the current value is close to the final solution.
4. A Gaussian elimination procedure is then applied to determine an even better solution. If the previous step brings the solution close enough the use of the Gaussian elimination procedure will remain valid.

A.4.4 Discussion

The methods of generalized descent have been developed primarily for unconstrained optimization (and in practice for variable-bounded optimization). As such they assume an inherent feasibility in all of the points it generates. In addition, the detection of an unbounded problem is by chance, usually when a local optimization method (or a trajectory) fails to terminate properly.

The modification to constrained optimization is possible by using penalty functions, etc. to transform the problem into an unconstrained optimization problem. The use of penalty functions will add some error due to the added complexity of the modified objective function.

These methods invariably require that the objective function is at least differentiable (in most cases twice). They are mostly impractical for problems with large numbers of local minima as they may be required to identify all of them.

Specifically, the trajectory methods have difficulties in determining the proper values for parameters and in ensuring that a global minimum has been found. The penalty methods have difficulties in that as the number of local minima identified increases the function to be minimized tends to become flat, thus making subsequent local optimizations difficult and fraught with possible errors such as cycling through previously found solutions. The interval Newton's method is handicapped by the fact that it only bounds the possible solution and thus requires an additional method by which the existence of a solution can be made. In addition, these methods have the additional problem of having to maintain a possibly large

list of boxes.

A.5 Stochastic Methods

Stochastic methods (with the exception of simulated annealing and tabu search methods) use as an uniform sampling of the feasible region to provide a 'cover' of the complete feasible region. What is required is a distribution of the sample such that each subregion has a strictly positive probability of obtaining a sample point. Thus the probability density function for the sample must be non-zero for each subregion. As most means of obtaining a sample from a probability density function explicitly or implicitly use an uniform sample, in effect, an uniform sample must be attainable for the feasible region.

As a uniform sampling can only be generated for a box (or interval) the minimum requirement for the use of these methods is that the feasible region is contained within a box. If this is the case then a uniform sampling of the box can then be made into a uniform sampling of the feasible region by simply rejecting all of those sample points which are infeasible. If this procedure is to be efficient then the feasible region should be contained in as small a box as possible and that the excess 'volume' of the box that is infeasible should be relatively small.

Unfortunately this is not in general true as the feasible region for a constrained global optimization problem may be disjoint and widely spaced. In addition the presence of any equality constraints effectively reduces the dimension of the feasible region which translates into a probability of zero that any sample point will also be feasible. The requirement for a uniform sample has effectively restricted the practical application of these methods to variable-bounded optimization problems, or at best to constrained optimization problems with well defined feasible regions (for example, feasible regions defined by a set of linear inequalities, etc.). Transforming the problem by the use of penalty functions is undesirable due to the fact that it becomes difficult to determine how to define the box that encloses the feasible region [78] and too large of a choice of box will lead to inefficiencies due to excess infeasible 'volume'.

Inherent in these methods is the assumption that there exists a finite distance, ϵ , such that for any two local minima the distance between the two minima is greater than or equal to ϵ . Furthermore, as the feasible set is a compact set this implies that only a finite number of local minima exist within the feasible region. In addition, many of these methods assume

that the boundary of the feasible set does not include any of the global minima.

We examine six classes of stochastic methods : pure random search, multistart, clustering methods, simulated annealing methods, genetic algorithms, and tabu search methods.

A.5.1 Pure Random Search

The simplest stochastic method is called pure random search and consists of the uniform sampling of N points of the feasible region and the subsequent evaluation of the objective function at these points. The lowest functional value is taken as the incumbent solution. If a stopping condition is not satisfied the sample size is enlarged. If the objective function is continuous, then due to the uniform sampling the incumbent solution converges to the global minimum as N goes to infinity. This is a very inefficient algorithm and as such is of little practical value.

A.5.2 Multistart

A simple modification to the pure random search is to apply a local optimization algorithm to each element of the sample. This algorithm is called multistart. A stopping condition is then used to determine if the algorithm is to stop or to increase the sample size. As the size of the sample is increased the probability of finding a global minimum goes to 1 [107]. The obvious problem is that in general each local minimum will have been identified many times over in the course of the algorithm which is a serious waste of computational resources.

As these methods spend equal amounts of effort in each region the obvious way to improve these methods is to reduce the amount of effort that is spent in unproductive regions. Thus various methods try to evaluate regions of the search for their potential (using Lipschitz constants [5], their distance from the incumbent solution [47]) and to direct the sampling so that the more promising regions are biased in the sampling. A more complete record of these variations is given in [115].

An algorithm developed by Snyman and Fatti [113] is a modification of the multistart method but replaces the local optimization routine with a trajectory method. It makes use of a Bayesian stopping rule. The advantage of this method is that each trajectory will produce several local minima, instead of the single local minimum produced by a standard local optimization method.

Stopping Conditions for Multistart algorithms

The stopping condition for a multistart algorithm can either be passive or sequential. Passive stopping conditions are derived from information on sample sizes and on any *a priori* information, whereas sequential stopping conditions make use of information that is available from the optimization process.

Passive stopping conditions are either quite simplistic or they are dependent upon some previous knowledge or assumptions about the local minima in the feasible region. For example, knowledge may be required about the probability of finding a point in the region of attraction of the global minimum or on the number of local minima, etc.

Sequential stopping conditions use either ordered statistics to develop *a posteriori* confidence intervals or Bayesian statistics are used to develop posterior distributions, expectations, etc. of various parameters. These distributions and expectations can then be used to develop stopping rules that depend on assumptions about the cost and benefit of further sampling. An example of an ordered statistic is from Golden and Alt [17, 33] where a confidence interval for the global minimum is given based on a random sample of iterative solutions and an estimate of the global minimum.

The Bayesian approach is to develop distributions for a set of unknown parameters by using an initial guess and then modifying the guess as more information is obtained from the optimization process. A set of Bayesian stopping conditions for the multistart algorithm have been developed by Boender and Rinnooy Kan [9] using parameters for the number of local minima and the relative volumes of their regions of attraction. Other possible stopping rules can be developed that take into account the cost of sampling [9].

A.5.3 Clustering Methods

The next logical step is to realize that each region of attraction needs only to be examined once for its local minimum to be found. Thus the idea of these methods is to determine these regions of attraction by using some clustering procedure and then to apply a local optimization procedure to one point of the cluster. In this way the number of applications of the local optimization method is minimized.

There are of course many ways by which a clustering procedure can be introduced into the multistart methods. A general algorithm [115] is given as follows:

Step 1 Sample points in the feasible region.

The initial sample is taken over the feasible region and as the goal is to find a global optimum, a uniform sampling is used. The number of sample points to be used must be determined and this will depend on either some predetermined knowledge (or assumptions) about the function to be optimized.

Step 2 Concentrate the sample to obtain groups around the local minima.

In this step the goal is to try to concentrate the sample points about local minima to aid the clustering algorithm. This can be accomplished by applying a few steps of a local optimization procedure to each point [116], by removing a predetermined number of the highest points [8] or by some other similar procedure or combination of such procedures.

Step 3 Form clusters using some clustering method.

A seed point (usually a local minimum) is used in conjunction with a threshold distance to determine membership within a cluster. Some points may not be found to be elements of any clusters.

Step 4 If a stopping condition is met, go to step 6.

In general the stopping conditions are determined from information on the number of local minima that remain, or on how the previous solution relates to the new solution.

Step 5 Transform, sample for the next iteration, go to step 2.

A subset of the current sample (usually the local minima found or a subset of low values) is mixed with a set of newly generated points.

Step 6 Perform final computations and stop.

This may entail an application of a local optimization method to one or more sample points depending on the use of local optimization methods in the previous steps.

The actual formulation of a cluster does not necessarily need to be carried out if a testable condition is available to determine if a point is a member of a cluster. That is, the clustering is implicitly carried out. This implicit clustering is used in the multi level single linkage algorithm developed by Rinnooy Kan and Timmer [101].

Multi Level Single Linkage Algorithm

Consider the following problem:

$$\min_{x \in S} f(x) \quad (\text{A.7})$$

where $S \subset R^n$ is convex and compact and $f : R^n \rightarrow R$.

Let $lm(S)$ denote the Lebesgue measure of the set S . The Lebesgue measure can be thought of as the 'volume' of the n -dimensional figure. We define a threshold distance, r_k as follows:

$$r_k = \pi^{-1/2} (\sigma lm(S)) \Gamma(1 + \frac{n}{2}) \frac{\log k}{k}^{1/n} \quad (\text{A.8})$$

where σ is a parameter usually set to 4 or 2.

For a set of points X^* , let $X_v^* = \{x \in S \mid \|x - x^*\| \leq v, x^* \in X^*\}$. Let Q_τ be the set of points that are within a distance τ of the boundary of S .

The multi level single linkage algorithm is then as follows:

- Step 1 Choose values for N , σ , v , τ and γ . Set $X^* = \emptyset$
- Step 2 Choose uniformly N points from S and evaluate these points. Choose the fraction γ of the lowest functional values of the sample N . Set $k = N\gamma$.
- Step 3 Start a local optimizer from each new point, x_i unless $x_i \in X_v^* \cup Q_\tau$ or if there exists a sample point x_j with $f(x_j) < f(x_i)$ and $\|x_j - x_i\| \leq r_k$. Add any new stationary points to X^* . Set $w = |X^*|$, the number of local minima.
- Step 4 If $w(n-1)/(n-w-2) - 1/2 \leq w$, then stop (the estimated number of unknown local minima is 0). Else sample a new point, set $k = k + 1$ and go to Step 3.

The algorithm is relatively easy to implement and according to testing carried out in [101] quite effective. However, Step 3 requires that the closest neighbouring point of a given point must be found. A procedure called Spiral Search [7] can be used to do this, but this procedure is only practical for dimensionality less than 10 at best. Thus this algorithm can only be used for problems of small size.

Other problems arise in that in order to use the Bayesian stopping condition, the algorithm can not start a local optimization procedure in the neighbourhood of the boundary of

S (the feasible region) or in the neighbourhood of a known local minimum (that is, if the point is in the set X_v^*). Thus the choice of τ and v may eliminate the possibility of finding the global minimum.

A.5.4 Simulated Annealing

Simulated annealing is a stochastic method which is based upon the physical annealing process. Metropolis et al [75] proposed a method whereby the low energy states of a solid can be obtained by a simulation process. The general model is that of a set of particles in a heat bath by which the temperature of the particles can be adjusted. The goal is to coax the particles into their lowest energy state in some manner. As the temperature is increased the particles are melted and thus can flow freely. When the temperature is decreased the particles freeze into a solid. If the cooling phase is gradual enough the particles will freeze into their lowest energy state.

The basic operation of a simulated annealing algorithm is as follows. An initial feasible point and temperature is chosen. New points are chosen by randomly defining a new point, usually by determining a random direction and then stepping in that direction for some specified distance. The new point is accepted if its functional value is lower than the current value. If the functional value is higher then it is accepted with a probability that depends on the temperature (the higher the temperature, the higher the probability of acceptance). Once an 'equilibrium' condition is satisfied, the temperature is lowered and the process iterates until a stopping condition is satisfied. In some cases, a final local optimization procedure is applied to the final result, as the 'equilibrium' condition may not guarantee that it has stopped at a local minimum.

The determination of proper conditions for 'equilibrium' and for the overall stopping condition is difficult. Parameters which determine the initial temperature as well as the rate at which the temperature is decreased are usually determined by making an initial sample of the function. The rate at which the temperature is decreased is then further modified as the method progresses.

Implicit in these methods is the ability to find a new feasible point in some random fashion. Thus these methods have been applied only to unconstrained or variable-bounded optimization problems or to constrained optimization problems with well defined feasible regions.

Practical implementations [18] seem to indicate that these methods are applicable only if few local minima are present and the dimension of the problem is small. Its performance was seen as comparable to that of the multi level single linkage method. Variations of the basic algorithm deal with how to change c (after each new state, after n new states have been found, etc.), how to calculate the probability of acceptance, etc. Several examples of these methods can be found in [4, 18, 63, 117].

A.5.5 Genetic Algorithms

Genetic algorithms [32, 120] are methods based upon the mechanics of natural selection and genetics. A small representative uniform sample (or population of points) is generated. This sample is then evaluated against the objective function and the 'best' are then modified by procedures based on genetics to produce a new generation of points. The process iterates until an *a priori* determine stopping condition (for example, no improvement after n iterations) has been met.

A.5.6 Tabu Search

Tabu search based methods [30, 31] are actually a set of procedures that can be applied to search methods so as to help direct the search. In particular, these procedures aim to reduce the chance that the search method becomes trapped in either at local optimum or in a previously searched region. They do so by maintaining a list of tabu items or control mechanisms, that is, a list of items that are not to be used by the search mechanism. For example, the list could be a list of directions in which a search is not to be made. This list is maintained and updated and can act like a short term memory for a search method. Other structures have been developed which can provide a long term memory for the search mechanism. These procedures have been developed primarily for combinatorial problems but have been extended to global optimization type problems.

A.5.7 Discussion

The stochastic methods are in general easy to implement. From the results of comparison testing they range from ineffective (Pure Random Search) to reasonably effective methods (Multi Level Single Linkage). A major problem is that one can not guarantee that the final

result is indeed a global optimum. A probabilistic measure of the amount of uncertainty in the result is available, but an absolute error is impossible. For some applications this may be sufficient but for critical problems where the exact global optimum is required it is questionable.

A second problem is that they are also subject to the 'curse of dimensionality'. As the number of dimensions increases either the size of the initial random sample must greatly increase to 'cover' the space or (if a clustering procedure is used) the computational expense of finding the nearest neighbour greatly increases. Thus these methods are practical only for at best small sized problems, although both the simulated annealing and genetic algorithm methods have been applied to large problems.

It is suggested in [101] that the 'curse of dimensionality' may perhaps be countered by the use of Bayesian analysis to develop new methods based upon the Multistart algorithm. This is due to the fact that Bayesian analysis is dependent more on the number of local minima and the relative sizes of the regions of attraction than on the dimensionality of the problem. A suggested possibility is to use non-uniform sampling procedures. However, it is noted that currently the use of Bayesian analysis requires that the sampling is carried out uniformly and the process of uniform sampling is subject to the 'curse of dimensionality'.

A third problem for these methods is that they have been developed primarily for unconstrained (or variable-bounded) optimization problems. While the methods can be used for constrained problems, these methods do not directly address the problem of how to determine a uniform sampling (or even of a random sample) of an irregularly shaped feasible region and thus the application of these methods to constrained problems can be difficult.

Finally, the simulated annealing and genetic algorithm methods rely on the proper setting and modification of algorithm parameters. This means that multiple applications with varying parameter settings or modification procedures may be necessary in order to obtain a good result.

Thus these methods avoid the issue of feasibility and identify unbounded problems only if an unbounded point is sampled or if a local optimization method fails to terminate properly.

A.6 Conclusions

As solving a general global optimization problem is difficult, research has been directed at solving special sub-classes. As the methods improve, the classes of problems that are being investigated expands. Typically, the classes of problems are determined by the conditions placed on the objective function and on the constraints. For example, the objective function and the constraints may be restricted to certain classes of functions, such as convex, continuous, twice differentiable, etc. Except for simple classes of functions, such as polynomial, etc., the determination of continuity, differentiability or a Lipschitz constant is difficult. As well, the set of constraints may be required to be a null set or restricted to a set of simple upper and lower bounds on the variables.

This approach has resulted in the development of a set of methods, each of which is applicable to some sub-class of problems. Methods for a general class of problems are usually less efficient than those for a special class of problems. In addition, often the performance of any method over the class of problems for which it is applicable is not consistent. That is, it will perform better for certain sub-classes of the class for which it is applicable.

Thus the current state of the art in global optimization methods is a collection of methods which in general solve various specific problem types and require various restrictions on the set of functions. Thus for most methods, *a priori* knowledge is required about the mathematical model to be solved, or such knowledge must be acquired if valid results are to be obtained.

In addition these methods have been developed for problems which are assumed to be feasible and bounded and thus with the exception of a few methods which allow for infeasible problems, the issue of determining the feasibility and/or the boundedness of the problem as well as the provision of causal information has not been addressed. The rationale for centering on feasible and bounded problems has been that the feasibility of the problem can be determined by transforming the problem and solving a problem of equivalent difficulty and that the unboundedness can be determined if a very large negative number is given as a solution to the problem. While the rationale is essentially correct, it is limited in that it requires that the user of a global optimization method must have a certain level of expertise in global optimization theory and practice.

Appendix B

Test Set

B.1 Problems from Existing Test Sets

B.1.1 Problems from Floudas and Pardalos

The following problems are taken from Floudas and Pardalos [26].

Floudas and Pardalos 3.1

$$\begin{array}{llll} \min & x_1 + x_2 + x_3 & & \\ s.t. & & & \\ c_1 & -1 + 0.0025(x_4 + x_6) & \leq & 0 \\ c_2 & -1 + 0.0025(-x_4 + x_5 + x_7) & \leq & 0 \\ c_3 & -1 + 0.01(-x_5 + x_8) & \leq & 0 \\ c_4 & 100x_1 - x_1x_6 + 833.33252x_4 - 83333.333 & \leq & 0 \\ c_5 & x_2x_4 - x_2x_7 - 1250x_4 + 1250x_5 & \leq & 0 \\ c_6 & x_3x_5 - x_3x_8 - 2500x_5 + 1250000 & \leq & 0 \\ & 100 \leq x_1 \leq 10000 & & \\ & 1000 \leq x_i \leq 10000 & \text{for } i = 2, 3 & \\ & 10 \leq x_i \leq 1000 & \text{for } i = 4, \dots, 8 & \end{array}$$

Best Known Solution

$$f(x^*) = 7049.25$$

$$x^* = (579.31, 1359.97, 5109.97, 182.02, 295.6, 217.98, 286.42, 395.60)$$

Floudas and Pardalos 3.2

$$\min \quad 37.293239x_1 + 0.8356891x_1x_5 + 5.3578547x_3^2 - 40792.141$$

s.t.

$$c_1 \quad -0.0022653x_3x_5 + 0.0056858x_2x_5 + 0.0006262x_1x_4 - 6.665593 \leq 0$$

$$c_2 \quad 0.0022053x_3x_5 - 0.0056858x_2x_5 - 0.0006262x_1x_4 - 85.334407 \leq 0$$

$$c_3 \quad 0.0071317x_2x_5 + 0.0021813x_3^2 + 0.0029955x_1x_2 - 29.48751 \leq 0$$

$$c_4 \quad -0.0071317x_2x_5 - 0.0021813x_3^2 - 0.0029955x_1x_2 + 9.48751 \leq 0$$

$$c_5 \quad 0.0047026x_3x_5 + 0.0019085x_3x_4 + 0.0012547x_1x_3 - 15.699039 \leq 0$$

$$c_6 \quad -0.0047026x_3x_5 - 0.0019085x_3x_4 - 0.0012547x_1x_3 + 10.699039 \leq 0$$

$$78 \leq x_1 \leq 102$$

$$33 \leq x_2 \leq 45$$

$$27 \leq x_i \leq 45 \quad \text{for } i = 3, 4, 5$$

Best Known Solution

$$f(x^*) = -30665.5387$$

$$x^* = (78, 33, 29.9953, 45, 36.7758)$$

Floudas and Pardalos 4.6

$$\min \quad -x_1 - x_2$$

s.t.

$$c_1 \quad -2x_1^4 + 8x_1^3 - 8x_1^2 + x_2 - 2 \leq 0$$

$$c_2 \quad -4x_1^4 + 32x_1^3 - 88x_1^2 + 96x_1 + x_2 - 36 \leq 0$$

$$0 \leq x_1 \leq 3$$

$$0 \leq x_2 \leq 4$$

Best Known Solution

$$f(x^*) = -5.5079$$

$$x^* = (2.3295, 3.1783)$$

Floudas and Pardalos 4.7

$$\min \quad -12x_1 - 7x_2 + x_2^2$$

s.t.

$$c_1 \quad -2x_1^4 - x_2 + 2 = 0$$

$$0 \leq x_1 \leq 2$$

$$0 \leq x_2 \leq 3$$

Best Known Solution

$$f(x^*) = -16.73889$$

$$x^* = (0.71751, 1.470)$$

Floudas and Pardalos 4.7 SUB

$$\min \quad 4x_1^8 + 6x_1^4 - 12x_1 - 10$$

$$\text{s.t.} \quad 0 \leq x_1 \leq 1$$

Best Known Solution

$$f(x^*) = -16.73889$$

$$x^* = (0.71751)$$

The above problem is Floudas and Pardalos 4.7 with the substitution of $x_2 = 2 - 2x_1^4$.

Floudas and Pardalos 6.2

$$\max \quad 9x_1 + 15x_2 - 6x_3 - 16x_4 - 10(x_5 + x_6)$$

s.t.

$$\begin{aligned}
c_1 \quad & x_7 + x_8 - x_3 - x_4 = 0 \\
c_2 \quad & x_1 - x_7 - x_5 = 0 \\
c_3 \quad & x_2 - x_8 - x_6 = 0 \\
c_4 \quad & x_9 x_7 + 2x_5 - 2.5x_1 \leq 0 \\
c_5 \quad & x_9 x_8 + 2x_6 - 1.5x_2 \leq 0 \\
c_6 \quad & x_9 x_7 + x_9 x_8 - 3x_3 - x_4 = 0 \\
& x_1 \leq 100 \\
& x_2 \leq 200
\end{aligned}$$

Best Known Solution

$$f(x^*) = 400.0$$

$$x^* = (0.0, 200.0, 0.0, 100.0, 0.0, 100.0, 0.0, 100.0, 1.0)$$

Floudas and Pardalos 11.1 - Modified

$$\min \quad 19380 + 32154x_3 + 13189x_{20} + 2376x_{15} + 5329(x_{37}^2 + 2x_{38}^2)$$

s.t.

$$\begin{aligned}
c_1 \quad & \frac{x_1 x_2}{2x_3} + \frac{x_4(x_2^2 + 8100)}{5\pi x_2^2 x_3} - x_{43} \leq 0 \\
c_2 \quad & \frac{x_1 x_2}{2x_3} + \frac{x_4(x_2^2 + 178850)}{5\pi x_2^2(x_3 + \frac{x_9}{250})} - x_{11} x_{43} \leq 0 \\
c_3 \quad & \frac{x_1 x_2}{2x_3} - \frac{x_4(x_2^2 + 8100)}{3.015x_2^2 x_3} - \frac{x_{12} x_3}{29x_2} (2 - \frac{200x_3}{3x_2}) \leq 0 \\
c_4 \quad & \frac{x_1 x_2}{2x_3} - \frac{x_4(x_2^2 + 178850)}{5\pi x_2^2(x_3 + \frac{x_9}{250})} - \frac{x_{12} x_3}{29x_2} (2 - \frac{200x_3}{3x_2}) \leq 0 \\
c_5 \quad & \frac{x_1 x_2}{2x_3} - \frac{x_4(x_2^2 + 8100)}{3.015x_2^2 x_3} - 0.5x_{13} \leq 0 \\
c_6 \quad & \frac{x_1 x_2}{2x_3} - \frac{x_4(x_2^2 + 178850)}{5\pi x_2^2(x_3 + \frac{x_9}{250})} - 0.5x_{13} \leq 0 \\
c_7 \quad & \frac{x_{14} x_4}{(x_3 + x_{15})(x_{16} + 1.56x_{17} x_{18})} - 0.5x_{13} \leq 0 \\
c_8 \quad & \frac{x_{19} x_4 \frac{25}{32}}{x_2(x_3 + x_{15})} - 0.8x_{43} \leq 0 \\
c_9 \quad & \frac{1}{x_{20}} - 0.8x_{43} \leq 0 \\
c_{10} \quad & \frac{x_{21} x_4 x_7 x_{22}}{x_{23}} - \frac{x_{24} x_4}{x_{25}} - 0.5x_{26} \leq 0 \\
c_{11} \quad & \frac{x_{21} x_4 x_7 x_{27}}{x_{23}} + \frac{x_{24} x_4}{x_{25}} - 0.5x_{26} \leq 0 \\
c_{12} \quad & \frac{3x_{28} x_4}{x_2 x_{29}} - \frac{2x_{30}}{3} \leq 0
\end{aligned}$$

$$\begin{aligned}
c_{13} \quad & \frac{x_{16}}{2} + 0.78x_{17}x_{18} - 46 \leq 0 \\
c_{14} \quad & x_3 - x_{20} - 0.125 \leq 0 \\
c_{15} \quad & x_3 - x_{15} \leq 0 \\
c_{16} \quad & x_3 - \frac{x_2}{5} \leq 0 \\
c_{17} \quad & x_{20} - \frac{x_2}{5} \leq 0 \\
c_{18} \quad & x_{32} - x_{33} \leq 0 \\
c_{19} \quad & x_{32} - x_{23} \leq 0 \\
c_{20} \quad & x_{34} - \frac{x_{12}x_{29}^2}{12.96x_{35}^2} \leq 0 \\
c_{21} \quad & x_{34} - \frac{232.5x_{12}x_3^2x_{18}}{x_{36}^2x_{50}} \leq 0 \\
c_{22} \quad & 2x_3 - x_{36} + 120 = 0 \\
c_{23} \quad & 2x_{20} - x_{35} + 120 = 0 \\
c_{24} \quad & 22176\pi(x_3 + 0.125) + 9580.032(x_{20} + 0.125) - x_{39} = 0 \\
c_{25} \quad & x_{10}(2160000\pi + 452736) - x_{40} = 0 \\
c_{26} \quad & x_{39} + x_{40} - 2x_4 = 0 \\
c_{27} \quad & \left(\frac{x_{41}}{\pi x_3}\right)^8 - (120 + 2x_3)^3 = 0 \\
c_{28} \quad & \frac{x_4}{320} - x_{42} = 0 \\
c_{29} \quad & x_{42} \frac{2211718750}{3x_{12}x_{41}} - x_{31} = 0 \\
c_{30} \quad & (x_{38} + 1.56x_{17}x_{18})x_3 - x_{44} = 0 \\
c_{31} \quad & x_{45} - 8x_{38}^2 = 0 \\
c_{32} \quad & x_{44} + x_{45} - x_{25} = 0 \\
c_{33} \quad & \frac{x_{44}x_3}{2} + x_{45} \frac{x_3 + 4x_{38}}{x_{25}} - x_{22} = 0 \\
c_{34} \quad & x_3 + 8x_{38} - x_{22} - x_{27} = 0 \\
c_{35} \quad & x_{46} - x_{22} + \frac{x_3}{2} = 0 \\
c_{36} \quad & x_{47} - x_3 - 4x_{38} + x_{22} = 0 \\
c_{37} \quad & x_{44}x_{46}^2 + x_{45}x_{47}^2 + \frac{x_{44}x_3^2}{12} + \frac{128}{3}x_{38}^4 - x_{23} = 0 \\
c_{38} \quad & 1.56x_3x_{17}x_{18} - x_{48} = 0 \\
c_{39} \quad & 8x_{37}^2 - x_{49} = 0 \\
c_{40} \quad & x_9 - x_{48} - x_{49} = 0
\end{aligned}$$

$$c_{41} \quad \frac{x_{48}x_3}{2} + x_{49}\frac{x_3+4x_{37}}{x_9} - x_5 = 0$$

$$c_{42} \quad x_3 + 8x_{37} - x_5 - x_6 = 0$$

$$c_{43} \quad x_7 + \frac{x_3}{2} - x_5 = 0$$

$$c_{44} \quad x_8 + x_5 - x_3 - 4x_{37} = 0$$

$$c_{45} \quad x_{48}x_7^2 + x_{49}x_8^2 + \frac{x_{48}x_3^2}{12} + \frac{128}{3} * x_{37}^4 - x_{33} = 0$$

$$c_{46} \quad 221149.44x_3x_{18}\frac{x_3+\frac{x_9}{250}}{x_{36}x_{50}} - x_{32} = 0$$

$$c_{47} \quad x_{18}^2 - x_3 = 0$$

$$c_{48} \quad x_{17}^2 - x_2 = 0$$

$$c_{49} \quad x_{50}^2 - x_{36} = 0$$

$$49 \leq x_1 \leq 51$$

$$59 \leq x_2 \leq 61$$

$$0 \leq x_3 \leq \infty$$

$$0 \leq x_4 \leq \infty$$

$$-\infty \leq x_5 \leq \infty$$

$$-\infty \leq x_6 \leq \infty$$

$$-\infty \leq x_7 \leq \infty$$

$$-\infty \leq x_8 \leq \infty$$

$$0 \leq x_9 \leq \infty$$

$$0.0215 \leq x_{10} \leq 0.0217$$

$$0.84 \leq x_{11} \leq 0.86$$

$$2,890,000 \leq x_{12} \leq 2,901,000$$

$$37,990 \leq x_{13} \leq 38,010$$

$$0.759 \leq x_{14} \leq 0.761$$

$$0.0625 \leq x_{15} \leq 3$$

$$0.879 \leq x_{16} \leq 0.881$$

$$7.681 \leq x_{17} \leq 7.811$$

$$0.25 \leq x_{18} \leq 1.8$$

$$0.318 \leq x_{19} \leq 0.320$$

$$0.0625 \leq x_{20} \leq 3$$

$$0.052 \leq x_{21} \leq 0.054$$

$$0.006 \leq x_{22} \leq 4,857$$

$$0.0007 \leq x_{23} \leq 3,410,000,000$$

$$0.339 \leq x_{24} \leq 0.341$$

$$0.22 \leq x_{25} \leq 144.5$$

$$35,990 \leq x_{26} \leq 36,010$$

$$-4,857 \leq x_{27} \leq 27$$

$$0.203 \leq x_{28} \leq 0.205$$

$$0.4 \leq x_{29} \leq 0.6$$

$$12,690 \leq x_{30} \leq 12,710$$

$$117 \leq x_{31} \leq 13,600$$

$$0.15 \leq x_{32} \leq 3,095$$

$$0.0007 \leq x_{33} \leq 3,440,000,000$$

$$4 \leq x_{34} \leq 6$$

$$120.125 \leq x_{35} \leq 126$$

$$120.125 \leq x_{36} \leq 126$$

$$0.0625 \leq x_{37} \leq 3$$

$$0.0625 \leq x_{38} \leq 3$$

$$14,855 \leq x_{39} \leq 247,650$$

$$155,625 \leq x_{40} \leq 157,080$$

$$1.18 \leq x_{41} \leq 58$$

$$266 \leq x_{42} \leq 633$$

$$17,490 \leq x_{43} \leq 17,510$$

$$0.19 \leq x_{44} \leq 72.5$$

$$0.03125 \leq x_{45} \leq 72$$

$$-1.5 \leq x_{46} \leq 4,857$$

$$-4,857 \leq x_{47} \leq 15$$

$$0.18 \leq x_{48} \leq 63.5$$

$$0.03125 \leq x_{49} \leq 72$$

$$10.95 \leq x_{50} \leq 11.3$$

An alternative set of bounds is as follows:

$49 \leq x_1 \leq 51$	$59 \leq x_2 \leq 61$	$0.0625 \leq x_3 \leq 3$
$85,240 \leq x_4 \leq 202,365$	$0.005 \leq x_5 \leq 5,040$	$-5,038 \leq x_6 \leq 27$
$-1.5 \leq x_7 \leq 5,040$	$-5,038 \leq x_8 \leq 15$	$0.215 \leq x_9 \leq \infty$
$0.0215 \leq x_{10} \leq 0.0217$	$0.84 \leq x_{11} \leq 0.86$	$2,890,000 \leq x_{12} \leq 2,901,000$
$37,990 \leq x_{13} \leq 38,010$	$0.759 \leq x_{14} \leq 0.761$	$0 \leq x_{15} \leq \infty$
$0.879 \leq x_{16} \leq 0.881$	$7.681 \leq x_{17} \leq 7.811$	$0 \leq x_{18} \leq \infty$
$0.318 \leq x_{19} \leq 0.320$	$0 \leq x_{20} \leq \infty$	$0.052 \leq x_{21} \leq 0.054$
$0 \leq x_{22} \leq \infty$	$0 \leq x_{23} \leq \infty$	$0.339 \leq x_{24} \leq 0.341$
$0 \leq x_{25} \leq \infty$	$35,990 \leq x_{26} \leq 36,010$	$-\infty \leq x_{27} \leq \infty$
$0.203 \leq x_{28} \leq 0.205$	$0.4 \leq x_{29} \leq 0.6$	$12,690 \leq x_{30} \leq 12,710$
$0 \leq x_{31} \leq \infty$	$0 \leq x_{32} \leq \infty$	$0 \leq x_{33} \leq \infty$
$4 \leq x_{34} \leq 6$	$120 \leq x_{35} \leq \infty$	$120 \leq x_{36} \leq \infty$
$0 \leq x_{37} \leq \infty$	$0 \leq x_{38} \leq \infty$	$0 \leq x_{39} \leq \infty$
$0 \leq x_{40} \leq \infty$	$0 \leq x_{41} \leq \infty$	$0 \leq x_{42} \leq \infty$
$17,490 \leq x_{43} \leq 17,510$	$0 \leq x_{44} \leq \infty$	$0 \leq x_{45} \leq \infty$
$-\infty \leq x_{46} \leq \infty$	$-\infty \leq x_{47} \leq \infty$	$0 \leq x_{48} \leq \infty$
$0 \leq x_{49} \leq \infty$	$10 \leq x_{50} \leq \infty$	

The above problem has been modified to eliminate the obvious typos in the given problem. In addition, the design variables, $x_3, x_{15}, x_{20}, x_{37}, x_{38}$, have been changed from discrete variables to continuous variables. The given best known solution with $x_3 = 0.1875, x_{15} = 0.3125, x_{20} = 0.3125, x_{37} = 1.125, x_{38} = 0.5625$ is infeasible for the constraint set as presented in Floudas and Pardalos.

B.1.2 Problems from Schwefel

The following problems are taken from Schwefel [108].

Schwefel 2.1 (or Beale)

$$\min (1.5 - x_1 + x_1 x_2)^2 + (2.25 - x_1 + x_1 x_2^2)^2 + (2.625 - x_1 + x_1 x_2^3)^2$$

Best Known Solution

$$\begin{aligned}f(x^*) &= 0 \\x^* &= (3, 0.5)\end{aligned}$$

Schwefel 3.1 - $n = 3$

$$\min \sum_{i=1}^3 [(x_1 - x_i^2)^2 + (x_i - 1)^2]$$

Global Solution

$$\begin{aligned}f(x^*) &= 0.0 \\x^* &= (1.0, 1.0, 1.0)\end{aligned}$$

B.1.3 Problems from other Sources

Six Hump Back

$$\begin{aligned}\min \quad & 4x_1^2 - 2.1x_1^4 + (1/3)x_1^6 + (x_1x_2) - 4x_2^2 + 4x_2^4 \\s.t. \quad & -5 \leq x_i \leq 5 \quad \text{for } i = 1, 2\end{aligned}$$

Best Known Solution

$$\begin{aligned}f(x^*) &= -1.0316285 \\x^* &= (0.08983, -0.7126), (-0.08983, 0.7126)\end{aligned}$$

The above problem is taken from Törn and Žilinskas [115].

Three Hump Camel

$$\begin{aligned}\min \quad & 12x_1^2 - 6.3x_1^4 + x_1^6 + 6x_2(x_2 - x_1) \\s.t. \quad & -3.0 \leq x_i \leq 3.0\end{aligned}$$

Best Known Solution

$$\begin{aligned}f(x^*) &= 0 \\x^* &= (0, 0)\end{aligned}$$

The above problem is taken from Hansen [46]

Zhang No. 1

$$\begin{aligned}
 \min \quad & 4x_1^2 - 2.1x_1^4 + (1/3)x_1^6 + (x_1x_2) - 4x_2^2 + 4x_2^4 \\
 \text{s.t.} \quad & \\
 c_1 \quad & 2\sin(2\pi x_2) - \sin(4\pi x_1) \leq 0 \\
 c_2 \quad & 2x_1 + x_2 \leq 0 \\
 & -1 \leq x_i \leq 1 \quad \text{for } i = 1, 2
 \end{aligned}$$

Best Known Solution

$$\begin{aligned}
 f(x^*) &= -0.911480 \\
 x^* &= (0.11999, -0.58211)
 \end{aligned}$$

The above problem is taken from Zhang and Sha [121]

Spring Design Problem

$$\begin{aligned}
 \min \quad & 0.698x_1x_2x_3^2 \\
 \text{s.t.} \quad & \\
 c_1 \quad & \frac{0.005x_2}{x_3} - 1.0 \leq 0 \\
 c_2 \quad & \frac{0.004x_2}{x_3} - 1.0 \leq 0 \\
 c_3 \quad & \frac{0.060x_1x_2^2}{x_3} - 1.0 \leq 0 \\
 c_4 \quad & \frac{x_1x_4}{4x_2} + \frac{x_3}{2x_2} - 1.0 \leq 0 \\
 c_5 \quad & 0.625x_1x_3 + 1.25x_3 - 1.0 \leq 0 \\
 c_6 \quad & \frac{0.083x_2}{x_3} - 1.0 \leq 0 \\
 c_7 \quad & 1.0 - \frac{6x_3}{x_2} \leq 0 \\
 c_8 \quad & \sin(\pi x_1) = 0 \\
 & 4 \leq x_1 \leq 7 \\
 & 0.5 \leq x_2 \leq 2 \\
 & 0.032 \leq x_3 \leq 0.437 \\
 & 0.3 \leq x_4 \leq 0.6
 \end{aligned}$$

Best Known Solution

$$f(x^*) = 0.025715$$

$$x^* = (4.50000, 0.13572, 0.43214)$$

The above problem is taken from Jha and Mokaddem [60] and is based on the optimal design of a high duty helical compression spring..

B.2 New Test Set

B.2.1 Unbounded Problems

Unbounded 1

$$\min \frac{-1.0}{x_1 - 2.0}$$

Best Known Solution

$$f(x^*) = -\infty$$

$$x^* = (2.0)$$

Unbounded 2

$$\min \frac{-1.0}{(x_1 - 2.0)^2 + (x_2 - 4.0)^2}$$

Best Known Solution

$$f(x^*) = -\infty$$

$$x^* = (2.0, 4.0)$$

Unbounded 3

$$\min \frac{8.0 - 2.0x_1 - x_1^2}{x_1 - 2.0}$$

Best Known Solution

$$f(x^*) = -\infty$$

$$x^* = (\infty)$$

Unbounded 4

$$\min -4.0 - x_1$$

Best Known Solution

$$f(x^*) = -\infty$$

$$x^* = (\infty)$$

Unbounded 5

$$\min \frac{-1}{(x_1-2)*(x_2-3)+(x_2-4)^2}$$

Best Known Solution

$$f(x^*) = -\infty$$

$$\text{when } x_1 = \frac{x_2^2 - 10x_2 + 24}{3 - x_2}$$

Unbounded 6

$$\min \frac{-1}{(x_1-2)*(x_2-3)*(x_2-4)^2}$$

Best Known Solution

$$f(x^*) = -\infty$$

$$\text{when } x_1 = 2, x_2 = 3, \text{ or } x_2 = 4$$

Unbounded 7

$$\min \frac{x_1*x_2-3x_2-4x_1+12}{(x_1-3)^3}$$

Best Known Solution

$$f(x^*) = -\infty$$

$$\text{when } x_1 = 3$$

Unbounded 8

$$\min \frac{3x_1 - 4x_2 + 6x_3 - 2x_4}{(x_1 - 2)^2 + (x_2 - 3)^2 + (x_3 - 1)^2 + (x_4 - 1)^2 + (x_5 - 10)^2}$$

Best Known Solution

$$f(x^*) = -\infty$$

$$x^* = (2, 3, 1, 1, 10)$$

Unbounded 2 cs2

$$\min \frac{-1.0}{(x_1 - 2.0)^2 + (x_2 - 4.0)^2}$$

s.t.

$$c_1 \quad 2 - x_2 + x_1 = 0$$

$$c_2 \quad 3(x_1 - 2)^2 - x_2 + 4 \leq 0$$

$$c_3 \quad (x_1 - 2)^2 + x_2 - 4 \leq 0$$

Best Known Solution

$$f(x^*) = -\infty$$

$$x^* = (2.0, 4.0)$$

The feasible region is the single point (2.0, 4.0).

Unbounded 8 con 21

$$\min \frac{3x_1 - 4x_2 + 6x_3 - 2x_4}{(x_1 - 2)^2 + (x_2 - 3)^2 + (x_3 - 1)^2 + (x_4 - 1)^2 + (x_5 - 10)^2}$$

$$\text{s.t. } c_1 \sin(4\pi x_1) - \sum_{i=2}^5 2 \sin^2(2\pi x_i) = 0$$

Best Known Solution

$$f(x^*) = -\infty$$

$$x^* = (2, 3, 1, 1, 10)$$

B.2.2 Infeasible Problems

Unbounded 2 cs2 mod

$$\begin{array}{ll}
 \min & \frac{-1.0}{(x_1-2.0)^2+(x_2-4.0)^2} \\
 \text{s.t.} & \\
 c_1 & 3(x_1-2)^2 - x_2 + 4 \leq 0 \\
 c_2 & (x_1-2)^2 + x_2 - 4 \leq 0 \\
 c_3 & 2 - x_2 + x_1 = 0 \\
 & x_2 \leq 3.9999 \\
 HS & (c_2, c_3, x_2 \leq 3.9999)
 \end{array}$$

Unbounded 8 cs8

$$\begin{array}{ll}
 \min & \frac{3x_1-4x_2+6x_3-2x_4}{(x_1-2)^2+(x_2-3)^2+(x_3-1)^2+(x_4-1)^2+(x_5-10)^2} \\
 \text{s.t.} & \\
 c_1 & \sum_{i=1}^5 x_i^2 - 1 \leq 0 \\
 c_2 & 25 - \sum_{i=1}^5 x_i^2 \leq 0 \\
 HS & (c_1, c_2)
 \end{array}$$

Floudas and Pardalos 3.2 cs5

$$\begin{array}{ll}
 \min & 37.293239x_1 + 0.8356891x_1x_5 + 5.3578547x_3^2 - 40792.141 \\
 \text{s.t.} & \\
 c_1 & \sum_{i=1}^5 x_i^2 - 1 \leq 0 \\
 c_2 & \sum_{i=1}^4 x_i^2 - x_5 + 1 \leq 0 \\
 c_3 & \sum_{i=1}^5 x_i \leq 0 \\
 c_4 & \sum_{i=1}^5 x_i^2 - 25 \leq 0 \\
 c_5 & 0.5 - x_1^2 - x_5^4 \leq 0 \\
 HS & (c_1, c_2, c_3)
 \end{array}$$

Six Hump Back cs10

$$\begin{aligned}
\min \quad & 4x_1^2 - 2.1x_1^4 + (1/3)x_1^6 + (x_1x_2) - 4x_2^2 + 4x_2^4 \\
s.t. \quad & \\
c_1 \quad & x_2^2 - 8x_1 - 16 \leq 0 \\
c_2 \quad & x_2 - 4\cos(x_1)\sin^2(13x_1) - 0.3 \leq 0 \\
c_3 \quad & 2x_1^4 - 8x_1^3 + 8x_1^2 - x_2 + 2 \leq 0 \\
c_4 \quad & 4x_1^3 + 6x_1^2 - 3x_1 + 2.5 \leq 0 \\
c_5 \quad & 2.0\cos(x_1) - 1.0 - x_2 \leq 0 \\
c_6 \quad & 10 - 5x_1^2 - x_2 \leq 0 \\
HIS \quad & (c_2, c_3, c_6)
\end{aligned}$$

Three Hump Camel cs12

$$\begin{aligned}
\min \quad & 12x_1^2 - 6.3x_1^4 + x_1^6 + 6x_2(x_2 - x_1) \\
s.t. \quad & \\
c_1 \quad & 2x_1^4 - 8x_1^3 + 8x_1^2 - x_2 + 2 \leq 0 \\
c_2 \quad & 4x_1^4 - 32x_1^3 + 88x_1^2 - 96x_1 - x_2 + 36 \leq 0 \\
c_3 \quad & 2x_1^4 + x_2 - 2 \leq 0 \\
c_4 \quad & 3(x_1 - 2)^2 - x_2 + 4 \leq 0 \\
c_5 \quad & 5x_1^2 + x_2 - 10 \leq 0 \\
c_6 \quad & 4x_1^3 + 6x_1^2 - 3x_1 - x_2 + 2.5 \leq 0 \\
c_7 \quad & x_2 - x_1 - 2 \leq 0 \\
c_8 \quad & \cos(x_1) + x_1^2 - x_2 - 6 \leq 0 \\
c_9 \quad & 2 - (x_2 + 2)^2 + x_1^2 \leq 0 \\
c_{10} \quad & -2x_1^4 + 8x_1^3 - 8x_1^2 + x_2 - 4 \leq 0 \\
c_{11} \quad & x_2 - 4\cos(x_1)\sin^2(13x_1) - 0.3 \leq 0 \\
c_{12} \quad & 2\cos(x_1) + 1 - x_2 \leq 0 \\
c_{13} \quad & 16 - x_2^2 + 8x_1 \leq 0
\end{aligned}$$

c_{14}	$4x_1^2 * \cos(50x_1^2) - x_2 \leq 0$
c_{15}	$x_2 - 40(x_1 + 3)^2 + 60 \leq 0$
c_{16}	$4 - (x_1 - 2)^2 - x_2 \leq 0$
c_{17}	$13 \cos(\frac{x_1}{2}) - x_2 \leq 0$
c_{18}	$x_2 + 13 \cos(\frac{x_1}{2}) \leq 0$
HS_1	(c_1, c_5, c_{11})
HS_2	(c_4, c_{18})
HS_3	(c_3, c_4)
	<i>possibly others</i>

B.2.3 Feasible and Bounded Test Set

Six Hump Back cs1

$$\begin{aligned}
 \min \quad & 4x_1^2 - 2.1x_1^4 + (1/3)x_1^6 + (x_1x_2) - 4x_2^2 + 4x_2^4 \\
 \text{s.t.} \quad & \\
 c_1 \quad & -2x_1^4 + 2 - x_2 \leq 0 \\
 c_2 \quad & 2 - (x_2 + 2)^2 + x_1^2 \leq 0 \\
 c_3 \quad & \cos(x_1 + 1) + (x_1 - 1)^2 - x_2 - 6 \leq 0
 \end{aligned}$$

Best Known Solution

$$\begin{aligned}
 f(x^*) &= -1.031628 \\
 x^* &= (-8.984201 * 10^{-2}, 0.7126564)
 \end{aligned}$$

The feasible region consists of two disjoint regions.

Six Hump Back cs11

$$\begin{aligned}
 \min \quad & 4x_1^2 - 2.1x_1^4 + (1/3)x_1^6 + (x_1x_2) - 4x_2^2 + 4x_2^4 \\
 \text{s.t.} \quad & \\
 c_1 \quad & x_2 - 4x_1^2 \cos(50x_1^2) \leq 0
 \end{aligned}$$

$$\begin{aligned}
c_2 \quad & 40(x_1 + 3)^2 - 60 - x_2 \leq 0 \\
c_3 \quad & x_2 - 4 \cos(x_1) \sin^2(1.3x_1) - 0.3 \leq 0
\end{aligned}$$

Best Known Solution

$$\begin{aligned}
f(x^*) &= 2.113815 \\
x^* &= (-1.933623, 0.2999741)
\end{aligned}$$

The feasible region consists of 21 thin and 1 large irregular shaped disjoint regions.

Three Hump Camel cs3

$$\begin{aligned}
\min \quad & 12x_1^2 - 6.3x_1^4 + x_1^6 + 6x_2(x_2 - x_1) \\
s.t. \quad & \\
c_1 \quad & x_2 - 13x_1 \cos(x_1/2) = 0 \\
c_2 \quad & x_2 + 13x_1 \cos(x_1/2) = 0 \\
& 0 \leq x_1 \leq 30
\end{aligned}$$

Best Known Solution

$$\begin{aligned}
f(x^*) &= 4.66147 \\
x^* &= (\pi, 0)
\end{aligned}$$

The feasible region consists of 5 points, equidistant on the line $x_2 = 0$.

Three Hump Camel cs7

$$\begin{aligned}
\min \quad & 12x_1^2 - 6.3x_1^4 + x_1^6 + 6x_2(x_2 - x_1) \\
s.t. \quad & \\
c_1 \quad & -2x_1^4 + 8x_1^3 - 8x_1^2 + x_2 - 4 \leq 0 \\
c_2 \quad & 4x_1^4 - 32x_1^3 + 88x_1^2 - 96x_1 - x_2 + 36 \leq 0 \\
c_3 \quad & 4x_1^3 + 6x_1^2 - 3x_1 + 2.5 \leq 0 \\
& 0 \leq x_1 \leq 10
\end{aligned}$$

Best Known Solution

$$f(x^*) = 5.523270 * 10^6$$

$$x^* = (5.783544, 9.596695 * 10^2)$$

The feasible region consists of two disjoint feasible regions, separated by a distance > 900 .

Unbounded 2 mod

$$\begin{aligned} \min \quad & \frac{-1.0}{(x_1-2.0)^2+(x_2-4.0)^2} \\ \text{s.t.} \quad & x_2 \leq 3.9999 \end{aligned}$$

Best Known Solution

$$f(x^*) = -1.0 * 10^8$$

$$x^* = (2.0, 3.9999)$$

An infeasible unbounded from below point exists within a distance of 0.0001 from the boundary of the feasible region.

Unbounded 2 mod2

$$\begin{aligned} \min \quad & \frac{-1.0}{(x_1-2.0)^2+(x_2-4.0)^2} \\ \text{s.t.} \quad & x_2 \leq 3.999 \end{aligned}$$

Best Known Solution

$$f(x^*) = -1.0 * 10^6$$

$$x^* = (2.0, 3.999)$$

An infeasible unbounded from below point exists within a distance of 0.001 from the boundary of the feasible region.

Melville Shipping Ltd. Problem

$$\begin{aligned} \min \quad & (23,616A_2u^2(1-x_2)^2(1-x_1) + 113,356.8A_1u(1-x_2)(1-x_1)n + \\ & 544,112.64A_0n^2(1-x_1) - 1,043,450x_3u^2(1-x_1))^2 \\ \text{s.t.} \quad & 0 \leq x_1 \leq 0.6 \\ & -0.1 \leq x_2 \leq 0.5 \\ & 0.003 \leq x_3 \leq 0.019 \end{aligned}$$

where

case	u	n	A_0	A_1	A_2
a1	5.5694	2.1667	0.239580	-0.308760	-0.107579
b1	6.6195	2.1667	0.334962	-0.354097	-0.070406
b10	8.0511	2.1667	0.511790	-0.403934	-0.043294

Best Known Solution a1

$$\begin{aligned} f(x^*) &= 1.90582 * 10^{-19} \\ x^* &= (3.66311 * 10^{-3}, -2.16319 * 10^{-2}, 3.02415 * 10^{-3}) \end{aligned}$$

Best Known Solution b1

$$\begin{aligned} f(x^*) &= 1.66018 * 10^{-19} \\ x^* &= (5.94704 * 10^{-1}, 2.050548 * 10^{-1}, 3.00100 * 10^{-3}) \end{aligned}$$

Best Known Solution b10

$$\begin{aligned} f(x^*) &= 5.72594 * 10^{-19} \\ x^* &= (5.14944 * 10^{-1}, 6.047277 * 10^{-3}, 3.21220 * 10^{-3}) \end{aligned}$$

The above problem was obtained from a communication with Mellville Shipping Ltd. of Ottawa, Ontario.

Variation on Rosenbrock

$$\begin{aligned} \min \quad & \sum_{i=1}^5 100 * [(x_{10i} - x_{10i-1}^2)^2 + \sin(x_{5i} - x_{5i-1}^2) + \cos(x_{5i+25} - x_{5i+24}^2)] + \\ & (x_{10i-1} - 1)^2 + \sin(x_{5i-1} - 1) + \cos(x_{5i+24} - 1)) + \\ & \sum_{i=0}^9 \exp^{x_{5i+2} - x_{5i+1} - x_{5i+3}} \\ \text{s.t.} \quad & -10 \leq x_i \leq 10 \text{ for } i = 11 \dots 50 \end{aligned}$$

Best Known Solution

$$f(x^*) = -238.94$$

$$x^* = (-3.3357, -6.6638, -3.3357, -0.60479, -1.2058, \\ 0.0000, 0.0000, 0.0000, 0.53701, -0.16176, \\ 0.0000, 0.0000, 0.0000, -0.40004, -1.4147, \\ 0.0000, 0.0000, 0.0000, 0.51900, -0.18152, \\ 0.0000, 0.0000, 0.0000, -0.56180, -1.2538, \\ 0.0000, 0.0000, 0.0000, 0.98785, 0.97566, \\ 0.0000, 0.0000, 0.0000, -2.1645, 1.5452, \\ 0.0000, 0.0000, 0.0000, 0.98466, 0.96961, \\ 0.0000, 0.0000, 0.0000, -2.1908, 1.6591, \\ 0.0000, 0.0000, 0.0000, 1.0096, 1.0193)$$

Variation on Rosenbrock cs16

$$\min \sum_{i=1}^5 100 * [(x_{10i} - x_{10i-1}^2)^2 + \sin(x_{5i} - x_{5i-1}^2) + \cos(x_{5i+25} - x_{5i+24}^2)] + \\ (x_{10i-1} - 1)^2 + \sin(x_{5i-1} - 1) + \cos(x_{5i+24} - 1)) + \\ \sum_{i=0}^9 \exp^{x_{5i+2} - x_{5i+1} - x_{5i+3}}$$

$$s.t. \quad -10 \leq x_i \leq 10 \text{ for } i = 11 \dots 50$$

$$c_i \quad \sin(4\pi x_{5(i-1)}) - \sum_{j=2}^5 2 \sin^2(2\pi x_{5(i-1)+j}) \leq 0 \text{ for } i = 1 \dots 10$$

$$c_i \quad \sum_{j=1}^4 x_{5(i-11)+j}^2 - x_{5(i-10)} - 10 \leq 0 \text{ for } i = 11 \dots 20$$

$$c_{21} \quad \sum_{i=1}^{10} x_{5i}^2 - 1000 \leq 0$$

$$c_{22} \quad \sum_{i=1}^{10} \sin(x_{5i-1}) \leq 0$$

$$c_{23} \quad \sum_{i=1}^{10} \sin(x_{5i-2}) \leq 0$$

$$c_{24} \quad \sum_{i=1}^{10} \sin(x_{5i-3}) \leq 0$$

$$c_{25} \quad \sum_{i=1}^{10} \sin(x_{5i-4}) \leq 0$$

Best Known Solution

$$f(x^*) = -238.87$$

$$\begin{aligned} x^* = & (-1.0605, -2.3134, -1.2414, -0.59334, 5.06383, \\ & 8.4399 * 10^{-3}, 4.7752 * 10^{-2}, 4.3381 * 10^{-2}, 0.51568, -0.18291, \\ & 1.2212 * 10^{-2}, 4.9720 * 10^{-2}, 4.7153 * 10^{-2}, -0.73707, -1.0253, \\ & 7.4406 * 10^{-3}, 5.3012 * 10^{-2}, 4.2381 * 10^{-2}, 0.56282, -0.13320, \\ & 9.4666 * 10^{-3}, 4.7144 * 10^{-2}, 4.4407 * 10^{-2}, -0.66213, -1.1294, \\ & 8.8465 * 10^{-3}, 4.9318 * 10^{-2}, 4.3787 * 10^{-2}, 0.81366, 0.66387, \\ & 1.1564 * 10^{-2}, 4.8429 * 10^{-2}, 4.6505 * 10^{-2}, -2.0255, 0.96145, \\ & 1.1564 * 10^{-2}, 4.8429 * 10^{-2}, 4.6505 * 10^{-2}, 0.81361, 0.66421, \\ & 9.1841 * 10^{-3}, 5.0827 * 10^{-2}, 4.4125 * 10^{-2}, -2.0206, 0.94213, \\ & 1.0773 * 10^{-2}, 5.3967 * 10^{-2}, 4.5713 * 10^{-2}, 0.81369, 0.66368) \end{aligned}$$

Variation on Rosenbrock cs17

$$\begin{aligned} \min \quad & \sum_{i=1}^5 100 * [(x_{10i} - x_{10i-1}^2)^2 + \sin(x_{5i} - x_{5i-1}^2) + \cos(x_{5i+25} - x_{5i+24}^2)] + \\ & (x_{10i-1} - 1)^2 + \sin(x_{5i-1} - 1) + \cos(x_{5i+24} - 1)) + \\ & \sum_{i=0}^9 \exp^{x_{5i+2} - x_{5i+1} - x_{5i+3}} \\ s.t. \quad & -10 \leq x_i \leq 10 \text{ for } i = 11 \dots 50 \\ c_i \quad & \sin(4\pi x_{5(i-1)}) - \sum_{j=2}^5 2 \sin^2(2\pi x_{5(i-1)+j}) \leq 0 \text{ for } i = 1 \dots 10 \\ c_i \quad & \sum_{j=1}^4 x_{5(i-11)+j}^2 - x_{5(i-10)} - 10 \leq 0 \text{ for } i = 11 \dots 20 \\ c_{21} \quad & \sum_{i=1}^{10} x_{5i}^2 - 1000 \leq 0 \\ c_{22} \quad & \sum_{i=1}^{10} \sin(x_{5i-1}) \leq 0 \\ c_{23} \quad & \sum_{i=1}^{10} \sin(x_{5i-2}) \leq 0 \\ c_{24} \quad & \sum_{i=1}^{10} \sin(x_{5i-3}) \leq 0 \\ c_{25} \quad & \sum_{i=1}^{10} \sin(x_{5i-4}) \leq 0 \\ c_i \quad & -4x_{2(i-26)+1}^4 + 32x_{2(i-26)+1}^3 - 88x_{2(i-26)+1}^2 + \\ & 96x_{2(i-26)+1} + x_{2(i-25)} - 36 \leq 0 \text{ for } i = 26 \dots 50 \end{aligned}$$

Best Known Solution

$$f(x^*) = -238.87$$

$$x^* = (-1.0605, -2.3134, -1.2414, -0.59334, 5.06383, \\ 8.4399 * 10^{-3}, 4.7752 * 10^{-2}, 4.3381 * 10^{-2}, 0.51568, -0.18291, \\ 1.2212 * 10^{-2}, 4.9720 * 10^{-2}, 4.7153 * 10^{-2}, -0.73707, -1.0253, \\ 7.4406 * 10^{-3}, 5.3012 * 10^{-2}, 4.2381 * 10^{-2}, 0.56282, -0.13320, \\ 9.4666 * 10^{-3}, 4.7144 * 10^{-2}, 4.4407 * 10^{-2}, -0.66213, -1.1294, \\ 8.8465 * 10^{-3}, 4.9318 * 10^{-2}, 4.3787 * 10^{-2}, 0.81366, 0.66387, \\ 1.1564 * 10^{-2}, 4.8429 * 10^{-2}, 4.6505 * 10^{-2}, -2.0255, 0.96145, \\ 1.1564 * 10^{-2}, 4.8429 * 10^{-2}, 4.6505 * 10^{-2}, 0.81361, 0.66421, \\ 9.1841 * 10^{-3}, 5.0827 * 10^{-2}, 4.4125 * 10^{-2}, -2.0206, 0.94213, \\ 1.0773 * 10^{-2}, 5.3967 * 10^{-2}, 4.5713 * 10^{-2}, 0.81369, 0.66368)$$

Example No. 1

$$\begin{aligned} \min \quad & 2 - 29x + 16x^2 \\ \text{s.t.} \quad & -1.5 - 36x + 36x^2 \leq 0 \\ & -11 + 45.5x - 82x^2 \leq 0 \\ & 0 \leq x \leq 0.25 \end{aligned}$$

Best Known Solution

$$\begin{aligned} f(x^*) &= -4.25 \\ x^* &= (0.25) \end{aligned}$$

B.2.4 Constraint Sets

The following constraint sets have been developed. For the inconsistent constraint sets at least one known IIS is given.

Constraint Set 1

$$c_1 \quad -2x_1^4 + 2 - x_2 \leq 0$$

$$c_2 \quad 2 - (x_2 + 2)^2 + x_1^2 \leq 0$$

$$c_3 \quad \cos(x_1 + 1) + (x_1 - 1)^2 - x_2 - 6 \leq 0$$

The feasible region consists of two disjoint regions.

Constraint Set 2

$$c_1 \quad 3(x_1 - 2)^2 - x_2 + 4 \leq 0$$

$$c_2 \quad (x_1 - 2)^2 + x_2 - 4 \leq 0$$

$$c_3 \quad 2 - x_2 + x_1 = 0$$

The feasible region consists of the single point (2, 4).

Constraint Set 3

$$c_1 \quad x_2 - 13x_1 \cos(x_1/2) = 0$$

$$c_2 \quad x_2 + 13x_1 \cos(x_1/2) = 0$$

$$0 \leq x_1 \leq 30$$

The feasible region consists of 5 points, equidistant on the line $x_2 = 0$.

Constraint Set 4

$$c_1 \quad \sum_{i=1}^5 x_i^2 - 1 \leq 0$$

$$c_2 \quad \sum_{i=1}^5 x_i = 0$$

$$c_3 \quad 25 - \sum_{i=1}^5 x_i^2 \leq 0$$

$$c_4 \quad x_1^2 + x_5^4 - 0.5 \leq 0$$

$$HS \quad (c_1, c_3)$$

Constraint Set 5

$$c_1 \quad \sum_{i=1}^5 x_i^2 - 1 \leq 0$$

$$c_2 \quad \sum_{i=1}^4 x_i^2 - x_5 + 1 \leq 0$$

$$c_3 \quad \sum_{i=1}^5 x_i \leq 0$$

$$c_4 \quad \sum_{i=1}^5 x_i^2 - 25 \leq 0$$

$$c_5 \quad 0.5 - x_1^2 - x_5^4 \leq 0$$

$$HS \quad (c_1, c_2, c_3)$$

Constraint Set 6

$$c_1 \quad -2x_1^4 + 8x_1^3 - 8x_1^2 + x_2 - 2 \leq 0$$

$$c_2 \quad 4x_1^4 - 32x_1^3 + 88x_1^2 - 96x_1 - x_2 + 36 \leq 0$$

$$c_3 \quad 5x_1^2 + x_2 - 10 \leq 0$$

$$c_4 \quad 4x_1^3 + 6x_1^2 - 3x_1 + 2.5 \leq 0$$

$$0 \leq x_1 \leq 3$$

$$-10 \leq x_2 \leq 20$$

$$HS \quad (c_1, c_2, c_4, x_2 \leq 20)$$

Constraint Set 7

$$c_1 \quad -2x_1^4 + 8x_1^3 - 8x_1^2 + x_2 - 4 \leq 0$$

$$c_2 \quad 4x_1^4 - 32x_1^3 + 88x_1^2 - 96x_1 - x_2 + 36 \leq 0$$

$$c_3 \quad 4x_1^3 + 6x_1^2 - 3x_1 + 2.5 \leq 0$$

$$0 \leq x_1 \leq 10$$

The feasible region consists of two disjoint feasible regions, separated by a distance > 900 .

Constraint Set 8

$$c_1 \quad \sum_{i=1}^5 x_i^2 - 1 \leq 0$$

$$c_2 \quad 25 - \sum_{i=1}^5 x_i^2 \leq 0$$

$$HS \quad (c_1, c_2)$$

Constraint Set 9

$$c_1 \quad 2x_1^4 + x_2 - 2 \leq 0$$

$$c_2 \quad -4x_1^3 - 6x_1^2 + 3x_1 - 2.5 \leq 0$$

$$c_3 \quad x_2 - 40(x_1 + 3)^2 + 60 \leq 0$$

$$c_4 \quad -5x_1^2 - x_2 + 10 \leq 0$$

$$c_5 \quad -4x_1^4 + 32x_1^3 - 88x_1^2 + 96x_1 + x_2 - 36 \leq 0$$

$$c_6 \quad 2\cos(x_1) + 1 - x_2 \leq 0$$

$$HS \quad (c_1, c_6)$$

Constraint Set 10

$$c_1 \quad x_2^2 - 8x_1 - 16 \leq 0$$

$$c_2 \quad x_2 - 4\cos(x_1)\sin^2(13x_1) - 0.3 \leq 0$$

$$c_3 \quad 2x_1^4 - 8x_1^3 + 8x_1^2 - x_2 + 2 \leq 0$$

$$c_4 \quad 4x_1^3 + 6x_1^2 - 3x_1 + 2.5 \leq 0$$

$$c_5 \quad 2.0\cos(x_1) - 1.0 - x_2 \leq 0$$

$$c_6 \quad 10 - 5x_1^2 - x_2 \leq 0$$

$$HS \quad (c_2, c_3, c_6)$$

Constraint Set 11

$$c_1 \quad x_2 - 4x_1^2\cos(50x_1^2) \leq 0$$

$$c_2 \quad 40(x_1 + 3)^2 - 60 - x_2 \leq 0$$

$$c_3 \quad x_2 - 4\cos(x_1)\sin^2(13x_1) - 0.3 \leq 0$$

The feasible region consists of 21 thin and 1 large irregular shaped disjoint regions.

Constraint Set 12

c_1	$2x_1^4 - 8x_1^3 + 8x_1^2 - x_2 + 2 \leq 0$
c_2	$4x_1^4 - 32x_1^3 + 88x_1^2 - 96x_1 - x_2 + 36 \leq 0$
c_3	$2x_1^4 + x_2 - 2 \leq 0$
c_4	$3(x_1 - 2)^2 - x_2 + 4 \leq 0$
c_5	$5x_1^2 + x_2 - 10 \leq 0$
c_6	$4x_1^3 + 6x_1^2 - 3x_1 - x_2 + 2.5 \leq 0$
c_7	$x_2 - x_1 - 2 \leq 0$
c_8	$\cos(x_1) + x_1^2 - x_2 - 6 \leq 0$
c_9	$2 - (x_2 + 2)^2 + x_1^2 \leq 0$
c_{10}	$-2x_1^4 + 8x_1^3 - 8x_1^2 + x_2 - 4 \leq 0$
c_{11}	$x_2 - 4\cos(x_1)\sin^2(13x_1) - 0.3 \leq 0$
c_{12}	$2\cos(x_1) + 1 - x_2 \leq 0$
c_{13}	$16 - x_2^2 + 8x_1 \leq 0$
c_{14}	$4x_1^2 * \cos(50x_1^2) - x_2 \leq 0$
c_{15}	$x_2 - 40(x_1 + 3)^2 + 60 \leq 0$
c_{16}	$4 - (x_1 - 2)^2 - x_2 \leq 0$
c_{17}	$13\cos(\frac{x_1}{2}) - x_2 \leq 0$
c_{18}	$x_2 + 13\cos(\frac{x_1}{2}) \leq 0$
HS_1	(c_1, c_5, c_{11})
HS_2	(c_4, c_{18})
HS_3	(c_3, c_4)

*possibly others***Constraint Set 13**

c_1	$-2x_1^4 + 8x_1^3 - 8x_1^2 + x_2 - 2 = 0$
c_2	$4x_1^3 + 6x_1^2 - 3x_1 - x_2 + 2.5 = 0$

$$c_3 \quad 4 \cos(x_1) \sin^2(13x_1) + 0.3 - x_2 = 0$$

$$HIS \quad (c_1, c_2, c_3)$$

Constraint Set 14

$$c_1 \quad 16 - x_2^2 + 8x_1 \leq 0$$

$$c_2 \quad 2x_1^4 + x_2 - 2 \leq 0$$

$$c_3 \quad 60 - 40(x_1 + 3)^2 + x_2 \leq 0$$

$$c_4 \quad 13 \cos\left(\frac{x_1}{2}\right) - x_2 \leq 0$$

$$HIS \quad (c_1, c_3)$$

Constraint Set 15

$$c_1 \quad 100x_1 - x_1x_6 + 833.33252x_4 - 83333.333 = 0$$

$$c_2 \quad 0.0022053x_3x_5 - 0.0056858x_2x_5 - 0.0006262x_1x_4 - 85.334407 \leq 0$$

$$c_3 \quad -1 + 0.0025(x_4 + x_6) = 0$$

$$c_4 \quad 0.5 - x_1^2 - x_5^4 \leq 0$$

$$c_5 \quad \sum_{i=1}^5 x_i^2 - 1 \leq 0$$

$$HIS \quad (c_1, c_3, c_5)$$

Constraint Set 16

$$c_i \quad \sin(4\pi x_{5(i-1)}) - \sum_{j=2}^5 2 \sin^2(2\pi x_{5(i-1)+j}) \leq 0 \quad \text{for } i = 1 \dots 10$$

$$c_i \quad \sum_{j=1}^4 x_{5(i-11)+j}^2 - x_{5(i-10)} - 10 \leq 0 \quad \text{for } i = 11 \dots 20$$

$$c_{21} \quad \sum_{i=1}^{10} x_{5i}^2 - 1000 \leq 0$$

$$c_{22} \quad \sum_{i=1}^{10} \sin(x_{5i-1}) \leq 0$$

$$c_{23} \quad \sum_{i=1}^{10} \sin(x_{5i-2}) \leq 0$$

$$c_{24} \quad \sum_{i=1}^{10} \sin(x_{5i-3}) \leq 0$$

$$c_{25} \quad \sum_{i=1}^{10} \sin(x_{5i-4}) \leq 0$$

Constraint Set 17

$$c_i \quad \sin(4\pi x_{5(i-1)}) - \sum_{j=2}^5 2\sin^2(2\pi x_{5(i-1)+j}) \leq 0 \quad \text{for } i = 1 \dots 10$$

$$c_i \quad \sum_{j=1}^4 x_{5(i-1)+j}^2 - x_{5(i-10)} - 10 \leq 0 \quad \text{for } i = 11 \dots 20$$

$$c_{21} \quad \sum_{i=1}^{10} x_{5i}^2 - 1000 \leq 0$$

$$c_{22} \quad \sum_{i=1}^{10} \sin(x_{5i-1}) \leq 0$$

$$c_{23} \quad \sum_{i=1}^{10} \sin(x_{5i-2}) \leq 0$$

$$c_{24} \quad \sum_{i=1}^{10} \sin(x_{5i-3}) \leq 0$$

$$c_{25} \quad \sum_{i=1}^{10} \sin(x_{5i-4}) \leq 0$$

$$c_i \quad -4x_{2(i-26)+1}^4 + 32x_{2(i-26)+1}^3 - 88x_{2(i-26)+1}^2 + 96x_{2(i-26)+1} + x_{2(i-25)} - 36 \leq 0 \quad \text{for } i = 26 \dots 50$$

Appendix C

GOS-BB : An Algorithm for a Global Optimization System

STEP 1 : INITIALIZE

- 1.1 · Choose a stopping condition.
- 1.2 · Determine a bounding function strategy, *BFS*.
- 1.3 · Determine a branching (subdivision) strategy, *BSS*.
- 1.4 · Determine an initial inclusion, the Initial Box B , of the feasible region and initialize the Set of Boxes, $SB = \{B\}$.
- 1.5 · Initialize the Incumbent Solution (default: $+\infty$).
- 1.6 · Initialize the Lower Bound to $-\infty$.
- 1.7 · Initialize program status:
 - 1.7.1 · Set problem feasibility status (default: unknown).
 - 1.7.2 · Set problem boundedness status (default: unknown).
- 1.8 · Initialize the set P_{inf} (default: $\emptyset$).

STEP 2 : TEST INITIAL BOX

2.1 · Test B for feasibility.

2.2 · IF B is feasible THEN

2.2.1 · Check the lower point, the upper point and the mid-point of B as candidate incumbent solutions.

ELSE IF B is infeasible THEN

2.2.2 · Change problem feasibility status to infeasible.

2.2.3 · Apply the deletion filter IIS_{Θ} detection algorithm to the set
 $FC_{inf} = P_{inf} \cup VB$.

2.2.4 · STOP.

2.3 · Apply BFS over B .

2.4 · IF B is bounded THEN

2.4.1 · Change problem boundedness status to bounded.

2.4.2 · Update Lower Bound = lower bound for B .

ELSE IF B is unbounded THEN

2.4.3 · Change problem boundedness status to unbounded.

2.4.4 · STOP.

2.5 · IF stopping condition is true THEN

2.5.1 · goto 6.1.2.

2.6 · OPTIONAL · Seek an improved incumbent solution.

2.7 · IF an improved incumbent solution has been found in 2.2.1 or 2.6 THEN

2.7.1 · Change problem feasibility status to feasible.

2.7.2 · Update Incumbent Solution.

STEP 3 : BRANCH

3.0 · Initialize Set of New Boxes, SNB , to $\emptyset$.

3.1 · Create New Boxes:

3.1.1 · From BSS , determine the set of boxes to Subdivide, SSB .3.1.2 · For each box, $B \in SSB$:3.1.2.1 · Subdivide B and enter each New Box into SNB .3.1.2.2 · Delete B from SSB and SB .3.2 · Evaluation of New Boxes: For each $NB \in SNB$:3.2.1 · Apply BFS over NB .3.2.2 · IF NB is unbounded THEN

3.2.2.1 · Change problem boundedness status to unbounded.

3.2.2.3 · STOP

ELSE IF the lower bound for $NB >$ Incumbent Solution THEN

3.2.2.3 · Goto 3.2

3.2.3 · IF the parent of NB is of unknown feasibility THEN3.2.3.1 · Test NB for feasibility.3.2.3.2 · IF NB is infeasible THEN

3.2.3.2.1 · Goto 3.2

ELSE IF NB is feasible AND the problem is of unknown feasibility
THEN

3.2.3.2.2 · Change problem feasibility status to feasible.

3.2.4 · Enter NB into SB .3.2.5 · If NB is feasible THEN

3.2.5.1 · Check the lower point, the upper point and the mid-point of NB as candidate incumbent solutions.

3.2.6 · OPTIONAL · Seek an improved incumbent solution

3.2.7 · IF an improved incumbent solution is found in 3.2.5.1 OR 3.2.6 THEN

3.2.7.1 · Update Incumbent Solution

3.2.7.2 · IF problem feasibility status is unknown THEN

3.2.6.2.1 · Change problem feasibility status to feasible

STEP 4 : FATHOM

4.1 - OPTIONAL · Seek an improved incumbent solution

4.2 · IF an improved incumbent solution has been found in 3.2.5.1 OR 3.2.6 OR 4.1 THEN

4.2.1 · Remove all boxes in SB whose lower bound $>$ Incumbent Solution.

STEP 5 : UPDATE

5.1 · IF $SB = \emptyset$ THEN

5.1.1 · Change problem feasibility status to infeasible.

5.1.2 · Apply the deletion filter $IIS_{\emptyset}$ detection algorithm to the set
 $FC_{inf} = P_{inf} \cup VB.$

5.1.3 · STOP.

ELSE

5.1.4 · Lower Bound = least lower bound of the boxes in SB .

5.1.5 · If Lower Bound is $> lbN_{\emptyset}$ and problem boundedness status is unknown
 THEN

5.1.5.1 · Change problem boundedness status to bounded

6: CHECK STOPPING CONDITION

6.1 · IF stopping condition is false THEN

6.1.1 · goto 3.

ELSE

6.1.2 · Output the following:

- * Incumbent Solution.
- * Lower Bound.
- * Set of Boxes.

6.1.3 - OPTIONAL · Determine (and output) flatness of the objective function, and/or other formulation aids.

6.1.4 · STOP

C.1 Procedures for GOS-BB**C.1.1 Determine Feasibility of a Box, B**

0 · Initialize B 's feasibility status as feasible.

1 · For each constraint, $C_i \in P_{inf}$:

1.1 · Determine the feasibility of B relative to C_i .

1.2 · IF B is infeasible THEN

1.2.1 · Set B 's feasibility status to infeasible.

1.2.2 · STOP.

ELSE IF B is of unknown feasibility THEN

1.2.3 · Set B 's feasibility status to unknown.

2 · For each constraint, $C_i \notin P_{inf}$:

2.1 · Determine the feasibility of B relative to C_i .

2.2 · IF B is infeasible THEN

2.2.1 · Set B 's feasibility status to infeasible.

2.2.2 · Add C_i to P_{inf} .

2.2.3 · STOP.

ELSE IF B is of unknown feasibility THEN

2.2.4 · Set B 's feasibility status to unknown.

Determine Feasibility of a Box, B , Relative to a Constraint, C_i

- For a constraint of the form $c_i(x) \leq 0$:
 - The box is feasible if $C_i(B) \leq \delta_{equality}$.
 - The box is infeasible if $C_i(B) > \delta_{equality}$.
 - Otherwise the box is of unknown feasibility.
- For a constraint of the form $c_i(x) = 0$:
 - The box is feasible if $-\delta_{equality} \leq C_i(B) \leq \delta_{equality}$.
 - The box is infeasible if $C_i(B) < -\delta_{equality}$ or $C_i(B) > \delta_{equality}$.
 - Otherwise the box is of unknown feasibility.

C.1.2 Application of a Bounding Function Strategy, BFS , over A Box, B

1 · FOR each $BF_i \in BFS$:

1.1 · Determine the bound, $BF_i(B)$.

2 · IF an unbounded domain point is not found in 1.1 THEN

2.1 · Determine the final bound, $BF(B) = \cap_i BF_i(B)$.

2.2 · Set the lower bound for the box, $B = lb(BF(B))$.

2.3 · IF $lb(BF(B)) > lbN_{\Theta}$ THEN

2.3.1 · Set B 's boundedness status to bounded.

ELSE

2.3.2 · Set B 's boundedness status to unknown.

ELSE

2.4 · Set B 's boundedness status to unbounded.

Determine the Bound, $BF_i(B)$

Apply the bounding function, BF_i to the box B with the following modifications:

1 · IF a division by a box A such that $-\delta_{unbounded} \leq Q(A) \leq \delta_{unbounded}$ is to occur THEN

1.1 · Determine A 's feasibility.

1.2 · IF A is feasible THEN

1.2.1 · Call the unbounded point detection algorithm
with arguments A and $m(A)$.

1.2.2 · IF an unbounded domain point is found THEN

1.2.2.1 · Set B 's boundedness to unbounded.

ELSE

1.2.2.2 · goto 3.

ELSE

1.3 · Determine if A contains a feasible point, a .

1.4 · IF there exists an $a \in A$ such that a is feasible THEN

1.4.1 · Call the unbounded point detection algorithm
with arguments A and a .

1.4.2 · IF an unbounded domain point is found THEN

1.4.2.1 · Set B 's boundedness to unbounded.

ELSE

1.4.2.2 · goto 3.

2 · IF a division by a point a such that $-\delta_{unbounded} \leq Q(a) \leq \delta_{unbounded}$ is to occur THEN

2.1 · Determine a 's feasibility.

2.2 · IF a is feasible THEN

2.2.1 · Determine $\triangleleft(a_i)$ and $\triangleright(a_i)$, the next smallest and largest representable point from a_i , the i^{th} component of the point a .

2.2.2 · Use the $\triangleleft(a_i)$'s and $\triangleright(a_i)$'s to generate a box,
 $A = [(\triangleleft(a_1), \dots, \triangleleft(a_n), (\triangleright(a_1), \dots, \triangleright(a_n))]$.

2.2.3 · Call the unbounded point detection algorithm
 with arguments A and a .

2.2.4 · IF an unbounded domain point is found THEN

2.2.4.1 · Set B 's boundedness to unbounded.

ELSE

2.2.4.2 · goto 3.

3 · Indicate an arithmetic exception has occurred, division by zero and apply the appropriate error handler to maintain continuation of 'normal' GOS-BB operation by providing a correct bound.

C.1.3 Deletion Filter IISU₀ Algorithm

0 · Replace f with the following 'Phase I' objective function (i.e. the sum of the constraint violations):

$$f_{Phase I} = \sum_i \max(g_i(x), 0) + \sum_j |h_j(x)|$$

where $g_i(x)$ and $h_j(x)$ are the constraint functions of the constraints (of the appropriate type) of FC_{inf} .

1 · For each constraint $C_i \in FC_{inf}$ DO:

- 1.1 · Remove the constraint from FC_{inf} . Apply a GOS-BB to the new problem, stopping when a feasible point has been detected or the set of constraints has been shown to be infeasible. If the result is an infeasible set, label this set FC_{inf_i} .
- 1.2 · IF a feasible point is found THEN
 - 1.2.1 · Return the constraint C_i to FC_{inf} .
 - ELSE IF FC_{inf_i} is a proper sub-set of FC_{inf} THEN
 - 1.2.3 · Replace FC_{inf} by FC_{inf_i} .

C.1.4 Unbounded Point Detection Algorithm

- 1 · Use A and x_{ubp} , the input parameters to determine if $V : 1$ or $V : 2$ are true.
- 2 · IF either $V : 1$ or $V : 2$ are true THEN
 - 2.1 · x_{ubp} is an unbounded domain point.
 - 2.2 · Output x_{ubp} as an unbounded domain point.

Appendix D

Interval Branch and Bound Line Search Method

A method is presented in this section which uses interval bounding functions within a branch and bound framework in order to determine the minimum feasible point of a multivariate real valued function, $f(x)$ for $x \in R^n$, over a line segment, L_s , subject to a set of constraints, C .

The basic concept is to parameterize the line segment in R^n in terms of a variable α with $0 \leq \alpha \leq \alpha_{max}$. Each variable of $f(x)$ can be solved for in terms of α resulting in an expression for $f(x)$ in terms of α .

Thus the problem has been reduced to finding a minimum value for a real valued function, $f(\alpha) : R \rightarrow R$, over $\alpha \in \{0 \leq \alpha \leq \alpha_{max}\}$. This problem can then be solved by using an interval branch and bound global optimization method, modified to allow for the possibility of infeasible and/or unbounded domain points within sub-regions of the line segment, L_s .

Once the parameterization in terms of α has been carried out and the appropriate substitutions have been made, the expression should be 'simplified' in terms of the number of occurrences of α , in order to allow for a more accurate result for the interval bounding function employed. This can possibly be carried out by the use of symbolic manipulation, etc.

The basic algorithm is as follows:

Step 0 Determine the parameterization of the variables, x_i , in terms of α using the endpoints of the line segment, L_s .

- Step 1 For each x_i : substitute the appropriate expression for each occurrence of x_i in the objective function and in the functional component of each constraint.
- Step 2 Simplify the resulting functional expressions for the objective function and the functional components of the constraints.
- Step 3 Apply an interval branch and bound global optimization method to the resulting one dimensional problem.

D.1 Example

For example : Consider the application of the above method within a local optimization method at a point, x_a , within a Box, B , for the following problem.

$$\begin{aligned}
 \min f(x) &= (x_1^2 - 2x_2 + 3x_3) \\
 \text{subject to :} \\
 3x_1x_3 - 3x_2 &\leq 0 \\
 -4x_1^2 - x_2 + 3x_3x_2 &\leq 0 \\
 1 &\leq x_1 \leq 3 \\
 1 &\leq x_2 \leq 4 \\
 0 &\leq x_3 \leq 2
 \end{aligned}$$

Consider the box, $B = ([1, 3], [1, 4], [0, 2])$ and a point, $x_a = (2, 2.5, 1) \in B$ with $f(x_a) = 2$. We choose a direction, D , as the negative gradient of $f(x)$ evaluated at x_a . Thus, $D = -(2x_1, -2, 3) = (-2x_1, 2, -3) = (-4, 2, -3)$.

We now compute the intersection of the line from x_a along the direction D to a point x_b on the boundary of B , using the equation :

$$X_{b_i} = x_{a_i} + \alpha_i D_i$$

Where

$$X_{B_i} = \begin{cases} X_{ub_i} & \text{if } D_i > 0 \\ X_{lb_i} & \text{if } D_i < 0 \end{cases}$$

Thus we get the following equations :

$$\begin{aligned}X_{lb_1} &= x_{a_1} - 4\alpha_1 = 1 = 2 - 4\alpha_1 \\X_{ub_2} &= x_{a_2} + 2\alpha_2 = 4 = 2.5 + 2\alpha_2 \\X_{lb_3} &= x_{a_3} - 3\alpha_3 = 0 = 1 - 3\alpha_3\end{aligned}$$

which resolves as $\alpha_1 = 1/4$, $\alpha_2 = 3/4$ and $\alpha_3 = 1/3$. Thus the smallest positive α is $1/4$. Thus $0 \leq \alpha \leq 1/4$.

Thus $x_b = \{2 - 1, 2.5 + .5, 1 - .75\} = \{1, 3, 0.25\}$ and we have the following parameterizations:

$$\begin{aligned}x_1 &= 2 - 4\alpha \\x_2 &= 2.5 + 2\alpha \\x_3 &= 1 - 3\alpha \\0 &\leq \alpha \leq 1/4\end{aligned}$$

We substitute the parameterizations into the problem to get :

$$\begin{aligned}\min f(x) &= (x_1^2 - 2x_2 + 3x_3) = \\(2 - 4\alpha)^2 - 2(2.5 + 2\alpha) + 3(1 - 3\alpha) &= \\2 - 29\alpha + 16\alpha^2 & \\ \text{subject to :} & \\3x_1x_3 - 3x_2 &\leq 0 \Rightarrow \\3(2 - 4\alpha)(1 - 3\alpha) - 3(2.5 + 2\alpha) &= -1.5 - 36\alpha + 36\alpha^2 \leq 0 \\-4x_1^2 - x_2 + 3x_3x_2 &\leq 0 \Rightarrow \\-4(2 - 4\alpha)^2 - (2.5 + 2\alpha) + 3(1 - 3\alpha)(2.5 + 2\alpha) &= \\-11 + 45.5\alpha - 82\alpha^2 &\leq 0 \\0 &\leq \alpha \leq 0.25\end{aligned}$$

or

$$\min f(\alpha) = 2 - 29\alpha + 16\alpha^2$$

subject to :

$$-1.5 - 36\alpha + 36\alpha^2 \leq 0$$

$$-11 + 45.5\alpha - 82\alpha^2 \leq 0$$

$$0 \leq \alpha \leq 0.25$$

which when the above problem (problem 6.3.1-1 of Appendix B.2.3) is solved by GOPS (see Section 9.1), results in $\alpha = 0.25$ and a corresponding $f(\alpha) = -4.25$ (at an accuracy of $1.0\text{e-}5$, in 15 iterations) which corresponds to the point $(x_1, x_2, x_3) = (1, 3, 0.25) \in B$.

D.2 Discussion

The interval branch and bound line search algorithm (IBBLSA) is useful in that it can be applied within the presence of constraints. As such it can be used by descent methods in constrained problems. Unlike most line search methods, which are strictly steepest descent based methods, the above method will not stop at the first local minimum, but will determine a global minimum. This means that its use within a local optimization method will result in a method which will find a local minimum, but the local minimum that is found may be different than the local minimum of the region of attraction in which the initial point resides. Thus it is not a steepest descent method.

One possible use of such a method is with the application of a stochastic method within a branch and bound algorithm. The basic algorithm being:

Step 0 Create a random set of points to cover the sub-region.

Step 1 For each feasible point apply the IBBLSA using the value of the derivative or a finite difference method to determine a direction.

Step 2 For each infeasible point apply the IBBLSA using a feasibility measure to determine a direction.

Step 3 If a stopping condition is met, STOP else goto Step 0.

For an objective function, f , a set of inequality constraints, $g_i(x)$, a set of equality constraints $h_j(x)$ and a box, B consider the following function:

$$d(x) = \sum_i \max(g_i(x), 0) + \sum_j |h_j(x)|$$

This function is a measure of how 'infeasible' a point x is. It is zero when x is feasible and > 0 when x is infeasible. The derivative for this function, or a similar value generated by a finite difference method can be used to generate a direction 'towards' a feasible point. Thus this type of function could possibly be used to determine a feasibility measure for the above algorithm.

Bibliography

- [1] A.V. Aho, J.E. Hopcraft, and J.D. Ullman. *The Design and Analysis of Computer Algorithms*. Addison-Wesley Series in Computer Science and Information Processing. Addison-Wesley Publishing Company, Reading, 1974.
- [2] G. Alefeld. Existence of Solutions and Iterations for Nonlinear Equations. In R. E. Moore, editor, *Reliability in Computing : The Role of Interval Methods in Scientific Computing*, Vol. 19 of *Perspectives in Computing*, pgs. 207-227. Academic Press, New York, 1988.
- [3] G. Alefeld and J. Herzberger. *Introduction to Interval Computations*. Academic Press, New York, 1983.
- [4] F. Aluffi-Pentini, V. Parisi, and F. Zirilli. Global Optimization and Stochastic Differential Equations. *Journal of Optimization Theory and Applications*, 47(1):1-16, September 1985.
- [5] N. Baba. Convergence of a Random Optimization Method for Constrained Optimization Problems. *Journal of Optimization Theory and Applications*, 33:451-461, 1981.
- [6] P. G. Bao and J. G. Rokne. Low Complexity k-Dimensional Taylor Forms. *Applied Mathematics and Computation*, 27(3):265-280, 1988.
- [7] J. L. Bentley, B. W. Weide, and A. C. Yao. Optimization Expected-Time Algorithms for Closest Point Problems. *ACM Transactions on Mathematical Software*, 6:563-580, 1980.
- [8] B. Betrò and R. Rotondi. A Bayesian Algorithm for Global Optimization. *Annals of Operations Research*, 1:111-128, 1984.

- [9] G. Boender and A. Rinnooy Kan. Bayesian Stopping Rules for Multistart Global Optimization Methods. *Mathematical Programming*, 37:59–80, 1987.
- [10] F. H. Branin. Widely Convergent Methods for Finding Multiple Solutions of Simultaneous Nonlinear Equations. *IBM Journal of Research and Development*, pgs. 504–522, 1972.
- [11] R. P. Brent. *Algorithms for Minimization Without Derivatives*. Prentice-Hall, New Jersey, 1973.
- [12] R.G. Brown, J.W. Chinneck, and G.K. Karam. Optimization with Constraint Programming Systems. In R. Sharda, B.L. Golden, E. Wasil, O. Balci, and W. Stewart, editors, *Impacts of Recent Computer Advances on Operations Research*, Vol. 9 of *Publications in Operations Research Series*, pgs. 463–473, New York, 1989. North-Holland.
- [13] J. W. Chinneck and E. W. Dravnieks. Locating Minimal Infeasible Constraints in Linear Programs. *ORSA Journal on Computing*, 3(2):157–168, 1991.
- [14] W. Chuba and W. Miller. Quadratic Convergence in Interval Arithmetic, Part I. *BIT*, 12:284–290, 1972.
- [15] A. R. Conn. Chairman's Column. *SIAG/OPT Views and News : A Forum for the SIAM Activity Group on Optimization*, (4):2, 1994.
- [16] A.R. Conn, N.I.M. Gould, and Ph.L. Toint. *LANCELOT: A Fortran Package for Large-scale Nonlinear Optimization*, Vol. 17 of *Springer Series in Computational Mathematics*. Springer-Verlag, Berlin, 1992.
- [17] P. Cooke. Statistical Inference for Bounds on Random Variables. *Biometrika*, 66:367–374, 1979.
- [18] A. Dekkers and E. Aarts. Global Optimization and Simulated Annealing. *Mathematical Programming*, 50:367–393, 1991.
- [19] J. J. Dinkel and M. Tretter. An Interval Arithmetic Approach to Sensitivity Analysis in Geometric Programming. *Operations Research*, 35(6):859–866, 1987.

- [20] J.J. Dinkel, M. Tretter, and D. Wong. Interval Newton Methods and Perturbed Problems. *Applied Mathematics and Computation*, 28(3):211-222, 1988.
- [21] L. C. W. Dixon and G. P. Szegö. The Global Optimization Problem. In L. C. W. Dixon and G. P. Szegö, editors, *Towards Global Optimization 2*, pgs. 1-15. North-Holland, Amsterdam, 1978.
- [22] L. C. W. Dixon and G. P. Szegö, editors. *Towards Global Optimization 2*. North Holland, Amsterdam, 1978.
- [23] E. W. Dravnieks. Identifying Minimally Inconsistent Sets of Constraints: Deletion, Squeeze and Sensitivity Filtering. Master's thesis, Carleton University, 1989.
- [24] Y. G. Evtushenko. *Numerical Optimization Techniques*. Optimization Software, Inc., New York, 1985.
- [25] R. Fletcher. *Practical Methods of Optimization*. John Wiley & Sons, Cichester, Second edition, 1987.
- [26] C.A. Floudas and P.M. Pardalos. *A Collection of Test Problems for Constrained Global Optimization Algorithms*, Vol. 455 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin, 1990.
- [27] R. P. Ge. The Theory of the Filled Function Method for Finding a Global Minimizer of a Nonlinearly Constrained Minimization. *Journal of Computational Mathematics*, 5:1-9, 1987.
- [28] R. P. Ge and Y. F. Qin. A Class of Filled Functions for Finding Global Minimizers of a Function of Several Variables. *Journal of Optimization Theory and Applications*, 54(2):241-252, August 1987.
- [29] F. Giannessi and F. Niccolucci. Connections between Nonlinear and Integer Programming Problems. In *Symposia Mathematica*, Vol. 19, pgs. 161-176, London, 1976. Academic Press.
- [30] F. Glover. Tabu Search : Part I. *ORSA Journal on Computing*, 1(3):190-206, 1989.
- [31] F. Glover. Tabu Search : Part II. *ORSA Journal on Computing*, 2(1):4-32, 1990.

- [32] D. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, Reading, 1989.
- [33] B. L. Golden and F. B. Alt. Interval Estimation of a Global Optimum for Large Combinatorial Problems. *Naval Research Logistics Quarterly*, 26:69-77, 1979.
- [34] A. A. Goldstein and J. F. Price. On Descent from Local Minima. *Mathematics of Computation*, 25(115):569-574, July 1971.
- [35] S. Gomez and A. V. Levy. Numerical Analysis. In J. P. Hennart, editor, *Numerical Analysis : Proceedings of the Third IIMAS Workshop Held at Cocoyoc, Mexico, January 1981*, Vol. 909 of *Lecture Notes in Mathematics*, pgs. 34-47, Berlin, 1982. Springer-Verlag.
- [36] Griewank. Direct Calculation of Newton Step Without Accumulating Jacobians. Technical report, Mathematics Computer Science Division, Argonne National Laboratories, 1990.
- [37] A. O. Griewank. Generalized Descent for Global Optimization. *Journal of Optimization Theory and Applications*, 34(1):11-39, May 1981.
- [38] E. R. Hansen. A Globally Convergent Interval Analytic Method for Computing and Bounding Real Roots. *BIT*, 18:415-424, 1978.
- [39] E. R. Hansen. Interval Forms of Newton's Method. *Computing*, 20:153-163, 1978.
- [40] E. R. Hansen. Global Optimization using Interval Analysis : The One-Dimensional Case. *Journal of Optimization Theory and Applications*, 29(3):331-344, November 1979.
- [41] E. R. Hansen. Global Optimization Using Interval Analysis - the Multi-Dimensional Case. *Numerische Mathematik*, 34:247-270, 1980.
- [42] E. R. Hansen. Global Optimization with Data Perturbations. *Computers and Operations Research*, 11(2):97-104, 1984.
- [43] E. R. Hansen and R. I. Greenberg. An Interval Newton Method. *Applied Mathematics and Computation*, 12(2/3):89-98, 1983.

- [44] E. R. Hansen and S. Sengupta. Bounding Solutions of Systems of Equations using Interval Analysis. *BIT*, 21:203-211, 1981.
- [45] E. R. Hansen and G. W. Walster. Equality Constrained Global Optimization. Preprint.
- [46] Eldon Hansen. *Global Optimization Using Interval Analysis*, Vol. 165 of *Monographs and Textbooks in Pure and Applied Mathematics*. Marcel Dekker, Inc., New York, 1992.
- [47] J. D. Hill. A Search Technique for Multimodal Surfaces. *I.E.E.E. Transactions on Systems Science and Cybernetics*, 5:2-8, 1969.
- [48] D. M. Himmelblau. *Applied Nonlinear Programming*. McGraw-Hill, New York, 1972.
- [49] W. Hock and K. Schittkowski. *Test Examples for Nonlinear Programming Codes*. No. 187 in *Lecture Notes in Economics and Mathematical Systems*. Springer-Verlag, Berlin, 1981.
- [50] R. Horst. On the Global Minimization of Concave Functions : Introduction and Survey. *Operations Research Spektrum*, 6:195-205, 1984.
- [51] R. Horst. A General Class of Branch and Bound Methods in Global Optimization with some new Approaches for Concave Minimization. *Journal of Optimization Theory and Applications*, 51:271-291, 1985.
- [52] R. Horst. Deterministic Global Optimization with Partition Sets whose Feasibility is not Known. Application to Concave Minimization, Reverse Convex Constraints, D.C.-Programming and Lipschitzian Optimization. *Journal of Optimization Theory and Applications*, 58(1):11-37, July 1988.
- [53] R. Horst. Deterministic Methods in Constrained Global Optimization : Some Recent Advances and New Fields of Application. *Naval Research Logistic*, 37:433-471, 1990.
- [54] R. Horst and H. Tuy. On the Convergence of Global Methods in Multiextremal Optimization. *Journal of Optimization Theory and Applications*, 54(2):253-271, August 1987.

- [55] R. Horst and H. Tuy. *Global Optimization : Deterministic Approaches*. Springer-Verlag, Berlin, 1990.
- [56] K. Ichida and Y. Fujii. An Interval Arithmetic Method for Global Optimization. *Computing*, 23:85-97, 1979.
- [57] S. Incerti, V. Parisi, and F. Zirilli. A New Method for Solving Nonlinear Simultaneous Equations. *SIAM Journal on Numerical Analysis*, 16:779-789, 1979.
- [58] S. Inomata and M. Cumada. On the Golf Method. *Bulletin of the Electrotechnical Laboratory*, 25:495-512, 1964.
- [59] C. Jansson. A Global Optimization Method using Interval Arithmetic. In L. Atanassova and J. Herzberger, editors, *Proceedings of the Third International IMACS-GAMM Symposium on Computer Arithmetic and Scientific Computing (SCAN-91)*, pgs. 259-267, North-Holland, 1992. Elsevier Science Publishers B.V.
- [60] N.K. Jha and N. Mokaddem. Computer Aided Optimization and Design for Manufacturing. In R. Raghavan and T.J. Cokonis, editors, *Computers in Engineering*, Vol. 2 of *Proceedings of the 1987 ASME International Computer in Engineering Conference and Exhibition*, pgs. 105-112, New York, 1987. The American Society of Mechanical Engineers.
- [61] B. Kalantari and J. B. Rosen. An Algorithm for Global Minimization of Linearly Constrained Concave Quadratic Functions. *Mathematics of Operations Research*, 12(3):544-561, August 1987.
- [62] G. Kedem. Automatic Differentiation of Computer Programs. *ACM Transactions on Mathematical Software*, 6(2):150-165, 1980.
- [63] A. Khachatryan. Statistical Mechanics Approach in Minimizing a Multivariate Function. *Journal of Mathematical Physics*, 27:1834-1838, 1986.
- [64] R. Krawczyk. Newton-Algorithmen zur Bestimmung von Nullstellen mit Fehler-schranken. *Computing*, 4:187-201, 1969.
- [65] R. Krawczyk and A. Neumaier. Interval Slopes for rational Functions and Associated Centered Forms. *SIAM Journal on Numerical Analysis*, 22(3):604-616, June 1985.

- [66] R. Krawczyk and K. Nickel. Die zentrische Form in der Intervallarithmetik, ihre quadratische Konvergenz und ihre Inklusionsisotonie. *Computing*, 28:117–137, 1982.
- [67] C. Lemaréchal. Mathematical Classification of Optimization Problems. In J.-P. Penot, editor, *New Methods in Optimization and their Industrial Uses : State of the Art, Recent Advances , Perspectives ; Proceedings of the Symposia held in Pau and Paris*, Vol. 87 of *International Series of Numerical Mathematics*, pgs. 89–95, Basel, 1989. Birkhäuser Verlag.
- [68] A. V. Levy and A. Montalvo. The Tunneling Algorithm for the Global Minimization of Functions. *SIAM Journal of Scientific and Statistical Computing*, 6(1):15–29, January 1985.
- [69] A. V. Levy, A. Montalvo, S. Gomez, and A. Calderon. Topics in Global Optimization. In J. P. Hennart, editor, *Numerical Analysis : Proceedings of the Third HMAS Workshop Held at Cocoyoc, Mexico, January 1981*, Vol. 909 of *Lecture Notes in Mathematics*, pgs. 18–33, Berlin, 1982. Springer-Verlag.
- [70] W. A. Lodwick. Computing Bounds of Variables in Nonlinear Programming Problems Prior to Optimization. Copy, January 1991.
- [71] D. G. Luenberger. *Linear and Nonlinear Programming*. Addison Wesley, 1984.
- [72] G. P. McCormick. Computability of Global Solutions to Factorable Nonconvex Programs : Part I : Convex Underestimating Problems. *Mathematical Programming*, 10:147–175, 1976.
- [73] G. P. McCormick. *Nonlinear Programming : Theory, Algorithms and Applications*. John Wiley & Sons, Toronto, 1983.
- [74] C. C. Meewella and D. Q. Mayne. An Algorithm for Global Optimization of Lipschitz Continuous Functions. *Journal of Optimization Theory and Applications*, 57(2):307–322, May 1988.
- [75] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller. Equation of State Calculations by Fast Computing Machines. *The Journal of Chemical Physics*, 21:1087–1092, 1953.

- [76] W. Miller. Quadratic Convergence in Interval Arithmetic, Part II. *BIT*, 12:291–298, 1972.
- [77] R. H. Mladineo. An Algorithm for Finding the Global Maximum of a Multivariate Function. *Mathematical Programming*, 34:188–200, 1986.
- [78] J. B. Mockus and L. J. Mockus. Baysian Approach to Global Optimization and Applications to Multiobjective and Constrained Problems. *Journal of Optimization Theory and Applications*, 70(1):157–172, July 1991.
- [79] I. B. Mohd. *Global Optimization using Interval Arithmetic*. PhD thesis, University of St. Andrews, 1986.
- [80] R. E. Moore. *Interval arithmetic and automatic error analysis in digital computation*. PhD thesis, Stanford University, 1962.
- [81] R. E. Moore. *Interval Analysis*. Prentice-Hall, Englewood Cliffs, 1966.
- [82] R. E. Moore. On Computing the range of values of a rational Function of n Variables over a Bounded region. *Computing*, 16:1–15, 1976.
- [83] R. E. Moore. A Test for the Existence of Solutions to Non-linear Systems. *SIAM Journal on Numerical Analysis*, 14:611–615, 1977.
- [84] R. E. Moore. *Methods and Applications of Interval Analysis*. SIAM, Philadelphia, 1979.
- [85] K. G. Murty and S. N. Kabadi. Some NP-Complete Problems in Quadratic Nonlinear Programming. *Mathematical Programming*, 39:117–129, 1987.
- [86] A. Neumaier. Existence of Solutions of Piecewise Differentiable Systems of Equations. *Archiv der Mathematik*, 47:443–447, 1986.
- [87] A. Neumaier. *Interval Methods for Systems of Equations*, Vol. 37 of *Encyclopedia of Mathematics and its Applications*. Cambridge University Press, Cambridge, 1990.
- [88] P. M. Pardalos and J. B. Rosen. *Constrained Global Optimization : Algorithms and Applications*, Vol. 268 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin, 1987.

- [89] S. A. Pijavskij. An Algorithm for Finding the Absolute Extremum of a Function. *USSR Comput. Math. and Math. Phys.*, pgs. 57-67, 1972.
- [90] J. Pintér. Branch- and Bound Algorithms for Solving Global Optimization Problems with Lipschitzian Structure. *Optimization*, 19(1):101-110, 1988.
- [91] L. B. Rall. *Automatic Differentiation : Techniques and Applications*, Vol. 120 of *Lecture Notes in Computer Science*. Springer-Verlag, New York, 1981.
- [92] L. B. Rall. Mean Value and Taylor Forms in Interval Analysis. *SIAM Journal on Mathematical Analysis*, 14(2):223-238, 1983.
- [93] H. Ratschek. Zentrische Formen. *ZAMM*, 58:T434-T436, 1978.
- [94] H. Ratschek. Centered Forms. *SIAM Journal on Numerical Analysis*, 17(5):656-662, October 1980.
- [95] H. Ratschek. Optimal Approximations in Interval Analysis. In K. Nickel, editor, *Interval Mathematics 1980, Proceedings of the International Symposium, Freiburg 1980*, pgs. 181-202, New York, 1980. Academic Press.
- [96] H. Ratschek and J. Rokne. *Computer Methods for the Range of Functions*. Ellis Horwood Series in Mathematics and its Applications. Ellis Horwood Limited, Chichester, 1984.
- [97] H. Ratschek and J. Rokne. *New Computer Methods for Global Optimization*. Ellis Horwood Series in Mathematics and its Applications. Ellis Horwood Limited, Chichester, 1988.
- [98] H. Ratschek and G. Schroeder. Centered Forms for Functions in Several Variables. *Journal of Mathematical Analysis and Applications*, 82:543-552, 1981.
- [99] H. Ratschek and R. L. Voller. Global Optimization Over Unbounded Domains. *ZAMM*, 68(5):T447-T448, 1988.
- [100] A. H. G. Rinnooy Kan and G. T. Timmer. Stochastic Global Optimization Methods Part I : Clustering Methods. *Mathematical Programming*, 39:27-56, 1987.

- [101] A. H. G. Rinnooy Kan and G. T. Timmer. Stochastic Global Optimization Methods Part II : Multi Level Methods. *Mathematical Programming*, 39:57-78, 1987.
- [102] A. H. G. Rinnooy Kan and G. T. Timmer. Global Optimization : A Survey. In J.-P. Penot, editor, *New Methods in Optimization and their Industrial Uses : State of the Art, Recent Advances , Perspectives ; Proceddings of the Symposia held in Pau and Paris*, Vol 87 of *International Series of Numerical Mathematics*, pgs. 133-155, Basel, 1989. Birkhäuser Verlag.
- [103] S. M. Robinson. Computable Error Bounds for Nonlinear Programming. *Mathematical Programming*, 5:235-242, 1973.
- [104] J. Rokne. A Low Complexity Explicit Rational Centered Form. *Computing*, 34:261-263, 1985.
- [105] J. Rokne. Low Complexity k-Dimensional Centered Forms. *Computing*, 37:247-253, 1986.
- [106] J. Rokne and P. Bao. Interval Taylor Forms. *Computing*, 39:247-259, 1987.
- [107] R. Y. Rubinstein. *Simulation and the Monte Carlo Method*. John Wiley, New York, 1981.
- [108] H.-P. Schwefel. *Numerische Optimierung von Computer-Modellen mittels der Evolutionsstrategie*, Vol. 26 of *Interdisciplinary Systems Research*. Birkhäuser Verlag, Basel, 1977.
- [109] Zuhe Shen. Interval Test and Existence Theorem. In K. Nickel, editor, *Interval Mathematics 1985, Proceedings of the International Symposium, Freiburg 1985*, Vol. 212 of *Lecture Notes in Computer Science*, pgs. 175-196, Berlin, 1986. Springer-Verlag.
- [110] B. O. Shubert. A Sequential Method Seeking the Global Maximum of a Function. *SIAM Journal on Numerical Analysis*, 9:379-388, 1972.
- [111] S. Skelboe. Computation of Rational Interval Functions. *BIT*, 14:87-95, 1974.
- [112] S. Smith and L. Lasdon. Solving Large Sparse Nonlinear Programs Using GRG. *ORSA JOURNAL on Computing*, 4(1):2-15, 1992.

- [113] J. A. Snyman and L. P. Fatti. A Multi-start Global Minimization Algorithm with Dynamic Search Trajectories. *Journal of Optimization Theory and Applications*, 54(1):121–141, July 1987.
- [114] P. T. Thach and H. Tuy. The Relief Indicator Method for Constrained Global Optimization. *Naval Research Logistics*, 37:473–497, 1990.
- [115] A. Törn and A. Žilinskas. *Global Optimization*, Vol. 350 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin, 1989.
- [116] A. A. Törn. A Sampling- Search- Clustering Approach for Exploring Feasible/Efficient Solutions of MCDM Problems. *Computers and Operations Research*, 7:67–79, 1980.
- [117] D. Vanderbilt and S. G. Louie. A Monte Carlo Simulated Annealing Approach to Optimization over Continuous Variables. *Journal of Computational Physics*, 56:259–271, 1984.
- [118] J. Vignes. Optimal Implementation of Optimization Methods and Estimation of the Accuracy of the Results. In J.-P. Penot, editor, *New Methods in Optimization and their Industrial Uses : State of the Art, Recent Advances , Perspectives ; Proceedings of the Symposia held in Pau and Paris*, Vol. 87 of *International Series of Numerical Mathematics*, pgs. 219–227, Basel, 1989. Birkhäuser Verlag.
- [119] A. V. Vilkov, N. P. Zhidkov, and B. M. Shchedrin. A Method of Search for the Global Minimum of a Function of One Variable. *Journal of Computational Mathematics and Mathematical Physics*, 15:1040–1042, 1975.
- [120] A. Wright. Genetic Algorithms for Real Parameter Optimization. In G.J.E. Rawlins, editor, *Foundations of Genetic Algorithms*, pgs. 205–218. Morgan Kaufmann Publishing, San Mateo, 1991.
- [121] Jianzhong Zhang and Dan Sha. A Priority Measure in Interval Methods for Constrained Global Optimization. *Computers & Operations Research*, 21(5):511–519, 1994.
- [122] Shen Zuhe and M. A. Wolfe. On Interval Enclosures Using Slope Arithmetic. *Applied Mathematics and Computation*, 39(1):89–105, 1990.

END

2 6 - 0 5 - 9 5

FIN