DEVELOPMENT OF MODAL INTERVAL ALGORITHM FOR SOLVING CONTINUOUS MINIMAX PROBLEMS

by

XIN LUO

MIN SUN, COMMITTEE CHAIR
SHUHUI LI
BRUCE TRACE
JAMES WANG
SHAN ZHAO

A DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy
in the Department of Mathematics
in the Graduate School of
The University of Alabama

TUSCALOOSA, ALABAMA

2017

ABSTRACT

While there are a large variety of effective methods developed for solving more traditional minimization problems, much less success has been reported in solving the minimax problem $\min_{u\in U}\max_{v\in V} f(u,v)$ where $U \times V$ is a fixed domain in $\mathbb{R}^n$. Most of the existing work deal with a discrete V or even a finite V. Continuous minimax problems can be applied to engineering, finance and other fields. Sainz in 2008 proposed a modal interval algorithm based on their semantic extensions to solve continuous minimax problems. We developed an improved algorithm using modal intervals to solve unconstrained continuous minimax problems. A new interval method is introduced by taking advantage of both the original minimax problem and its dual problem. After theoretical analysis of major issues, the new algorithm is implemented in the framework of uniform partition of the search domain. Various improvement techniques including more bisecting choices, sampling methods and deletion conditions are applied to make the new method more powerful. Preliminary numerical results provide promising evidence of its effectiveness.

DEDICATION

This dissertation is dedicated to my beloved parents and my wife. To my parents: Thank you for all your unconditional love and support. You are the best parents in the world. I really appreciate your sacrifices and I wouldn't have been able to get to this stage without you. To my wife: Thank you for standing besides me all the time. I wouldn't have gotten through this if it wasn't for you.

ACKNOWLEDGMENTS

I would like to express my deepest gratitude to my advisor Dr. Min Sun for his excellent guidance, caring, patience, and insightful suggestions throughout my research career.

I would like to thank the committee members who were more than generous with their expertise and precious time.

I would like to extend my thanks to my colleagues in Mathematics Department. They are always willing to help me.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Minimax Problem

Minimax problem is well known in many fields. It is simply stated as $\min_{u\in U}\max_{v\in V} f(u,v)$ where $U \times V$ is a fixed domain in $\mathbb{R}^n$. We say that $(u^*, v^*) \in (U, V)$ is a minimax point of $f(u, v)$ in (U, V) if

$$f(u^*, v^*) = \min_{u\in U}\max_{v\in V} f(u, v).$$

Such examples would include finding the optimal strategies for both players in game theory, engineering design problems, and the Mandelshtam problem in electrical circuit theory [18]. Most of the existing work deal with a discrete V or even a finite V. The continuous case has been studied in Chebyshev approximation problem [18] and other applications such as finding minimax hedging strategy in finance and economics [14].

Example 1.1.0.1. ***Chebyshev Approximation.*** *Given a function $g : Y \subseteq \mathbb{R}^m \to \mathbb{R}$ and a function space P_n of functions $p_x : \mathbb{R}^m \to \mathbb{R}$, the Chebyshev approximation p_x of g in P_n solves the following minimax problem*

$$\min_{x}\max_{y\in Y} (g(y) - p_x(y))^2.$$

Example 1.1.0.2. ***Minimax Hedging Strategy.*** *The minimax hedging strategy aims to minimize the maximum potential hedging error between time t and $t + 1$. The objective*

function used is thus the potential hedging error. The minimax problem formulation is

$$\min_{x_t} \max_{y_{t+1}^S} f(x_t, y_{t+1}^S)$$

subject to

$$y_{t+1}^{S,lower} \leq y_{t+1}^S \leq y_{t+1}^{S,upper}$$

where x_t is the number of shares to be held at time t, y_t^S is the stock price at time t, and $y_{t+1}^{S,lower} \leq y_{t+1}^S \leq y_{t+1}^{S,upper}$ is the range defined under the scenario.

1.2 Some Facts about Minimax Problem

Minimax problem is actually a special type of minimization problem. Let's consider a minimization problem of the form $\min_{x\in X} f(x)$. A basic fact is that $\min_{x\in X_1} f(x) \geq \min_{x\in X} f(x)$ for any $X_1 \subseteq X$. $\min_{x\in X} f(x)$ will become a minimax problem if $f(x) = \max_{y\in Y} f(x,y)$. Some basic facts about $\min_{x\in X} \max_{y\in Y} f(x,y)$ are:

- It is simplified as $\min_{x\in X} f(x,y)$ if Y consists of one single value y.
- It becomes $\min_{x\in X} \big(\max\{f(x,y_1), f(x,y_2), \cdots, f(x,y_n)\}\big)$ with objective function as the maximum of a finite number of function values if Y consists of a finite number of values $\{y_1, y_2, \cdots, y_n\}$.
- It is a minimax problem $\min_{x\in X} \max_{y\in Y} f(x,y)$ if Y consists of infinitely many values, e.g., Y is an interval.

There are a large variety of effective methods developed for solving $\min_{x\in X} f(x,y)$ and $\min_{x\in X} \big(\max\{f(x,y_1), f(x,y$
A well-known method to solve $\min_{x\in X} f(x,y)$ is the classical Newton's method with iteration formula $x_{k+1} := x_k - \frac{f'(x_k)}{f''(x_k)}$. However, it's possible to miss other minimum points. The interval version of Newton's method would fix this issue without missing any possible minimum points, which is one major advantage of using interval method to solve minimization problems.

As for $\min\limits_{x\in X}\max\limits_{y\in Y} f(x,y)$, there is no close form to rewrite the objective function $f(x) = \max\limits_{y\in Y} f(x,y)$ of this type of minimization problems, and it is usually not computable directly. Unlike $\min\limits_{x\in X} f(x)$, it is not obvious to see how the change of domain affects the minimax value for minimax problem $\min\limits_{x\in X}\max\limits_{y\in Y} f(x,y)$. We try to illustrate how the domain change affect works in 6 cases shown below.

Case 1: Given a fixed X, $Y_1 \subseteq Y \subseteq Y_2$ implies that

$$\min_{x\in X}\max_{y\in Y_1} f(x,y) \leq \min_{x\in X}\max_{y\in Y} f(x,y) \leq \min_{x\in X}\max_{y\in Y_2} f(x,y).$$

Case 2: Given a fixed Y, and $X_1 \subseteq X \subseteq X_2$, then

$$\min_{x\in X_2}\max_{y\in Y} f(x,y) \leq \min_{x\in X}\max_{y\in Y} f(x,y) \leq \min_{x\in X_1}\max_{y\in Y} f(x,y).$$

Case 3: Given $X \subseteq X_2$ and $Y \subseteq Y_2$, the following examples show that we can not decide which one is greater between $\min\limits_{x\in X}\max\limits_{y\in Y} f(x,y)$ and $\min\limits_{x\in X_2}\max\limits_{y\in Y_2} f(x,y)$.

For $X = [1,3] \subseteq X_2 = [1,4]$ and $Y = [1,3] \subseteq Y_2 = [1,4]$, we have that

$$\min_{x\in X}\max_{y\in Y}(x^2+y) = 4 < \min_{x\in X_2}\max_{y\in Y_2}(x^2+y) = 5.$$

However, for $X = [1,3] \subseteq X_2 = [0,3]$ and $Y = [1,3] \subseteq Y_2 = [0,3]$, we have that

$$\min_{x\in X}\max_{y\in Y}(x^2+y) = 4 > \min_{x\in X_2}\max_{y\in Y_2}(x^2+y) = 3.$$

Case 4: Given $X_1 \subseteq X$ and $Y_1 \subseteq Y$, the examples below show that we can not decide which one is greater between $\min\limits_{x\in X}\max\limits_{y\in Y} f(x,y)$ and $\min\limits_{x\in X_1}\max\limits_{y\in Y_1} f(x,y)$.

For $X_1 = [1,2] \subseteq X = [1,3]$ and $Y_1 = [1,2] \subseteq Y = [1,3]$, we have that

$$\min_{x\in X}\max_{y\in Y}(x^2+y) = 4 > \min_{x\in X_1}\max_{y\in Y_1}(x^2+y) = 3.$$

However, for $X_1 = [2,3] \subseteq X = [1,3]$ and $Y_1 = [2,3] \subseteq Y = [1,3]$, we have that

$$\min_{x\in X}\max_{y\in Y}(x^2+y) = 4 < \min_{x\in X_1}\max_{y\in Y_1}(x^2+y) = 7.$$

Case 5: Given $X \subseteq X_2$ and $Y_1 \subseteq Y$, then $\min_{x\in X}\max_{y\in Y} f(x,y) \geq \min_{x\in X_2}\max_{y\in Y_1} f(x,y)$.

Case 6: Given $X_1 \subseteq X$ and $Y \subseteq Y_2$, then

$$\min_{x\in X}\max_{y\in Y} f(x,y) \leq \min_{x\in X_1}\max_{y\in Y_2} f(x,y).$$

The results in 6 cases tell us that it is difficult to tell how the change of domain affects the minimax value directly in real examples, which increases the degree of complexity of solving minimax problems.

1.3 Solve Minimax Problem by Interval Method

Optimization problems can be solved by methods using interval arithmetic. The work in interval arithmetic began especially with the papers of Moore [10]. An introduction to interval arithmetic and classic interval methods is given in [13] and many other references. Shen, Neumaier, and Eiermann (1990) proposed an interval method to compute the minimax problem of a twice continuously differentiable objective function [18]. Cao, Li, and Wu (2002) proposed an interval method for solving global solutions of continuous minimax problems with merely Lipschitz continuous objective functions [1]. Sainz, Herrero, Armengol, and Vehi (2008) proposed a minimax approach using modal interval analysis [16]. Goldsztejn studied generalized interval natural extension and generalized interval mean-value extension in modal interval analysis [3, 4]. Other than interval methods, Lu and Basar in 1995 proposed genetic algorithm [9], Storn and Price introduced differential evolution [21], to solve minimax problems with linear constraints. Lu, Cao, Yuan, and Zhou in 2008 designed a Karush-Kuhn-Tucker based minimax method [6]. Parpas and Rustem (2009) approximated the

minimax solution using techniques from semidefinite programming [12]. In this thesis, we propose an improved version of modal interval method to solve continuous minimax problems. Improvements techniques will be studied. In addition, a brand new algorithm is also designed for objective functions satisfying certain conditions.

CHAPTER 2

INTERVAL ANALYSIS AND MINIMAX ALGORITHM

In this chapter, some basic tools and techniques of both classic and modal interval analysis are introduced. The existing Sainz's minimax algorithm using modal interval analysis will also be presented.

2.1 Classic Interval Analysis

2.1.1 Arithmetic and Operations

Let $I(\mathbb{R})$ be the set of real compact intervals $[a, b]$ where $a, b \in \mathbb{R}$.

Definition 2.1.1.1. *For $X = [a, b] \in I(\mathbb{R})$,*

1. *The lower and upper bounds of X: $Inf(X) = a$, $Sup(X) = b$.*
2. *The midpoint of X: $mid(X) = \frac{a+b}{2}$.*
3. *The width of X: $w(X) = b - a$.*

Example 2.1.1.2. *If $X = [1, 2]$, then $Inf(X) = 1$, $Sup(X) = 2$, $mid(X) = \frac{1+2}{2} = 1.5$, and $w(X) = 2 - 1 = 1$.*

Definition 2.1.1.3. ***Interval Arithmetic Rules.*** *For any two intervals $[a, b]$ and $[c, d]$ in $I(\mathbb{R})$,*

1. *Addition: $[a, b] + [c, d] = [a + c, b + d]$.*
2. *Subtraction: $[a, b] - [c, d] = [a - d, b - c]$.*
3. *Multiplication: $[a, b] \times [c, d] = [\min\{ac, ad, bc, bd\}, \max\{ac, ad, bc, bd\}]$.*

4. *Division:* $[a,b]/[c,d] = [a,b] \times [1/d, 1/c]$ *if* $0 \notin [c,d]$.

Example 2.1.1.4. *If we have two classic intervals* $[1,2]$ *and* $[-4,-2]$*, then*

1. $[1,2] + [-4,-2] = [-3,0]$.

2. $[1,2] - [-4,-2] = [3,6]$.

3. $[1,2] \times [-4,-2] = [-8,-2]$.

4. $[1,2]/[-4,-2] = [1,2] \times [-1/2,-1/4] = [-1,-1/4]$.

It is easy to see that classic intervals are closed under interval arithmetic rules. However, the division rule could be very weak (possibly producing infinite intervals) as the divisor interval could contain 0 in practice. Therefore, Hansen in 1980 introduced extended interval division [5].

$$[a,b]/[c,d] = \begin{cases} [a/d, b/c] & \text{if } 0 \notin [c,d], \\ [a/d, \infty) & \text{if } [a,b] \geq 0 \text{ and } c = 0, \\ (-\infty, b/c] & \text{if } [a,b] \geq 0 \text{ and } d = 0, \\ (-\infty, a/d] & \text{if } [a,b] \leq 0 \text{ and } c = 0, \\ [b/c, \infty) & \text{if } [a,b] \leq 0 \text{ and } d = 0, \\ (-\infty, b/c] \cup [a/d, \infty) & \text{if } [a,b] \geq 0 \text{ and } 0 \in [c,d], \\ (-\infty, a/d] \cup [b/c, \infty) & \text{if } [a,b] \leq 0 \text{ and } 0 \in [c,d], \\ (-\infty, \infty) & \text{if } 0 \in [a,b] \text{ and } 0 \in [c,d]. \end{cases}$$

Definition 2.1.1.5. ***Interval Operations.*** *Let* $A, B, C, D \in I(\mathbb{R})$*, and* $*$ *be any interval arithmetic rule, then*

1. $A \cap B = \{x | x \in A \text{ and } x \in B\}$.

2. $A \cup B = \{x | x \in A \text{ or } x \in B\}$.

3. $A + B = B + A$.

4. $A + (B + C) = (A + B) + C$.

5. $AB = BA$.

6. $A(BC) = (AB)C$.

7. *$A \subseteq B, C \subseteq D$ implies $A * C \subseteq B * D$ if $B * D$ is defined.*

2.1.2 Inclusion Functions

In the treatment of optimization problems using interval arithmetic the main tool is the concept and application of inclusion functions. In classic interval analysis, one desirable extension of an $\mathbb{R}^n$ to $\mathbb{R}$ continuous function $f(x_1, x_2, ..., x_n)$ is the range $f(X)$ of all $f-$values over interval domain X. However, the exact range $f(X)$ is difficult to compute directly. Inclusion functions $F(X)$ for f are introduced instead. They contain the exact range of f and are easy to compute.

Definition 2.1.2.1. ***Natural interval extension.*** *The natural interval extensions of elementary functions (e.g. $\sin x$, $\log x$) are straightforward. You can actually determine their exact ranges. Thus $Log(X) = [\log(Inf(X)), \log(Sup(X))]$. Usually, additional controlled directed rounding is also used to make sure that $Log(X)$ would contain the exact range $[\log(Inf(X)), \log(Sup(X))]$ even under round-off errors. Let $f : D \to \mathbb{R}$, $D \subseteq \mathbb{R}^n$ be any composition of elementary functions. Let $I(D) = \{Y | Y \in I^n(\mathbb{R}), Y \subseteq D\}$ and $X \in I(D)$, then the natural interval extension of f to X is defined as that expression which is obtained from the expression $f(x)$ by replacing each occurrence of the variable x by the interval X.*

Besides natural interval extension, centered forms are also basic inclusion functions with special futures that were first introduced by Moore (1966) [11]. The most important centered forms are the meanvalue form and the Taylor form, described below.

Definition 2.1.2.2. ***Meanvalue form.*** *Let $f : D \to \mathbb{R}$, $D \subseteq \mathbb{R}^n$ be differentiable, $I(D) = \{Y | Y \in I^n(\mathbb{R}), Y \subseteq D\}$, and let $F' : I(D) \to I^n(\mathbb{R})$ be an inclusion function for the gradient, f'. Then $T_1 : I(D) \to I(\mathbb{R})$ defined by*

$$T_1(X) = f(c) + (X - c)^T F'(X) \text{ for } X \in I(D),$$

where $c = mid(X)$ or also some other point of X, is called the meanvalue form inclusion function of f.

Theorem 2.1.2.3. (Krawczyk, Nickel [7], 1982) *If F' is Lipschitz then the meanvalue form T_1 is of convergence order 2, i.e., $w(T_1(X)) - w(f(X)) = w(X)^2 = \mathcal{O}(w(X)^2)$.*

Definition 2.1.2.4. ***Taylor form.*** *Let $f : D \to \mathbb{R}$, $D \subseteq \mathbb{R}^n$ be twice differentiable, $I(D) = \{Y | Y \in I^n(\mathbb{R}), Y \subseteq D\}$, and let $F'' : I(D) \to I^{n \times n}(\mathbb{R})$ be an inclusion function for the Hessian matrix f''. Then $T_2 : I(D) \to I(\mathbb{R})$ defined by*

$$T_2(X) = f(c) + (X - c)^T f'(c) + \frac{1}{2}(X - c)^T F''(X)(X - c) \text{ for } X \in I(D),$$

where $c = mid(X)$ or any other point of X is called a Taylor form inclusion function of f.

Theorem 2.1.2.5. *If f is twice differentiable, and if f'' has a bounded inclusion function F'' then the Taylor form, T_2, is of convergence order 2, i.e., $w(T_2(X)) - w(f(X)) = w(X)^2 = \mathcal{O}(w(X)^2)$.*

Example 2.1.2.6. *Given $f(x) = x - x^2$ on $X = [0, 1]$, then $c = 1/2$, $f'(x) = 1 - 2x$, and $f''(x) = -2$. The exact range of $f(x)$ over $X = [0, 1]$ is $range(f(x)) = [0, 1/4]$. Following above definitions, natural interval extension is $F(X) = X - X^2 = [0, 1] - [0, 1][0, 1] = [0, 1] - [0, 1] = [-1, 1]$, the meanvalue form is $T_1(X) = [-1/4, 3/4]$, and the Taylor form is $T_2(X) = [0, 1/2]$. The inclusion relationship is $F(X) \supset T_1(X) \supset T_2(X) \supset range(f(x))$.*

2.2 Modal Interval Analysis

The main focus of classic interval arithmetic is to approximate left and right endpoints for the range of values of a function in one or more variables, in other words, to find an outer approximation of the exact range. In modal interval analysis, we are able to construct both inner and outer approximations of the exact range of a function instead of outer approximation only in classic interval analysis.

2.2.1 Arithmetic and Operations

Definition 2.2.1.1. *A modal interval is defined as* $X = (X', \forall)$ *or* $X = (X', \exists)$*, where* $X' \in I(\mathbb{R})$*. The set of modal intervals is represented by* $I^*(\mathbb{R})$*. A modal interval can be represented in the form*

$$X = [a, b] = \begin{cases} ([a, b]', \exists) & \text{if } a \leq b, \\ ([b, a]', \forall) & \text{if } a \geq b. \end{cases}$$

Modal intervals of the type $X = (X', \exists)$ *are called proper intervals, and modal intervals of the type* $X = (X', \forall)$ *are called improper intervals. The definitions of lower and upper bounds, midpoint, and width of a modal interval are consistent with the ones of a classic interval.*

Example 2.2.1.2. *A proper interval* $[1, 2]$ *is equal to* $([1, 2]', \exists)$*, and an improper interval* $[2, 1]$ *is equal to* $([1, 2]', \forall)$*.* $Inf([2, 1]) = 2$*,* $Sup([2, 1]) = 1$*,* $mid([1, 2]) = mid([2, 1]) = 3/2$*, and* $w([1, 2]) = w([2, 1]) = 1$*.*

Definition 2.2.1.3. ***Modal interval arithmetics rules.*** *For any two modal intervals* $[a, b]$ *and* $[c, d]$ *in* $I^*(\mathbb{R})$*,*

1. *Addition:* $[a, b] + [c, d] = [a + c, b + d]$.
2. *Subtraction:* $[a, b] - [c, d] = [a - d, b - c]$.
3. *Multiplication:* $[a, b] \times [c, d] = \big[\max\{a^+c^+, b^-d^-\} - \max\{a^-d^+, b^+c^-\}, \max\{a^-c^-, b^+d^+\} - \max\{a^+d^-, b^-c^+\}\big]$ *where* $x^+ = \max\{x, 0\}$*,* $x^- = \max\{-x, 0\}$.

4. *Division:* $[a,b]/[c,d] = [a,b] \times [1/d, 1/c]$ *if* $0 \notin [c,d]$.

Example 2.2.1.4. *If we have two modal intervals* $[1,2]$ *and* $[-2,-4]$*, then*

1. $[1,2] + [-2,-4] = [-1,-2]$.

2. $[1,2] - [-2,-4] = [5,4]$.

3. $[1,2] \times [-2,-4] = [-4,-4]$.

4. $[1,2]/[-2,-4] = [1,2] \times [-1/4, -1/2] = [-1/2, -1/2]$.

Definition 2.2.1.5. *For a modal interval* $X = [a,b]$ *with real numbers* a *and* b*, the operators* $Prop$, $Impr$, *and* $Dual$ *are defined as follows.*

1. $Prop([a,b]) = [a,b]' = \begin{cases} [a,b] & \text{if } a \leq b, \\ [b,a] & \text{if } a > b. \end{cases}$

2. $Impr([a,b]) = \begin{cases} [b,a] & \text{if } a \leq b, \\ [a,b] & \text{if } a > b. \end{cases}$

3. $Dual([a,b]) = [b,a]$.

Definition 2.2.1.6. *For any two modal intervals* A *and* B*,* $A \subseteq B$ *if and only if one of the following conditions holds.*

1. *Both* A *and* B *are proper and* A *is contained in* B.

2. A *is improper,* B *is proper, and* $Prop(A) \cap B \neq \emptyset$.

3. *Both* A *and* B *are improper and* $\mathrm{Dual}(B)$ *is contained in* $\mathrm{Dual}(A)$.

Definition 2.2.1.7. *For a sequence of modal intervals* $A(j) = [a_1(j), a_2(j)], j \in J$*, the operators* min, max, meet $\bigwedge$ *and* join $\bigvee$ *are defined as follows.*

1. $\min\limits_{j \in J} A(j) = [\min\limits_{j \in J} a_1(j), \min\limits_{j \in J} a_2(j)]$,

2. $\max\limits_{j\in J} A(j) = [\max\limits_{j\in J} a_1(j), \max\limits_{j\in J} a_2(j)]$,

3. $\bigwedge\limits_{j\in J} A(j) = [\max\limits_{j\in J} a_1(j), \min\limits_{j\in J} a_2(j)]$,

4. $\bigvee\limits_{j\in J} A(j) = [\min\limits_{j\in J} a_1(j), \max\limits_{j\in J} a_2(j)]$.

Both operators *meet* and *join* are isotonic; i.e., if $A_j \subseteq B_j$ for every $j \in J$, then $\bigwedge\limits_{j\in J} A(j) \subseteq \bigwedge\limits_{j\in J} B(j)$ and $\bigvee\limits_{j\in J} A(j) \subseteq \bigvee\limits_{j\in J} B(j)$. They are also associative and distributive with respect to each other.

2.2.2 Inclusion Functions

Sainz [16] in 2008 introduced the semantic functions f^* and f^{**} in modal intervals to play the role of the exact range in classic intervals, where

$$f^*(X) = \bigvee_{x_p\in X'_p} \bigwedge_{x_i\in X'_i} [f(x_p, x_i), f(x_p, x_i)] = [\min_{x_p\in X'_p} \max_{x_i\in X'_i} f(x_p, x_i), \max_{x_p\in X'_p} \min_{x_i\in X'_i} f(x_p, x_i)],$$

$$f^{**}(X) = \bigwedge_{x_i\in X'_i} \bigvee_{x_p\in X'_p} [f(x_p, x_i), f(x_p, x_i)] = [\max_{x_i\in X'_i} \min_{x_p\in X'_p} f(x_p, x_i), \min_{x_i\in X'_i} \max_{x_p\in X'_p} f(x_p, x_i)].$$

Here, $x = (x_p, x_i)$ is the component split corresponding to the proper and improper components of $X = (X_p, X_i)$.

Interesting properties of the semantic extensions are the isotonicity

$$X \subseteq Y \Rightarrow f^*(X) \subseteq f^*(Y) \text{ and } f^{**}(X) \subseteq f^{**}(Y)$$

and the inclusion $f^*(X) \subseteq f^{**}(X)$. For one-variable continuous functions $f(x)$ with $x \in X$, we have that $f^*(X) = f^{**}(X)$. For two-variable continuous functions $f(y, z)$ that are monotonic with respect to each incidence of every multi-incident variable in a domain (Y', Z'), we have $f^*(X) = f^{**}(X)$ where $X = (Y, Z)$. By this semantic definition, the minimax value turns out to be the left bound of $f^*(X)$, i.e., $\min\limits_{u\in U'} \max\limits_{v\in V'} f(u, v) = Inf(f^*(U, V))$. Therefore, solving minimax problem $\min\limits_{u\in U'} \max\limits_{v\in V'} f(u, v)$ becomes finding $Inf(f^*(U, V))$. Just like the ex-

act range $f(X')$, the semantic extensions f^* and f^{**} are out of reach for direct computation except for simple real functions. However, similar to the natural interval extension of $f(X')$ to an inclusion function $F(X')$, its modal extension is introduced by natural extension to modal intervals $X_1, \cdots, X_n$ and represented by $fR(X_1, \cdots, X_n)$. Computations with $fR(X)$ must be done with external truncation of each operator to obtain inclusion $f^*(X) \subseteq fR(X)$, and with the inner truncation to obtain inclusion $fR(X) \subseteq f^{**}(X)$ [16].

2.3 Sainz's Minimax Algorithm

Sainz's algorithm is based on modal interval analysis and branch-and-bound process. It approximates the minimax function value in terms of lower and upper approximations.

Let $X = (U, V)$, where U and V are proper and improper components, respectively. Let $\{U_1, U_2, \cdots, U_r\}$ be a partition of U, and $\{V_{1_i}, V_{2_i}, \cdots, V_{s_i}\}$ a partition of V for every $i = 1, 2, \cdots, r$. Each interval $U_i \times V_{k_i}$ is called a **cell**, each $V_{*_i}-$partition of V is called a **strip**, and the $U-$partition is called the **strips' list**. Figure 2.1 shows a partition on X where X consists of only one proper component and one improper component.

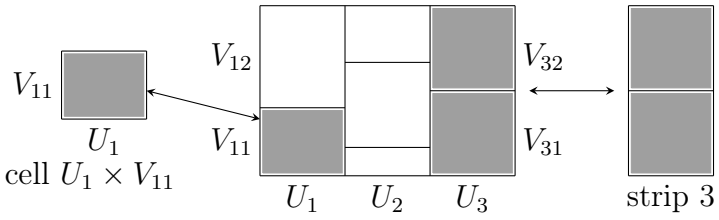

Figure 2.1: Partitions, Strips and Cells for Minimax

The algorithm is derived from Theorem 2.3.0.1.

Theorem 2.3.0.1. (Sainz, Herrero, Armengol, Vehí [16], 2008) *Given a real continuous function f over $X = (U, V)$ in $\mathbb{R}^n$ and a partition of X, $\min\limits_{u \in U'} \max\limits_{v \in V'} f(x)$ belongs to the interval*

$$\left[Inf\Big(\bigvee_{i \in \{1,\cdots,r\}} \bigwedge_{k_i \in \{1_i,\cdots,s_i\}} Out(fR(U_i, \tilde{v}_{k_i})) \Big), Inf\Big(\bigvee_{i \in \{1,\cdots,r\}} \bigwedge_{k_i \in \{1_i,\cdots,s_i\}} Inn(fR(\tilde{u}_i, V_{k_i})) \Big) \right]$$

where $\tilde{u}_i$ is any point in U_i' and $\tilde{v}_{k_i}$ is any point in V_{k_i}', for example the midpoints.

The final interval mentioned in Theorem 2.3.0.1 can be calculated via three types of intervals: cell interval, strip interval, and final interval.

- Cell Interval: $[\underline{f_{ij}}, \overline{f_{ij}}] = [Inf(Out(fR(U_i, \tilde{v}_{k_i}))), Inf(Inn(fR(\tilde{u}_i, V_{k_i})))]$.
- Strip Interval: $[\underline{f_i}, \overline{f_i}] = [\max_{j\in\{1_i,\cdots,s_i\}} \underline{f_{ij}}, \max_{j\in\{1_i,\cdots,s_i\}} \overline{f_{ij}}]$.
- Final Interval: $[\underline{f}, \overline{f}] = [\min_{i\in\{1,\cdots,r\}} \underline{f_i}, \min_{i\in\{1,\cdots,r\}} \overline{f_i}]$.

An example of partition at a certain step with possible further bisection choices as dashed lines for Sainz's minimax algorithm is given in Figure 2.2. Note that the vertical dashed line bisects strip 3 while the horizontal dashed line bisects cell $U_1 \times V_{12}$.

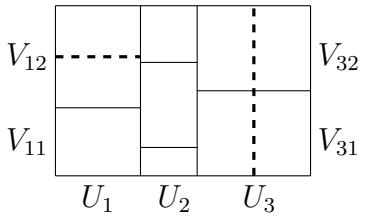

Figure 2.2: Possible bisection choices for minimax algorithm

Sainz's minimax algorithm is now outlined in Algorithm 1.

Algorithm 1 Sainz's Minimax Algorithm

1: Form an initial partition of domain through strips and cells.
2: Set initial cell intervals, strip intervals, and final interval.
3: Bisect a component according to your choice to get a finer partition.
4: Update cell intervals, strip intervals, and final interval.
5: Apply deletion check.
6: Apply termination check.
7: End with final interval output.

CHAPTER 3

NEW ALGORITHMS

3.1 Dual Problem of Minimax

While studying the properties of minimax problems, the dual problem of minimax (called maximin problem) came into consideration. We first introduce the definitions of strip and cell for maximin problems. Let $X = (U, V)$ where U and V are proper and improper components, respectively. Let $\{V_1, V_2, \cdots, V_t\}$ be a partition of V, and $\{U_{1_j}, U_{2_j}, \cdots, U_{t_j}\}$ a partition of U for every $j = 1, 2, \cdots, t$. Each interval $V_j \times U_{k_j}$ is called a **cell**, each $U_{*_j}-$partition of U is called a **strip**, and the $V-$partition is called the **strips' list**. Figure 3.1 shows a partition on X for maximin where X consists of only one proper component and one improper component.

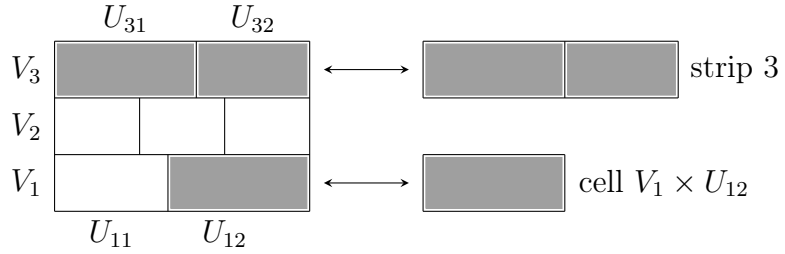

Figure 3.1: Partitions, Strips and Cells for Maximin

An example of maximin partition with possible further bisection choices as dashed lines is given in Figure 3.2 where strip 1 consists of cells $V_1 \times U_{11}$, $V_1 \times U_{12}$, and $V_1 \times U_{13}$. Note that the vertical dashed line bisects cell $V_2 \times U_{21}$ while the horizontal dashed line bisects strip 1.

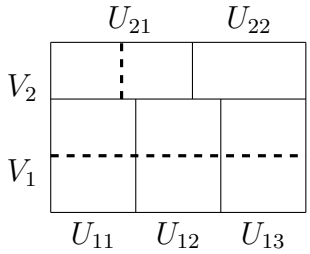

Figure 3.2: Possible bisection choices for maximin algorithm

To develop an algorithm to solve the dual problem, a new definition of semantic extensions is introduced as follows.

$$f^{\diamond}(X) = \bigvee_{x_i \in X_i'} \bigwedge_{x_p \in X_p'} [f(x_p, x_i), f(x_p, x_i)] = [\min_{x_i \in X_i'} \max_{x_p \in X_p'} f(x_p, x_i), \max_{x_i \in X_i'} \min_{x_p \in X_p'} f(x_p, x_i)],$$

$$f^{\diamond\diamond}(X) = \bigwedge_{x_p \in X_p'} \bigvee_{x_i \in X_i'} [f(x_p, x_i), f(x_p, x_i)] = [\max_{x_p \in X_p'} \min_{x_i \in X_i'} f(x_p, x_i), \min_{x_p \in X_p'} \max_{x_i \in X_i'} f(x_p, x_i)].$$

Here again, $x = (x_p, x_i)$ is the component split corresponding to the proper and improper components of $X = (X_p, X_i)$. It is easy to verify that the new semantic extension maintains the inclusion $f^{\diamond}(X) \subseteq f^{\diamond\diamond}(X)$. Similar to Theorem 2.3.0.1, we prove Theorem 3.1.0.2 that approximates the maximin problem.

Theorem 3.1.0.2. *Given a real continuous function f over $X = (U, V)$ in $\mathbb{R}^n$ and a partition of X, $\max_{v \in V'} \min_{u \in U'} f(x)$ belongs to the interval*

$$\left[Sup\Big(\bigvee_{j \in \{1, \cdots, r\}} \bigwedge_{k_j \in \{1_j, \cdots, s_j\}} Inn(fR(U_{k_j}, \tilde{v}_j)) \Big), Sup\Big(\bigvee_{j \in \{1, \cdots, r\}} \bigwedge_{k_j \in \{1_j, \cdots, s_j\}} Out(fR(\tilde{u}_{k_j}, V_j)) \Big) \right]$$

where $\tilde{v}_j$ is any point in V_j' and $\tilde{u}_{k_j}$ is any point in U_{k_j}', for example the midpoints.

Proof The new definition of semantic extensions shows that

$$\begin{aligned}\max_{v\in V'}\min_{u\in U'} f(x) = Sup(f^{\diamond}(X)) &= Sup(\bigvee_{v\in V'}\bigwedge_{u\in U'}[f(u,v)])\\ &= Sup(\bigvee_{j\in\{1,\cdots,r\}}\bigvee_{v_j\in V'_j}\bigwedge_{u\in U'}[f(u,v_j)])\\ &= Sup(\bigvee_{j\in\{1,\cdots,r\}}\bigvee_{v_j\in V'_j}\bigwedge_{k_j\in\{1_j,\cdots,s_j\}}\bigwedge_{u_{k_j}\in U'_{k_j}}[f(u_{k_j},v_j)]).\end{aligned}$$

As

$$\begin{aligned}f^{\diamond}(U_{k_j},v_j) = \bigvee_{v_j}\bigwedge_{u_{k_j}\in U_{k_j}}[f(u_{k_j},v_j)] &= [\min_{v_j}\max_{u_{k_j}\in U_{k_j}} f(u_{k_j},v_j), \max_{v_j}\min_{u_{k_j}\in U_{k_j}} f(u_{k_j},v_j)]\\ &= [\max_{u_{k_j}\in U_{k_j}} f(u_{k_j},v_j), \min_{u_{k_j}\in U_{k_j}} f(u_{k_j},v_j)]\\ &\subseteq [f(u_{k_j},v_j)] = f^{\diamond}(u_{k_j},v_j)\end{aligned}$$

implies that

$$\begin{aligned}\bigvee_{j\in\{1,\cdots,r\}}\bigvee_{v_j\in V'_j}\bigwedge_{k_j\in\{1_j,\cdots,s_j\}}\bigwedge_{u_{k_j}\in U'_{k_j}}[f(u_{k_j},v_j)] &\supseteq \bigvee_{j\in\{1,\cdots,r\}}\bigvee_{v_j\in V'_j}\bigwedge_{k_j\in\{1_j,\cdots,s_j\}} f^{\diamond}(U_{k_j},v_j)\\ &\supseteq \bigvee_{j\in\{1,\cdots,r\}}\bigwedge_{k_j\in\{1_j,\cdots,s_j\}} f^{\diamond}(U_{k_j},\tilde{v}_j)\\ &= \bigvee_{j\in\{1,\cdots,r\}}\bigwedge_{k_j\in\{1_j,\cdots,s_j\}} f^{\diamond\diamond}(U_{k_j},\tilde{v}_j)\\ &\supseteq \bigvee_{j\in\{1,\cdots,r\}}\bigwedge_{k_j\in\{1_j,\cdots,s_j\}} Inn(fR(U_{k_j},\tilde{v}_j)),\end{aligned}$$

we would obtain

$$Sup\Big(\bigvee_{j\in\{1,\cdots,r\}}\bigvee_{v_j\in V'_j}\bigwedge_{k_j\in\{1_j,\cdots,s_j\}}\bigwedge_{u_{k_j}\in U'_{k_j}}[f(u_{k_j},v_j)]\Big) \geq Sup\Big(\bigvee_{j\in\{1,\cdots,r\}}\bigwedge_{k_j\in\{1_j,\cdots,s_j\}} Inn(fR(U_{k_j},\tilde{v}_j))\Big).$$

Similarly, as

$$f^{\diamond}(u_{k_j}, V_j) = \bigvee_{v_j \in V_j} \bigwedge_{u_{k_j}} [f(u_{k_j}, v_j)] = [\min_{v_j \in V_j} \max_{u_{k_j}} f(u_{k_j}, v_j), \max_{v_j \in V_j} \min_{u_{k_j}} f(u_{k_j}, v_j)]$$

$$= [\min_{v_j \in V_j} f(u_{k_j}, v_j), \max_{v_j \in V_j} f(u_{k_j}, v_j)]$$

$$\supseteq [f(u_{k_j}, v_j)] = f^{\diamond}(u_{k_j}, v_j)$$

implies that

$$\bigvee_{j \in \{1,\cdots,r\}} \bigvee_{v_j \in V_j'} \bigwedge_{k_j \in \{1_j,\cdots,s_j\}} \bigwedge_{u_{k_j} \in U'_{k_j}} [f(u_{k_j}, v_j)] \subseteq \bigvee_{j \in \{1,\cdots,r\}} \bigwedge_{k_j \in \{1_j,\cdots,s_j\}} \bigwedge_{u_{k_j} \in U'_{k_j}} f^{\diamond}(u_{k_j}, V_j)$$

$$\subseteq \bigvee_{j \in \{1,\cdots,r\}} \bigwedge_{k_j \in \{1_j,\cdots,s_j\}} f^{\diamond}(\tilde{u}_{k_j}, V_j)$$

$$\subseteq \bigvee_{j \in \{1,\cdots,r\}} \bigwedge_{k_j \in \{1_j,\cdots,s_j\}} Out(fR(\tilde{u}_{k_j}, V_j)),$$

we would get the other inequality

$$Sup\Big(\bigvee_{j \in \{1,\cdots,r\}} \bigvee_{v_j \in V_j'} \bigwedge_{k_j \in \{1_j,\cdots,s_j\}} \bigwedge_{u_{k_j} \in U'_{k_j}} [f(u_{k_j}, v_j)] \Big) \leq Sup\Big(\bigvee_{j \in \{1,\cdots,r\}} \bigwedge_{k_j \in \{1_j,\cdots,s_j\}} Out(fR(\tilde{u}_{k_j}, V_j)) \Big).$$

Thus, our theorem holds. □

Three important types of intervals are used while implementing the maximin algorithm.

- Cell Interval: $[\underline{g_{ji}}, \overline{g_{ji}}] = [\text{Sup}(Inn(fR(U_{k_j}, \tilde{v}_j))), \text{Sup}(Out(fR(\tilde{u}_{k_j}, V_j)))]$.
- Strip Interval: $[\underline{g_j}, \overline{g_j}] = [\min_{i \in \{1_j,\cdots,s_j\}} \underline{g_{ji}}, \min_{i \in \{1_j,\cdots,s_j\}} \overline{g_{ji}}]$.
- Final Interval: $[\underline{g}, \overline{g}] = [\max_{j \in \{1,\cdots,r\}} \underline{g_j}, \max_{j \in \{1,\cdots,r\}} \overline{g_j}]$.

Our maximin algorithm is outlined in Algorithm 2.

Algorithm 2 Maximin Algorithm

1: Form an initial partition of domain through strips and cells.
2: Set initial cell intervals, strip intervals, and final interval.
3: Bisect a component according to your choice to get a finer partition.
4: Update cell intervals, strip intervals, and final interval.
5: Apply deletion check.
6: Apply termination check.
7: End with final interval output.

3.2 Separate Partition

The previous theorem provides a numerical method for solving maximin problems. Note that the statement

$$\min_{u\in U'}\max_{v\in V'} f(x) = \max_{v\in V'}\min_{u\in U'} f(x)$$

is not always true except for functions satisfying certain conditions. A basic result regarding their equivalence is stated in Theorem 3.2.0.3.

Theorem 3.2.0.3. (Sion [20], 1958) *Let X be a compact convex subset of a linear topological space and Y a compact convex subset of a linear topological space. If f is a real valued function on $X \times Y$ with $f(x,\cdot)$ upper semicontinuous and quasi-concave on Y, $\forall x \in X$, and $f(\cdot, y)$ lower semicontinuous and quasi-convex on X, $\forall y \in Y$, then*

$$\min_{x\in X}\sup_{y\in Y} f(x,y) = \sup_{y\in Y}\min_{x\in X} f(x,y).$$

Theorem 3.2.0.3 suggests that we may take advantage of both the original minimax problem and its dual problem while solving the minimax problem. For functions satisfying the conditions stated in Theorem 3.2.0.3, the minimax function value is equal to the maximin function value. Once cell intervals, strip intervals, and final intervals for both minimax and maximin are obtained, we introduce $fleft$ and $fright$ as $fleft = \max\{\underline{f}, \underline{g}\}$ and $fright = \min\{\overline{f}, \overline{g}\}$. It follows that we have a new final interval $[fleft, fright]$ to include the minimax function value, and the maximin value as well, and this final interval is better than both

$[\underline{f}, \overline{f}]$ and $[\underline{g}, \overline{g}]$. In this case, there are two separate and independent partitions going on for minimax and maximin, respectively. The only interaction between those two separate partitions happens at the step of updating $[fleft, fright]$. The idea leads to a new minimax algorithm illustrated in Algorithm 3.

Algorithm 3 Separate Partition Algorithm

1: Form an initial partition of domain for each of minimax and maximin.
2: Set initial cell intervals, strip intervals, and final intervals for each of minimax and maximin.
3: Bisect a selected component for each of minimax and maximin.
4: Update cell intervals, strip intervals, and final intervals for each of minimax and maximin.
5: Update $fleft$ and $fright$.
6: Apply deletion check for each of minimax and maximin.
7: Apply termination check.
8: End with final intervals output.

3.3 Uniform Partition

In Algorithm 3, minimax and maximin have its own partitions with distinct intervals. The only interaction of two partitions happens at updating $[fleft, fright]$. In order to take more advantage of minimax and maximin partitions, namely, more interactions, we propose a new type of partition, uniform partition as shown in Figure 3.3, where a partition in one direction is used uniformly in all other directions. Note that partition in one direction is not necessarily uniform. With this strategy, only one partition is needed and it works for both minimax and maximin. The bisection choices are illustrated in Figure 3.3. The other main difference, deletion check, should also be noticed. They are formally explained below.

Joint bisection: Possible bisection choices are graphed as dashed lines for minimax and maximin with separate partitions. As for the uniform partition, if you want to bisect a cell for minimax, bisect the corresponding strip for maximin as well. And if you want to bisect a cell for maximin, bisect the corresponding strip for minimax accordingly.

Joint deletion: When it comes to the deletion check, if a cell or strip is deleted due to minimax deletion conditions, we remove that cell or strip from the partition. Since there

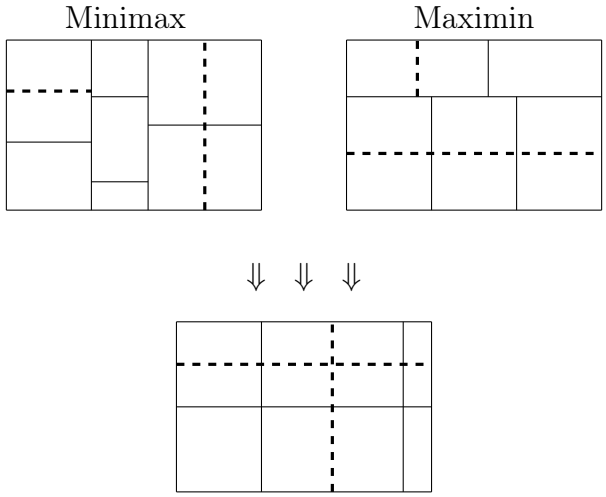

Figure 3.3: Uniform partition

is only one uniform partition, that cell or strip is also excluded for maximin. Likewisely, if a cell or strip is deleted due to maximin deletion conditions, that cell or strip is also excluded for minimax. In this way, we have a more efficient Algorithm 4.

Algorithm 4 Uniform Partition Algorithm

1: Form an initial uniform partition of domain through strips in both directions.
2: Set initial cell intervals, strip intervals, and final intervals for both minimax and maximin.
3: Bisect your choice of component via the joint bisection to get a finer partition.
4: Update cell intervals, strip intervals, and final intervals for both minimax and maximin.
5: Update $fleft$ and $fright$.
6: Apply deletion check for both minimax and maximin according to the joint deletion.
7: Apply termination check.
8: End with final interval output.

3.4 Necessary Conditions

3.4.1 Necessary Conditions Studied

This section presents necessary conditions studied theoretically for minimax problems. Necessary conditions would lead to deletion conditions in interval algorithms, namely, if a necessary condition is not satisfied on a subdomain, the subdomain should be removed from further consideration.

Schmitendorf (1977) proposed necessary conditions for static minimax problem [17].

The objective is to choose $x \in X \subseteq \mathbb{R}^n$ to minimize $f(x) = \sup_{y \in Y} \phi(x, y)$, where $\phi : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$ is continuously differentiable. $Y \subseteq \mathbb{R}^m$ is compact and $X = \{x | C(x) \leq 0\}$. The function $C : \mathbb{R}^n \to \mathbb{R}$ is continuously differentiable. For $x \in X$, define $I(x) = \{i | C_i(x) = 0\}$, and $\hat{Y}(x) = \{y \in Y | \phi(x, y) = \sup_{z \in Y} \phi(x, z)\}$. His necessary condition is presented in Theorem 3.4.1.1.

Theorem 3.4.1.1. (Schmitendorf [17], 1977) *Let x^* be a solution to the minimax problem. Then there exist a positive integer α, scalars $\lambda_i \geq 0$, $i = 1, \cdots, \alpha$, scalars $\mu_i \geq 0$, $i = 1, \cdots, p$, and vectors $y^i \in \hat{Y}(x^*)$, $i = 1, \cdots, \alpha$, such that*

$$\sum_{i=1}^{\alpha} \lambda_i \phi_x(x^*, y^*) + \sum_{i=1}^{p} \mu_i C_{ix}(x^*) = 0,$$

$$\mu_i C_i(x^*) = 0, \; i = 1, \cdots, p,$$

$$\sum_{i=1}^{\alpha} \lambda_i + \sum_{i=1}^{p} \mu_i \neq 0.$$

Also, if β is the number of nonzero μ_i, $1 \leq \alpha + \beta \leq n + 1$.

The existence of such positive integer α, scalars λ_i, and μ_i are confirmed in Theorem 3.4.1.1, however, the way to find those values is not provided, which increases the difficulty of applying Theorem 3.4.1.1 directly. Shimizu and Aiyoshi (1980) proposed similar necessary conditions for minimax problems [19].

Shen, Neumaier, and Eiermann (1990) mentioned a necessary condition for the minimax problem of a twice continuously differentiable function $f(x, y)$ [18]. Let $(x^*, y^*) \in (X, Y)$ be a minimax point of $f(x, y)$ in (X, Y) such that $f(x^*, y^*) = \min\limits_{x \in X} \max\limits_{y \in Y} f(x, y)$. Then $f(x^*, y^*) = \max\limits_{y \in Y} f(x^*, y)$ and either $y^* \in \{Inf(Y), Sup(Y)\}$, or

$$\frac{\partial f}{\partial y_i}(x^*, y^*) = 0 \text{ and } \frac{\partial^2 f}{\partial^2 y_i}(x^*, y^*) \leq 0, i = 1, 2, \cdots, m, \text{for interior point } y^* \in Y.$$

However, in real problems the minimax point (x^*, y^*) usually lie on the bound of Y, which

limits the use of Shen's necessary condition.

Other than above necessary conditions, Lai, Liu, and Tanaka (1999) constructed necessary conditions for minimax fractional programming [8]. Chen, Lai (2004) found optimality conditions for minimax programming of analytic functions [2].

3.4.2 New Deletion Conditions

Our goal is to find more necessary conditions that are easy to implement as deletion conditions in our algorithms. In addition to the base algorithms 1 and 2 presented earlier, we propose two deletion conditions for each algorithm. One deletion condition aims to delete strips while the other deals with cells. They are based on Theorems 3.4.2.1, 3.4.2.2, 3.4.2.3, and 3.4.2.4 presented below. We first introduce $f^* = \min\limits_{u\in U'} \max\limits_{v\in V'} f(x)$ and $g^* = \max\limits_{v\in V'} \min\limits_{u\in U'} f(x)$.

Theorem 3.4.2.1. *If $\underline{f_i} > \overline{f}$, then there is no minimax point in the* ***strip*** *(U_i, V).*

Proof Assume that $(u_1, v_1) \in (U_i, V)$ is a minimax point. Note that $\overline{f}$ is an upper bound of $\min\limits_{u\in U'} \max\limits_{v\in V'} f(u, v)$. Then

$$f^* = f(u_1, v_1) = \max_{v\in V'} f(u_1, v) = \max_j (\max_{v\in V'_{ij}} f(u_1, v)) \geq \max_j (f(u_1, midV_{ij}))$$

$$\geq \max_j (Inf(fR(U_i, midV_{ij}))) \geq \max_j (\underline{f_{ij}}) = \underline{f_i} > \overline{f} \geq f^*.$$

This contradiction happens because of the initial assumption. Therefore there is no minimax point in the strip (U_i, V). □

Theorem 3.4.2.2. *If $\min\limits_{u\in U_i} f(u, v) < \underline{f_i}$ for all $v \in V_{ij}$, then there is no minimax point in the* ***cell*** *(U_i, V_{ij}).*

Proof Assume that $(u_1, v_1) \in (U_i, V_{ij})$ is a minimax point. Note that $\underline{f_i}$ is a lower bound of $\max\limits_{v\in V'} f(u, v)$,$\forall u \in U_i$. Then

$$f(u_1, v_1) = \max_{v\in V'} f(u_1, v) \geq \underline{f_i}$$

and

$$f(u_1, v_1) = \min_{u \in U'} f(u, v_1) \leq \min_{u \in U_i} f(u, v_1).$$

It follows that $\min_{u \in U_i} f(u, v_1) \geq \underline{f_i}$. However, $\min_{u \in U_i} f(u, v_1) < \underline{f_i}$ since $v_1 \in V_{ij}$ and $\min_{u \in U_i} f(u, v) < \underline{f_i}$ for all $v \in V_{ij}$. So there is a contradiction. Therefore there is no minimax point in the cell (U_i, V_{ij}). □

Theorem 3.4.2.3. *If $\overline{g_j} < \underline{g}$, then there is no maximin point in the* ***strip*** (U, V_j).

Proof Assume that $(u_1, v_1) \in (U, V_j)$ is a maximin point. Note that $\underline{g}$ is a lower bound of $\max_{v \in V'} \min_{u \in U'} f(u, v)$. Then

$$g^* = f(u_1, v_1) = \min_{u \in U'} f(u, v_1) \leq \min_i (\min_{u \in U'_{ji}} f(u, v_1)) \leq \min_i (f(midU_{ji}, v_1))$$

$$\leq \min_i (Sup(fR(midU_{ji}, V_j))) \leq \min_i (\overline{g_{ji}}) = \overline{g_j} < \underline{g} \leq g^*.$$

This contradiction happens because of the initial assumption. Therefore there is no maximin point in the strip (U, V_j). □

Theorem 3.4.2.4. *If $\max_{v \in V_j} f(u, v) > \overline{g_j}$ for all $u \in U_{ji}$, then there is no maximin point in the* ***cell*** (U_{ji}, V_j).

Proof Assume that $(u_1, v_1) \in (U_{ji}, V_j)$ is a maximin point. Note that $\overline{g_j}$ is an upper bound of $\min_{u \in U'} f(u, v)$, $\forall v \in V_j$. Then

$$f(u_1, v_1) = \min_{u \in U'} f(u, v_1) \leq \overline{g_j}$$

and

$$\overline{g_j} \geq f(u_1, v_1) = \max_{v \in V'} f(u_1, v) \geq \max_{v \in V_j} f(u_1, v).$$

However, $\max_{v \in V_j} f(u_1, v) > \overline{g_j}$ since $u_1 \in U_{ji}$ and $\max_{v \in V_j} f(u, v) > \overline{g_j}$ for all $u \in U_{ji}$. So there is a contradiction. Therefore there is no maximin point in the cell (U_{ji}, V_j). □

The following deletion conditions are the direct consequences of above theorems, which can be implemented in our algorithms.

- Minimax Deletion Condition 1.1: $\underline{f_i} > \overline{f} \Longrightarrow$ Delete strip(U_i, V).
- Minimax Deletion Condition 1.2: $\overline{f_{ij}} < \underline{f_i} \Longrightarrow$ Delete cell(U_i, V_{ij}).
- Maximin Deletion Condition 2.1: $\overline{g_j} < \underline{g} \Longrightarrow$ Delete strip(U, V_j).
- Maximin Deletion Condition 2.2: $\underline{g_{ji}} > \overline{g_j} \Longrightarrow$ Delete cell(U_{ji}, V_j).

Termination conditions applied in algorithms are:

- $|fright - fleft| < \epsilon_1$ for some $\epsilon_1 > 0$.
- The length of the widest component of the boxes on the list is less than some $\epsilon_2 > 0$.

Other possible termination conditions include number of iterations, elapsed time, etc.

CHAPTER 4

FURTHER IMPROVEMENT TECHNIQUES IN OUR IMPLEMENTATION

4.1 Theoretical Techniques

Theoretical techniques are studied in terms of following aspects.

- Choice of inclusion functions

  A finer inclusion function will notably improve the performance of the algorithm. Programmable inclusion function forms include natural interval extension form, meanvalue form and Taylor form. Other inclusion functions can be found in [1,3,4,18,22].

- Deletion conditions

  Designs of deletion conditions are subject to the properties of the given objective function. For continuous functions, we proposed previous four conditions. More deletion conditions will be proposed given other properties of the function, such as Lipschitz continuity and differentiability.

4.2 Computing Techniques

These computational techniques have been studied.

- Multiple samples

  In previous algorithms, only one sample in each cell, namely, the midpoint sample, is used to obtain cell intervals, strip intervals, and final interval. Suppose more samples per cell are used in obtaining cell intervals. For each sample, a cell interval is calculated. The finer cell interval is obtained by taking the intersection of those cell intervals.

Therefore, We apply midpoint samples plus some endpoint samples instead of midpoint samples alone to the algorithm. By doing so, we obtain better cell intervals and thus better strip intervals and better final interval. Once this technique is applied in the previous algorithms, new versions are obtained. For instance, Algorithm 1.1 is obtained if the technique is applied in Algorithm 1.

- Recycled sampling

  Assume the original domain is (U, V). We apply midpoint samples plus some endpoint samples to calculate cell intervals. Recycled sampling is illustrated in Figure 4.1.

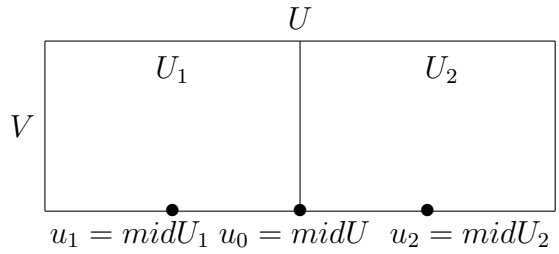

Figure 4.1: An example of recycled sampling

(u_0, V) where u_0 is the midpoint of U is a sample when calculating cell interval $U \times V$. This interval information is stored in memory. Assume U_1 and U_2 are obtained after bisection of U. To calculate cell interval $U_1 \times V$, we use both sample (u_1, V) where u_1 is the midpoint of U_1 and sample (u_0, V) where u_0 is the right endpoint of U_1 as samples. To calculate cell interval $U_2 \times V$, we use both sample (u_2, V) where u_2 is the midpoint of U_2 and sample (u_0, V) where u_0 is the left endpoint of U_2 as samples. That is to say, sample (u_0, V) is a recycled sample for cell intervals $U_1 \times V$ and $U_2 \times V$. So the stored interval information of cell interval $U \times V$ can be used without recalculation to estimate cell intervals $U_1 \times V$ and $U_2 \times V$. With the recycled sampling method, we are getting benefits of multiple samples with calculation of a single new sample. Once recycled sampling is applied in the previous algorithms, new versions are obtained. Namely, Algorithm 1.2 is obtained if the technique is applied in Algorithm 1.

CHAPTER 5

NUMERICAL RESULTS

5.1 Benchmark Problems

The following types of benchmark problems have been used to study the performance of the proposed algorithms.

TYPE I: Type I problems are used to test Algorithm 1 and Algorithm 2 separately.

Problem P_{11} [16]:

$$\min_{x\in[0,6]} \max_{y\in[2,8]} f(x,y) = \min_{x\in[0,6]} \max_{y\in[2,8]} x^2 + y^2 + 2xy - 20x - 20y + 100.$$

The minimax value is 9 at points $(5,2)$ and $(5,8)$ while the maximin value is 4 at point $(6,2)$.

Problem P_{12} [22]:

$$\min_{x\in[0,1]} \max_{y\in[0,1]} f(x,y) = \min_{x\in[0,1]} \max_{y\in[0,1]} (x-y)^2.$$

The minimax value is 0.25 at points $(\frac{1}{2},0)$ and $(\frac{1}{2},1)$. The maximin value is 0 at points (x,y) whenever $x = y$.

Problem P_{13}:

$$\min_{x\in[0,10]} \max_{y\in[0,10]} f(x,y) = \min_{x\in[0,10]} \max_{y\in[0,10]} \min\{3 - 0.2x + 0.3y, 3 + 0.2x - 0.1y\}.$$

The minimax value is 3 at point $(0,0)$ while the maximin value is 2.5 at $(0,5)$.

Problem P_{14} [18]:

$$\min_{x\in[0,1]} \max_{y\in[0,1]} f(x,y) = \min_{x\in[0,1]} \max_{y\in[0,1]} y(1-y)(y-x)^4.$$

The minimax value is $\frac{1}{432}$ at points $(\frac{1}{2}, \frac{1}{2}(1+\sqrt{\frac{2}{3}}))$ and $(\frac{1}{2}, \frac{1}{2}(1-\sqrt{\frac{2}{3}}))$ while the maximin value is 0 at points (x, y) whenever $x = y$.

TYPE II: Type II problems consist of 6 unconstrained minimax optimization problems. All objective functions satisfy the conditions stated in Theorem 3.2.0.3. That is, we have

$$\min_{x\in X}\max_{y\in Y} f(x,y) = \max_{y\in Y}\min_{x\in X} f(x,y)$$

for those 6 problems. Hence in addition to Algorithms 1 and 2, both Algorithm 3 and Algorithm 4 are tested as well.

Problem P_{21} [22]:

$$\min_{x\in[-1.4,1.4]}\max_{y\in[0,1]} f(x,y) = \min_{x\in[-1.4,1.4]}\max_{y\in[0,1]} x^2 + y.$$

The minimax value is 1 at point $(0, 1)$.

Problem P_{22} [1]:

$$\min_{x\in[-1,1]}\max_{y\in[-1,1]} f(x,y) = \min_{x\in[-1,1]}\max_{y\in[-1,1]} |x| - |y|.$$

The minimax value is 0 at point $(0, 0)$.

Problem P_{23}:

$$\min_{x\in[-1.4,2]^2}\max_{y\in[0,1]} f(x,y) = \min_{x\in[-1.4,2]^2}\max_{y\in[0,1]} 2x_1^2 + 2x_2^2 - x_1x_2 + y.$$

The minimax value is 1 at point $(0, 0, 1)$.

Problem P_{24}:

$$\min_{x\in X}\max_{y\in[0,10]} f(x,y) = \min_{x\in X}\max_{y\in[0,10]} (x_1-5)^2 - x_2(y-5)^2 \text{ where } X = [0,10]\times[1,10].$$

The minimax value is 0 at point $(5, \text{any } x_2, 5)$.

Problem P_{25}:

$$\min_{x\in X}\max_{y\in Y} f(x,y) = \min_{x\in X}\max_{y\in Y} 100(x_2 - x_1^2)^2 + (1-x_1)^2 - y_1(x_1 + x_2^2) - y_2(x_1^2 + x_2)$$

$$\text{where } X = [-0.5, 0.5] \times [0,1] \text{ and } Y = [0,10]^2.$$

The minimax value is 0.25 at point $(0.5, 0.25, 0, 0)$.

Problem P_{26}:

$$\min_{x\in X}\max_{y\in Y} f(x,y) = \min_{x\in X}\max_{y\in Y} 2x_1^2 + 2x_2^2 + 3x_3^2 - 2x_1x_2 - 2x_1x_3 + x_1 \sin y_1 + \cos y_2$$

$$\text{where } X = [0,10]^3 \text{ and } Y = [0, \pi/2]^2.$$

The minimax value is 1 at point $(0, 0, 0, \pi/2, 0)$.

5.2 Numerical Results

All algorithms will always stop after a finite number of iterations when a suitable termination condition is given. The numerical results for each test example are summarized in a table. Column 1 shows the algorithms used. Column 2 tells the output optimal final interval while column 3 tells the output optimal solution. Number of iterations and elapsed time are recorded in columns 4 and 5, respectively. Column 6 shows the total volume deleted by our deletion conditions. Column 7 explains due to what termination criterion the algorithm is terminated. The process of updating final interval at each step for an algorithm will be recorded in a figure. The x-axis of the figure is the number of iterations while each line segment on the y-axis represents the length of final interval at that iteration. All calculations are done in a personal computer with Intel Core i5-4590 3.30 GHz processor and 8GB of RAM.

5.2.1 Type I

Numerical results of Type I benchmark problems are obtained with $\epsilon_1 = 10^{-6}$ and $\epsilon_2 = 10^{-6}$, and recorded in Table 5.1, 5.2, 5.3, and 5.4. All values are subject to six decimal places. According to the numerical results, it requires fewer numbers of iterations as well as less elapsed time without affecting the accuracy of optimal solutions once an improvement technique is applied to Algorithm 1. The more improvement techniques are applied, the better performance the algorithm has. The last row in each table gives us a glimpse of the maximin function value, which is different from the minimax function value.

Problem P_{11} [16]:

$$\min_{x\in[0,6]} \max_{y\in[2,8]} f(x,y) = \min_{x\in[0,6]} \max_{y\in[2,8]} x^2 + y^2 + 2xy - 20x - 20y + 100.$$

The minimax value is 9 at points $(5,2)$ and $(5,8)$ while the maximin value is 4 at point $(6,2)$.

Table 5.1: Numerical Results of P_{11}

Alg.	Optimal Interval	Optimal Solution	No. of Iterations	Elapsed Time (ms)	Total Volume Deleted	Termination Criterion
Alg 1	[8.99998,9.00001]	(5,2),(5,8)	2742	2209	36	ϵ_2
Alg 1.1	[8.99998,9.00001]	(5,2),(5,8)	2581	2060	36	ϵ_2
Alg 1.2	[8.99998,9.00001]	(5,2),(5,8)	2460	1090	36	ϵ_2
Alg 2.2	[3.99999,4.00001]	(5.99999,2)	2016	1861	36	ϵ_2

In Table 5.1, all four algorithms tested converge to corresponding final value within a finite number of iterations. Fewer number of iterations and less time needed for algorithm 1.2 because of improvement techniques applied. The process of updating final interval at each step for problem P_{11} by algorithm 1 and 1.2 are given in Figure 5.1. Recall that the x-axis is the number of iterations and each vertical line segment on the y-axis represents the length of final interval at that iteration. From Figure 5.1 we can see that both algorithms almost converge to a single value around iteration 500. To better compare the performance

of algorithms 1 and 1.2, we graph their final interval information up to iteration 500 in Figure 5.2. Algorithm 1.2 performs slightly better than algorithm 1 for problem P_{11}.

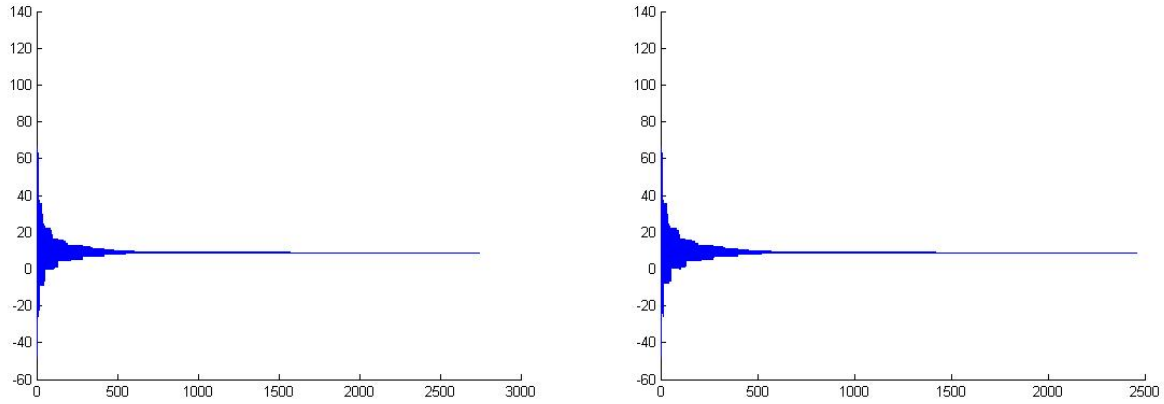

Figure 5.1: The process of updating final intervals for P_{11} using Alg. 1 (left) and Alg. 1.2 (right)

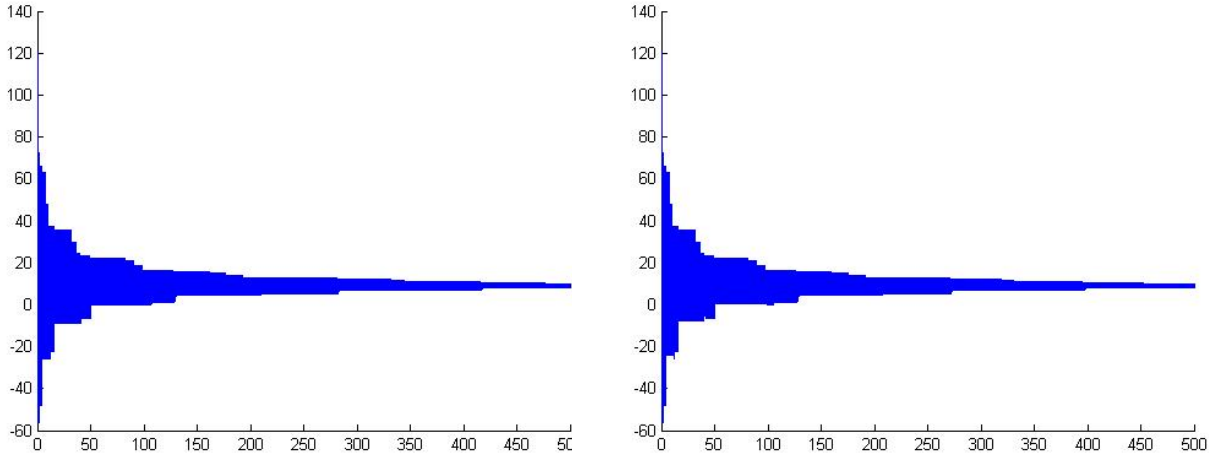

Figure 5.2: The process of updating final intervals within 500 iterations for P_{11} using Alg. 1 (left) and Alg. 1.2 (right)

Problem P_{12} [22]:

$$\min_{x\in[0,1]} \max_{y\in[0,1]} f(x,y) = \min_{x\in[0,1]} \max_{y\in[0,1]} (x-y)^2.$$

The minimax value is 0.25 at points $(\frac{1}{2}, 0)$ and $(\frac{1}{2}, 1)$. The maximin value is 0 at points (x, y)

whenever $x = y$.

Table 5.2: Numerical Results of P_{12}

Alg.	Optimal Interval	Optimal Solution	No. of Iterations	Elapsed Time (ms)	Total Volume Deleted	Termination Criterion
Alg 1	[0.25,0.25]	(0.499999,1), (0.499999,0.000001)	172	110	1	ϵ_1
Alg 1.1	[0.25,0.25]	(0.499999,1), (0.499999,0.000001)	113	71	1	ϵ_1
Alg 1.2	[0.249999, 0.25]	(0.5,0.999999), (0.5,0.000001)	96	49	1	ϵ_1
Alg 2.2	[0, 0]	any (x, y) where $x = y$	1534	12768	0.996096	ϵ_1

The process of updating final interval at each step for problem P_{12} by algorithm 1 and 1.2 are given in Figure 5.3.

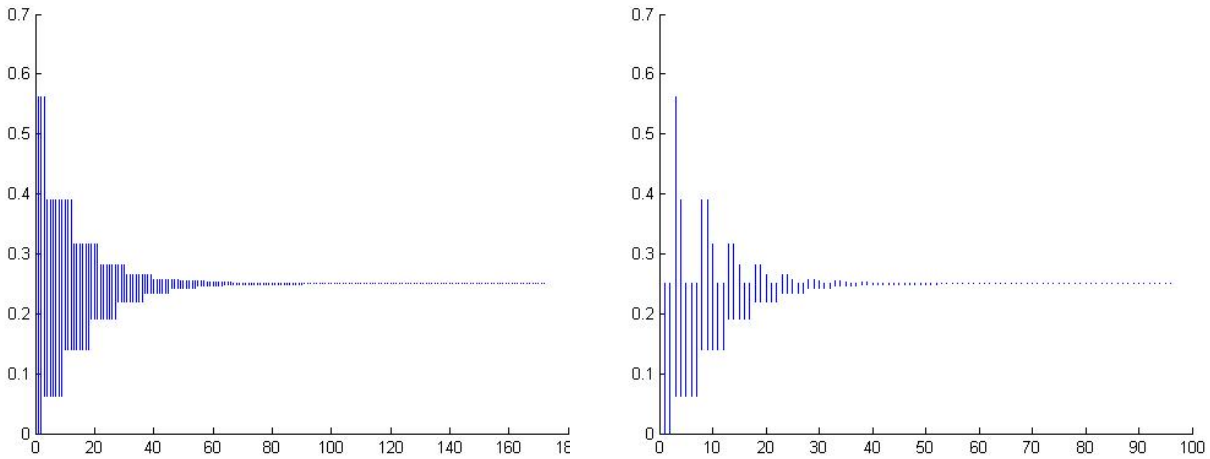

Figure 5.3: The process of updating final intervals for P_{12} using Alg. 1 (left) and Alg. 1.2 (right)

Table 5.2 tells us that algorithm 1.2 takes fewer number of iterations (96 iterations) to stop. To better compare the performance of algorithms 1 and 1.2, we graph their final interval information up to iteration 96 in Figure 5.4. Figure 5.4 confirms that algorithm 1.2 converges faster than algorithm 1 for problem P_{12}.

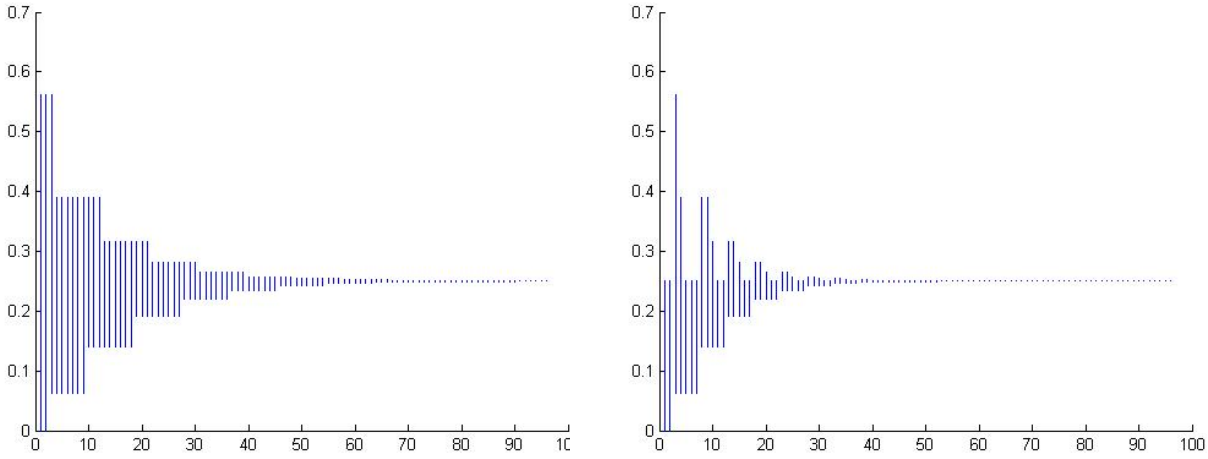

Figure 5.4: The process of updating final intervals within 96 iterations for P_{12} using Alg. 1 (left) and Alg. 1.2 (right)

Problem P_{13}:

$$\min_{x\in[0,10]} \max_{y\in[0,10]} f(x,y) = \min_{x\in[0,10]} \max_{y\in[0,10]} \min\{3-0.2x+0.3y, 3+0.2x-0.1y\}.$$

The minimax value is 3 at point $(0,0)$ while the maximin value is 2.5 at $(0,5)$.

Table 5.3: Numerical Results of P_{13}

Alg.	Optimal Interval	Optimal Solution	No. of Iterations	Elapsed Time (ms)	Total Volume Deleted	Termination Criterion
Alg 1	[3, 3]	(0.00001,0.00007)	221	112	100	ϵ_1
Alg 1.1	[3, 3]	(0.000001,0.000002)	182	101	100	ϵ_1
Alg 1.2	[3, 3]	(0.000001,0.000002)	182	65	100	ϵ_1
Alg 2.2	[2.5,2.5]	(0, 5)	95	61	100	ϵ_1

The process of updating final interval at each step for problem P_{13} by algorithm 1 and 1.2 are given in Figure 5.5. From Figure 5.5 we can see that both algorithms almost converge to a single value around iteration 60. To better compare the performance of algorithms 1 and 1.2, we graph their final interval information up to iteration 60 in Figure 5.6. Figure 5.6 confirms that algorithm 1.2 converges faster than algorithm 1 for problem P_{13}.

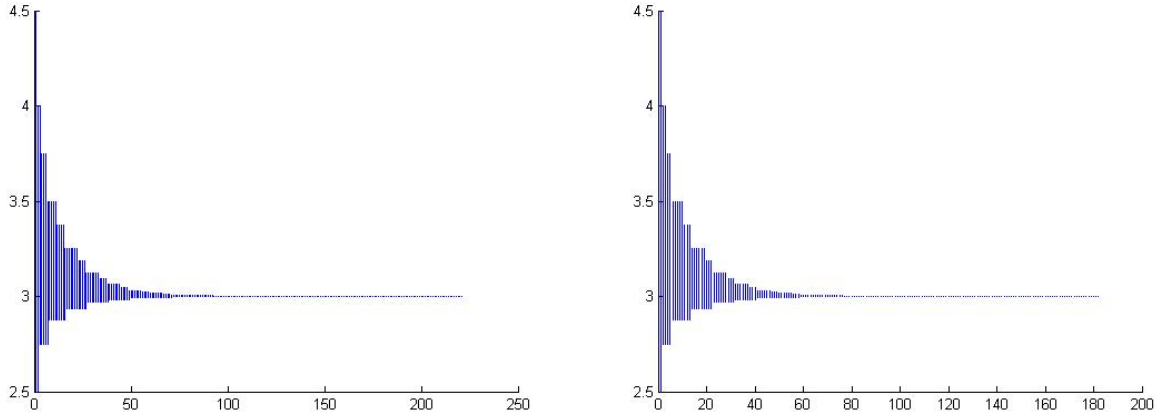

Figure 5.5: The process of updating final intervals for P_{13} using Alg. 1 (left) and Alg. 1.2 (right)

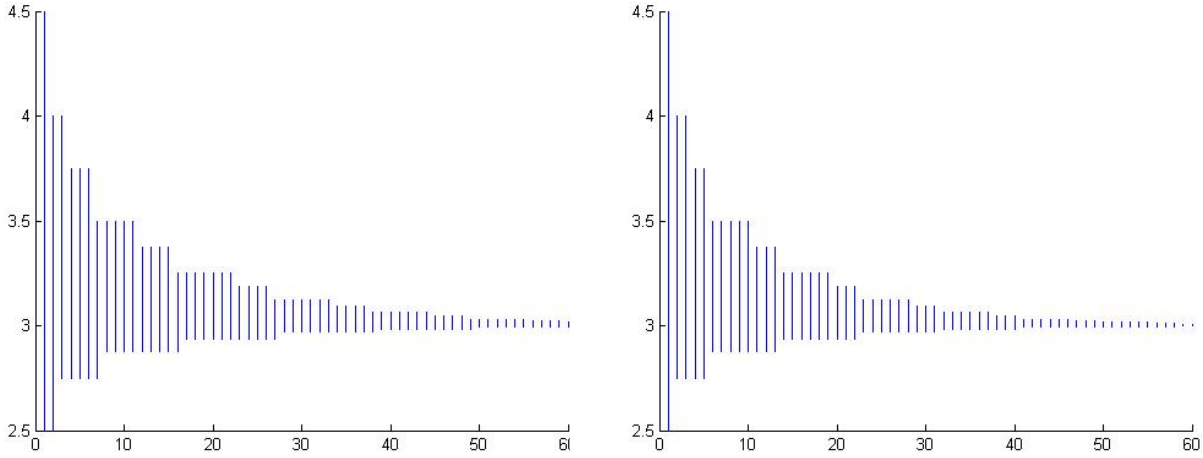

Figure 5.6: The process of updating final intervals within 60 iterations for P_{13} using Alg. 1 (left) and Alg. 1.2 (right)

Problem P_{14} [18]:

$$\min_{x\in[0,1]} \max_{y\in[0,1]} f(x,y) = \min_{x\in[0,1]} \max_{y\in[0,1]} y(1-y)(y-x)^4.$$

The minimax value is $\frac{1}{432}$ at points $(\frac{1}{2}, \frac{1}{2}(1+\sqrt{\frac{2}{3}}))$ and $(\frac{1}{2}, \frac{1}{2}(1-\sqrt{\frac{2}{3}}))$ while the maximin value is 0 at points (x, y) whenever $x = y$.

Table 5.4: Numerical Results of P_{14}

Alg.	Optimal Interval	Optimal Solution	No. of Iterations	Elapsed Time (ms)	Total Volume Deleted	Termination Criterion
Alg 1	[0.002315, 0.002316]	(0.499962,0.910599), (0.500046,0.0891876)	2438	2219	0.999999	ϵ_1
Alg 1.1	[0.002315, 0.002316]	(0.499992,0.905869), (0.500046,0.0891876)	2106	2074	0.999999	ϵ_1
Alg 1.2	[0.002315, 0.002316]	(0.499962,0.910599), (0.500046,0.0891876)	1971	1533	0.999999	ϵ_1
Alg 2.2	[0, 0]	$x = y$	44	47	0.875	ϵ_1

The process of updating final interval at each step for problem P_{14} by algorithm 1 and 1.2 are given in Figure 5.7.

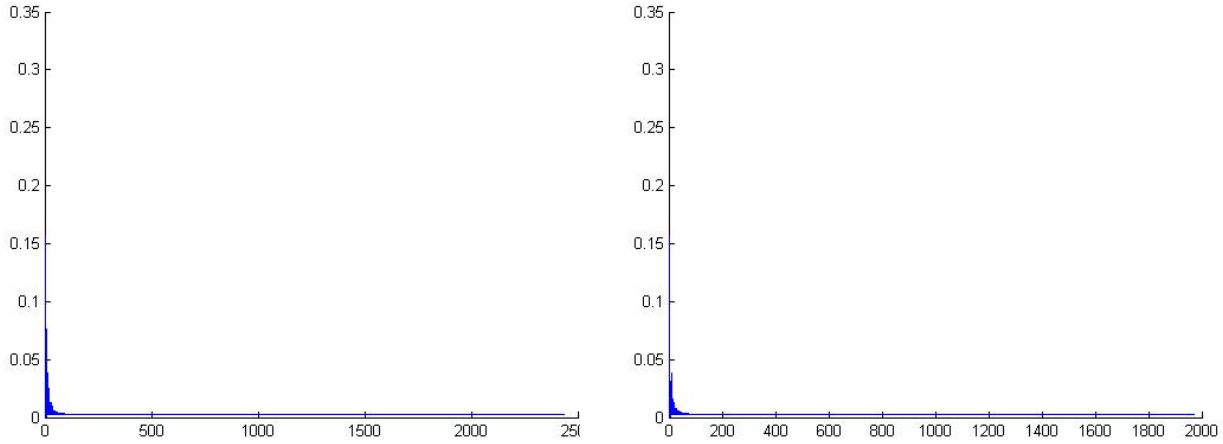

Figure 5.7: The process of updating final intervals for P_{14} using Alg. 1 (left) and Alg. 4.2 (right)

From Figure 5.7 we can see that both algorithms converge at a very early stage. To better compare the performance of algorithms 1 and 1.2, we graph their final interval information up to iteration 200 in Figure 5.8.

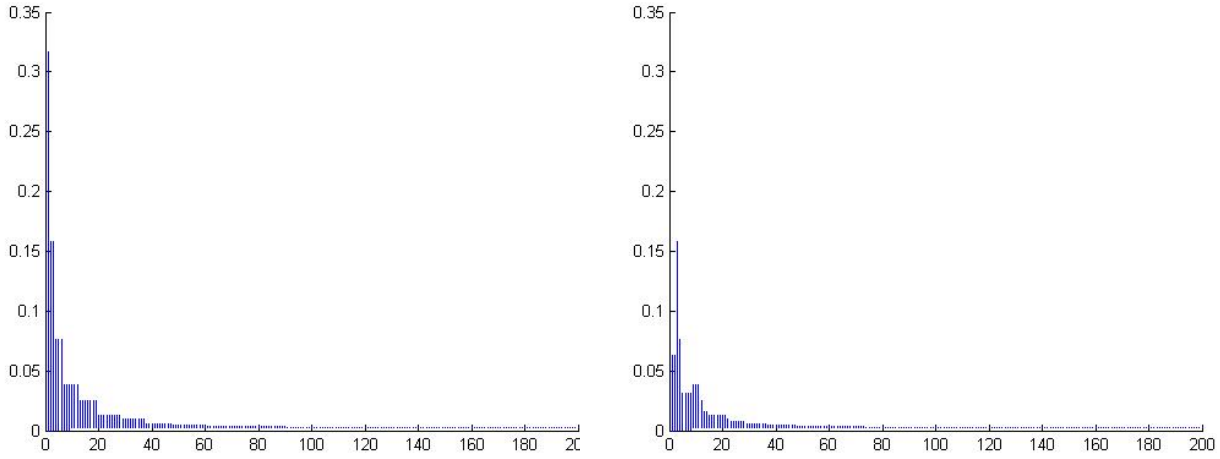

Figure 5.8: The process of updating final intervals within 200 iterations for P_{14} using Alg. 1 (left) and Alg. 4.2 (right)

5.2.2 Type II

For Type II benchmark problems, the minimax value is equal to the maximin value, so all four algorithms are tested and compared. Numerical results of Type II benchmark problems $P_{21} - P_{22}$ are obtained with $\epsilon_1 = 10^{-6}$ and $\epsilon_2 = 10^{-6}$, and recorded in Table 5.5 and 5.6. All values are subject to six decimal places. The results verify that the same optimal values as well as optimal solutions for each problem are obtained by each algorithm. Improvement techniques have a positive effect on those algorithms. Moreover, Algorithm 3 performs better than Algorithm 1 and Algorithm 2. Algorithm 4 has the best performance among all four algorithms, which confirms that the algorithm of uniform partition works better than the existing Sainz's algorithm.

Type II benchmark problems $P_{23} - P_{26}$ deal with minimax problems with 3 or more variables. In order to compare the ultimate performance of Algorithm 1 and our uniform partition Algorithm 4, numerical results of Algorithm 1.2 and Algorithm 4.2 are obtained with $\epsilon_1 = 10^{-6}$ and $\epsilon_2 = 10^{-6}$, and recorded in Table 5.7, 5.8, 5.9, and 5.10. For the increased dimensions, the tolerance values may be raised so that numerical experiments would terminate in a reasonable amount of time without using more advanced computers.

All values are subject to six decimal places. Numerical results of TYPE II benchmark problems $P_{23} - P_{26}$ show that the uniform partition algorithm significantly improves the performance in terms of number of iterations as well as elapsed time when the dimension of problems is increased.

Problem P_{21} [22]:

$$\min_{x\in[-1.4,1.4]} \max_{y\in[0,1]} f(x,y) = \min_{x\in[-1.4,1.4]} \max_{y\in[0,1]} x^2 + y.$$

The minimax value is 1 at point $(0, 1)$.

Table 5.5: Numerical Results of P_{21}

Alg.	Optimal Interval	Optimal Solution	No. of Iterations	Elapsed Time (ms)	Total Volume Deleted	Termination Criterion
Alg 1	[0.999999, 1]	(-0.000976, 0.999999)	4266	41361	3.4	ϵ_1
Alg 1.1	[1, 1]	(-0.000073, 0.999512)	26	65	3.4	ϵ_1
Alg 1.2	[1, 1]	(-0.000073, 0.999512)	26	55	3.4	ϵ_1
Alg 2	[0.999999, 1]	(0.000869, 0.999999)	1492	3597	3.4	ϵ_1
Alg 2.1	[0.999999, 1]	(0.000869, 0.999999)	1492	3803	3.4	ϵ_1
Alg 2.2	[0.999999, 1]	(0.000869, 0.999999)	1492	3751	3.4	ϵ_1
Alg 3	[0.999999, 1]	(-0.002758, 0.999992)	1492	7366	3.4	ϵ_1
Alg 3.1	[1, 1]	(-0.000073, 0.999512)	26	82	3.4	ϵ_1
Alg 3.2	[1, 1]	(-0.000073, 0.999512)	26	62	3.4	ϵ_1
Alg 4	[0.999999, 1]	(-0.000976, 0.999999)	1492	16742	3.4	ϵ_1
Alg 4.1	[1, 1]	(-0.000281, 0.999756)	26	97	3.4	ϵ_1
Alg 4.2	[1, 1]	(-0.000281, 0.999756)	26	54	3.4	ϵ_1

The process of updating final interval at each step for problem P_{21} by algorithm 1 and 4.2 are given in Figure 5.9. From Table 5.5 and Figure 5.9 we know that algorithm 4.2 already stop at iteration 26. To better compare the performance of algorithms 1 and 4.2, we graph their final interval information up to iteration 26 in Figure 5.10. Figure 5.10 confirms that algorithm 4.2 converges faster than algorithm 1 for problem P_{21}.

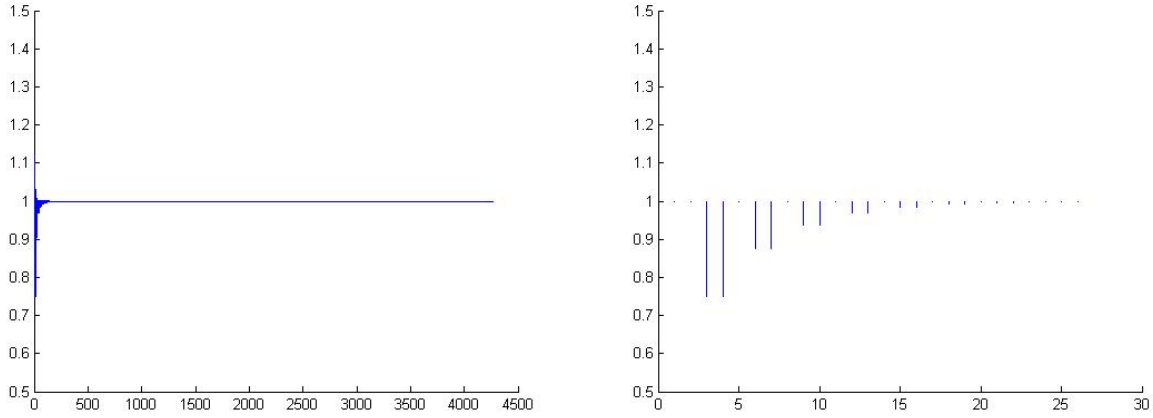

Figure 5.9: The process of updating final intervals for P_{21} using Alg. 1 (left) and Alg. 4.2 (right)

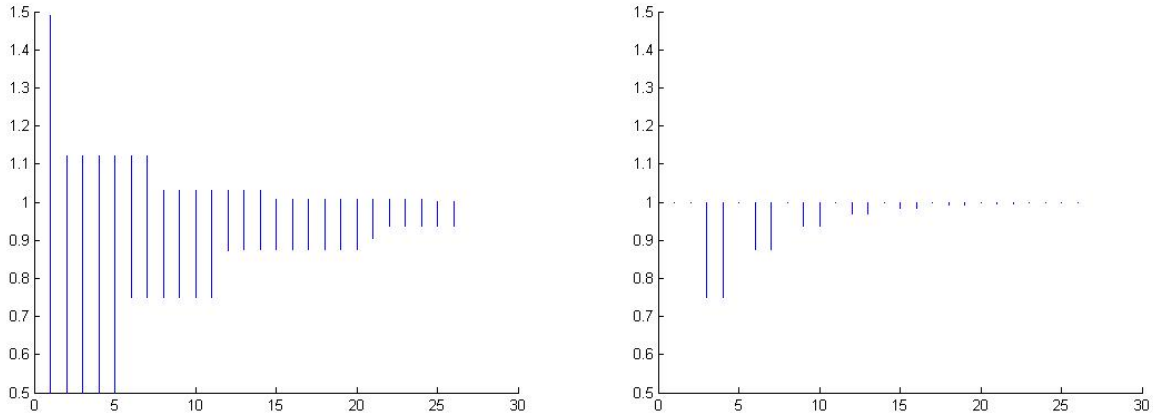

Figure 5.10: The process of updating final intervals within 26 iterations for P_{21} using Alg. 1 (left) and Alg. 4.2 (right)

Problem P_{22} [1]:

$$\min_{x\in[-1,1]} \max_{y\in[-1,1]} f(x,y) = \min_{x\in[-1,1]} \max_{y\in[-1,1]} |x| - |y|.$$

The minimax value is 0 at point $(0, 0)$.

Table 5.6: Numerical Results of P_{22}

Alg.	Optimal Interval	Optimal Solution	No. of Iterations	Elapsed Time (ms)	Total Volume Deleted	Termination Criterion
Alg 1	[0, 0]	(0, 0)	95	74	4	ϵ_1
Alg 1.1	[0, 0]	(0, 0)	77	62	4	ϵ_1
Alg 1.2	[0, 0]	(0, 0)	77	62	4	ϵ_1
Alg 2	[0, 0]	(0, 0)	91	188	4	ϵ_1
Alg 2.1	[0, 0]	(0, 0)	76	146	4	ϵ_1
Alg 2.2	[0, 0]	(0, 0)	76	142	4	ϵ_1
Alg 3	[0, 0]	(0,0.000005)	88	99	4	ϵ_1
Alg 3.1	[0, 0]	(0,0.000005)	74	93	4	ϵ_1
Alg 3.2	[0, 0]	(0,0.000005)	74	87	4	ϵ_1
Alg 4	[0, 0]	(0, 0)	54	173	4	ϵ_1
Alg 4.1	[0, 0]	(0, 0)	54	81	4	ϵ_1
Alg 4.2	[0, 0]	(0, 0)	54	76	4	ϵ_1

The process of updating final interval at each step for problem P_{22} by algorithm 1 and 4.2 are given in Figure 5.11. Table 5.6 and Figure 5.11 tell us that algorithm 4.2 takes fewer number of iterations (54 iterations) to stop. Therefore, we graph their final interval information up to iteration 54 in Figure 5.12 to better compare the performance of algorithms 1 and 4.2. Based on Figure 5.12, algorithm 4.2 has a faster speed of convergence than algorithm 1 for problem P_{22}.

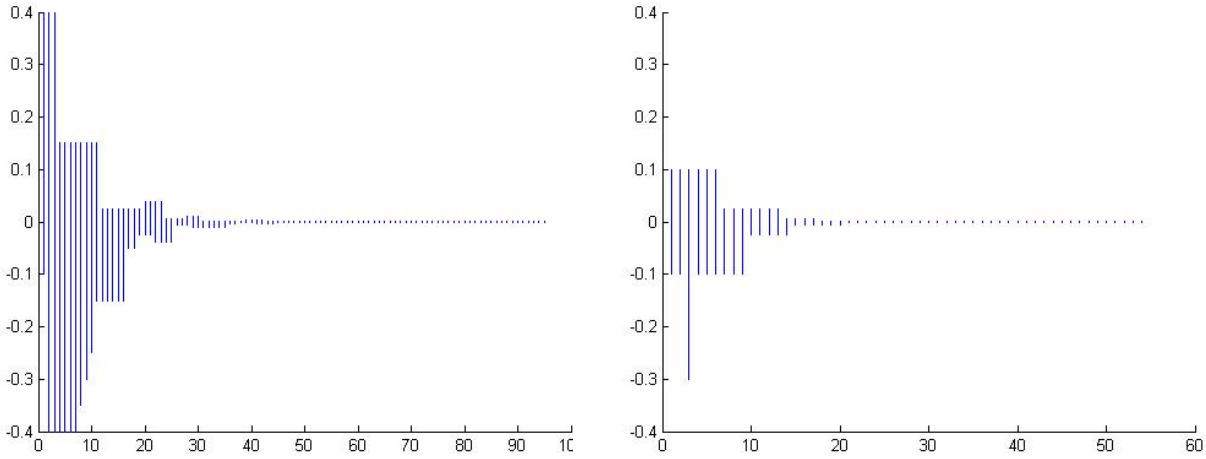

Figure 5.11: The process of updating final intervals for P_{22} using Alg. 1 (left) and Alg. 4.2 (right)

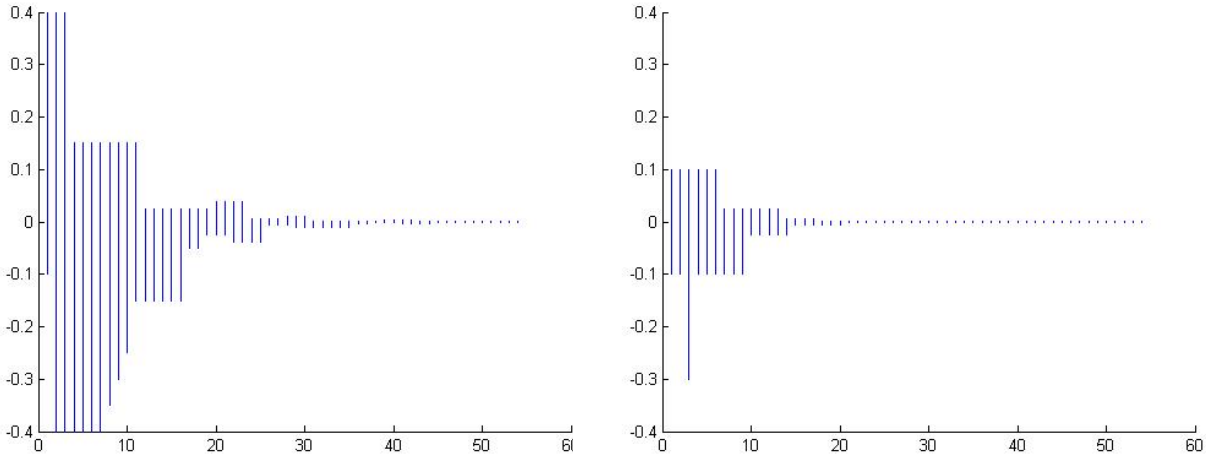

Figure 5.12: The process of updating final intervals within 54 iterations for P_{22} using Alg. 1 (left) and Alg. 4.2 (right)

Problem P_{23}:

$$\min_{x\in[-1.4,2]^2}\max_{y\in[0,1]} f(x,y) = \min_{x\in[-1.4,2]^2}\max_{y\in[0,1]} 2x_1^2 + 2x_2^2 - x_1x_2 + y.$$

The minimax value is 1 at point $(0, 0, 1)$.

Table 5.7: Numerical Results of P_{23}

Alg.	Optimal Interval	Optimal Solution	No. of Iterations	Elapsed Time (ms)	Total Volume Deleted	Termination Criterion
Alg 1.2	[0.999999,1]	(0,0,0.9995)	236	157	11.56	ϵ_1
Alg 4.2	[0.999999,1]	(0,0,0.9995)	176	165	11.56	ϵ_1

The process of updating final interval at each step for problem P_{23} by algorithm 1.2 and 4.2 are given in Figure 5.13.

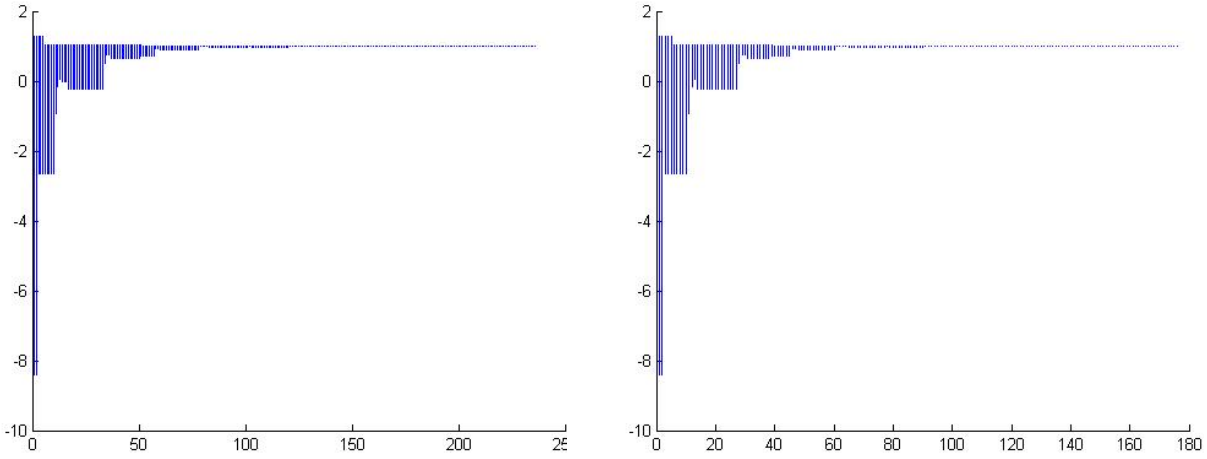

Figure 5.13: The process of updating final intervals for P_{23} using Alg. 1.2 (left) and Alg. 4.2 (right)

Table 5.7 tells us that algorithm 4.2 stops at iteration 176. From Figure 5.13 we can see that both algorithms almost converge to a single value around iteration 176. To better compare the performance of algorithms 1.2 and 4.2, we graph their final interval information up to iteration 176 in Figure 5.14. Based on Figure 5.14, algorithm 4.2 performs slightly better than algorithm 1 for problem P_{23}.

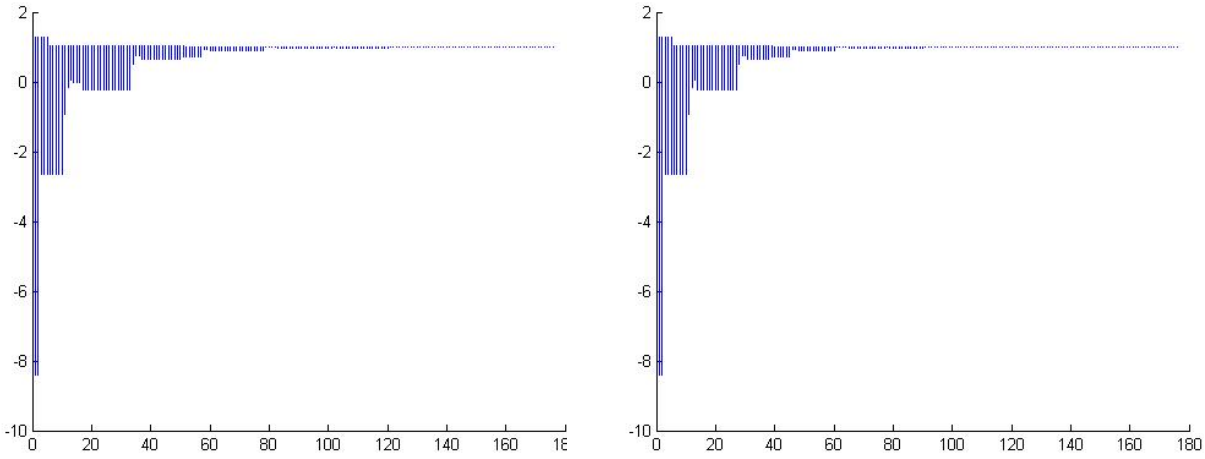

Figure 5.14: The process of updating final intervals within 176 iterations for P_{23} using Alg. 1.2 (left) and Alg. 4.2 (right)

Problem P_{24}:

$$\min_{x\in X}\max_{y\in[0,10]} f(x,y) = \min_{x\in X}\max_{y\in[0,10]} (x_1-5)^2 - x_2(y-5)^2 \text{ where } X=[0,10]\times[1,10].$$

The minimax value is 0 at point $(5, \text{any } x_2, 5)$.

Table 5.8: Numerical Results of P_{24}

Alg.	Optimal Interval	Optimal Solution	No. of Iterations	Elapsed Time (ms)	Total Volume Deleted	Termination Criterion
Alg 1.2	[0, 0]	(5.001,anyx_2,5.001)	18906	1091610	902.803	ϵ_1
Alg 4.2	[0, 0]	(5.001,anyx_2,5.001)	12441	974120	902.803	ϵ_1

The process of updating final interval at each step for problem P_{24} by algorithm 1.2 and 4.2 are given in Figure 5.15. From Figure 5.15 we can see that both algorithms converge at a very early stage. To better compare the performance of algorithms 1.2 and 4.2, we graph their final interval information up to iteration 100 in Figure 5.16.

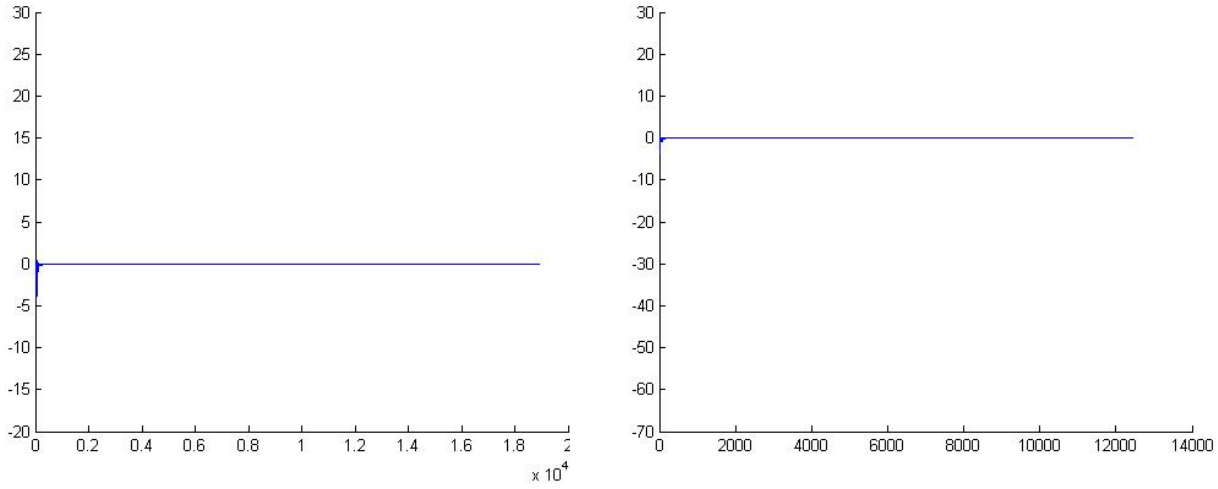

Figure 5.15: The process of updating final intervals for P_{24} using Alg. 1.2 (left) and Alg. 4.2 (right)

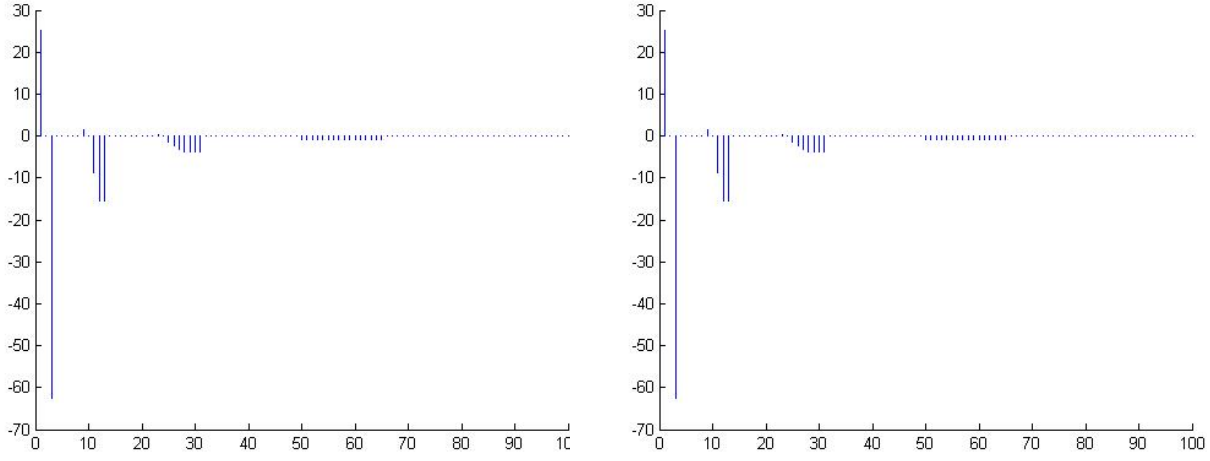

Figure 5.16: The process of updating final intervals within 100 iterations for P_{24} using Alg. 1.2 (left) and Alg. 4.2 (right)

Problem P_{25}:

$$\min_{x\in X}\max_{y\in Y} f(x,y) = \min_{x\in X}\max_{y\in Y} 100(x_2 - x_1^2)^2 + (1-x_1)^2 - y_1(x_1 + x_2^2) - y_2(x_1^2 + x_2)$$

$$\text{where } X = [-0.5, 0.5] \times [0,1] \text{ and } Y = [0,10]^2.$$

The minimax value is 0.25 at point $(0.5, 0.25, 0, 0)$.

Table 5.9: Numerical Results of P_{25}

Alg.	Optimal Interval	Optimal Solution	No. of Iterations	Elapsed Time (ms)	Total Volume Deleted	Termination Criterion
Alg 1.2	[0.249999, 0.25]	(0.5,0.25,0,0)	25744	225582	100	ϵ_1
Alg 4.2	[0.25, 0.250001]	(0.5,0.25,0,0)	4940	330029	100	ϵ_1

The process of updating final interval at each step for problem P_{25} by algorithm 1.2 and 4.2 are given in Figure 5.17. From Figure 5.17 we can see that both algorithms almost converge to a single value around iteration 4940. To better compare the performance of algorithms 1.2 and 4.2, we graph their final interval information up to iteration 4940 in Figure 5.18.

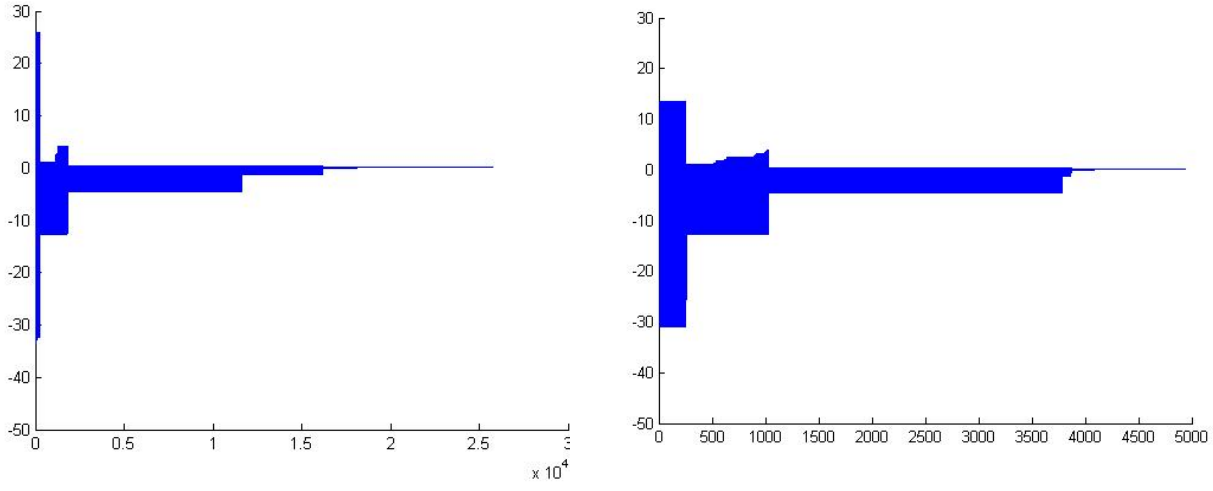

Figure 5.17: The process of updating final intervals for P_{25} using Alg. 1.2 (left) and Alg. 4.2 (right)

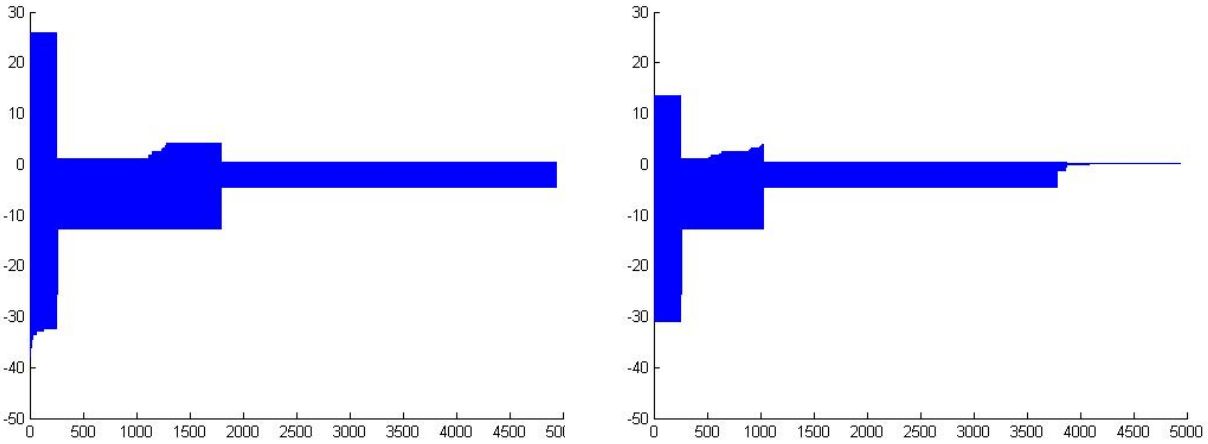

Figure 5.18: The process of updating final intervals within 4940 iterations for P_{25} using Alg. 1.2 (left) and Alg. 4.2 (right)

As shown in Figure 5.18, algorithm 4.2 performs better than algorithm 1.2 for problem P_{25}.

Problem P_{26}:

$$\min_{x\in X}\max_{y\in Y} f(x,y) = \min_{x\in X}\max_{y\in Y} 2x_1^2 + 2x_2^2 + 3x_3^2 - 2x_1x_2 - 2x_1x_3 + x_1\sin y_1 + \cos y_2$$

$$\text{where } X = [0,10]^3 \text{ and } Y = [0,\pi/2]^2.$$

The minimax value is 1 at point $(0,0,0,\pi/2,0)$.

Table 5.10: Numerical Results of P_{26}

Alg.	Optimal Interval	Optimal Solution	No. of Iters	Elapsed Time (ms)	Total Volume Deleted	Termination Criterion
Alg 1.2	[1, 1]	(0.000152,0.000152, 0.000152,1.28277, 0.002493)	211078	17333000	2468.04	ϵ_1
Alg 4.2	[1, 1]	(0.002441,0.002441, 0.002441,1.56793, 0.0153418)	3044	122697	2468.04	ϵ_1

As shown in Table 5.10 that algorithm 1.2 takes 211078 iterations to terminate while

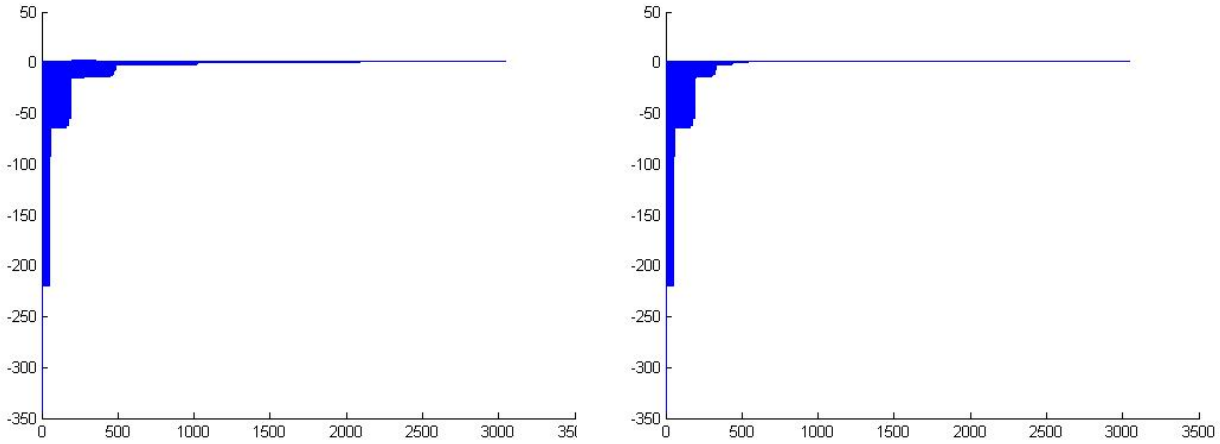

Figure 5.19: The process of updating final intervals within 3044 iterations for P_{26} using Alg. 1.2 (left) and Alg. 4.2 (right)

algorithm 4.2 takes 3044 iterations to terminate. We want to study the performance of two algorithms up to 3044 iterations. The process of updating final interval up to iteration 3044 for problem P_{26} by algorithm 1.2 and 4.2 are given in Figure 5.19, which confirms that algorithm 4.2 converges to minimax value faster than algorithm 1.2 for problem P_{26}.

CHAPTER 6

CONSTRAINED MINIMAX PROBLEMS

Constrained minimax problems are in the form

$$\min_{u \in U'} \max_{v \in V'} f(u, v)$$

$$s.t.\, g_r(u, v) \leq 0,\ r = 1, \cdots, m.$$

We cannot reply on simple interval computations to find constrained minimax value since the feasible region might not be an interval. Sainz [16] adapted their unconstrained minimax algorithm to the feasible region to approximate minimax solution. Their idea is presented in section 6.1.

6.1 Sainz's Constrained Minimax Algorithm

Let G_r be an inclusion function of g_r, $r = 1, \cdots, m$. Each cell $U_i \times V_{ij}$ of the partition will be included into one of the three types of regions according to the constraints:

- If $G_r(U_i, V_{ij}) \leq 0$ for all r, then cell $U_i \times V_{ij}$ is a member of region Π_1.
- If $G_r(U_i, V_{ij}) \geq 0$ for some r, then cell $U_i \times V_{ij}$ is a member of region Π_2.
- Otherwise, cell $U_i \times V_{ij}$ is a member of region Π_3.

It follows directly that region Π_1 is feasible, region Π_2 is not feasible, and region Π_3 needs further information for feasibility to be determined. If a cell is in region Π_2, it must be eliminated. If a cell is in region Π_3, it must be kept for subsequent partitions. Figure 6.1 shows an example of three types of regions in two dimensions. In Figure 6.1, the region

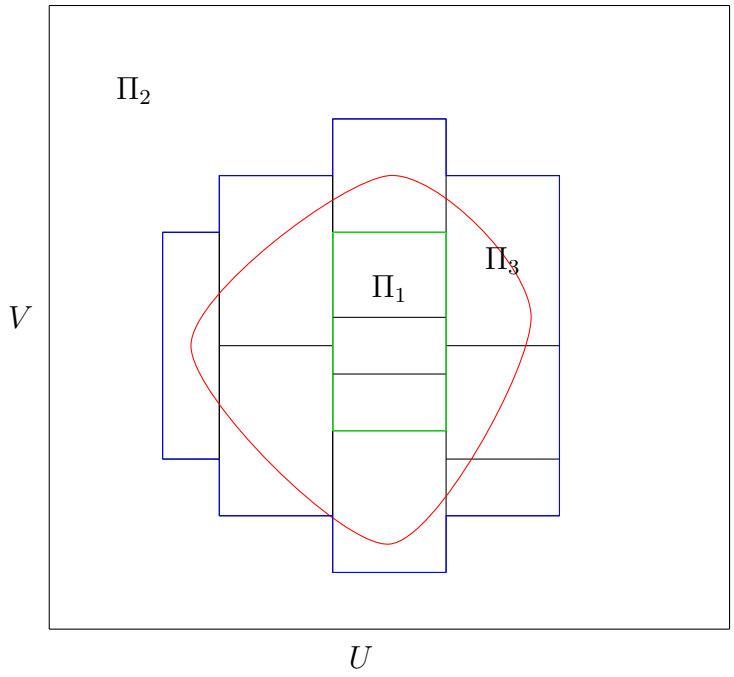

Figure 6.1: Feasibility region and partitions

outside of blue boundary is region Π_2 (not feasible), the region within green lines is region Π_1 (completely feasible), and the region between blue and green lines is region Π_3 (partially feasible). The region within red curve represents the actual feasible region.

For a partition Π_l, $l = 1, 3$, let $r^{(l)}$ be the number of strips and let $s_j^{(l)}$ be the number of cells of the strip j. In accordance with Theorem 2.3.0.1,

$$\min_{u\in U'}\max_{v\in V'} f(x) \leq Inf\left(\bigvee_{i\in\{1^{(1)},\cdots,r^{(1)}\}}\ \bigwedge_{k_i\in\{1_i^{(1)+(3)},\cdots,s_i^{(1)+(3)}\}} Inn(fR(\tilde{u}_i, V_{k_i}))\right)$$

and

$$\min_{u\in U'}\max_{v\in V'} f(x) \geq Inf\left(\left(\bigvee_{i\in\{1^{(1)},\cdots,r^{(1)}\}}\ \bigwedge_{k_i\in\{1_i^{(1)},\cdots,s_i^{(1)}\}} Out(fR(U_i, \tilde{v}_{k_i}))\right)\bigvee E\right),$$

where

$$E = \bigvee_{i\in\{1^{(3)},\cdots,r^{(3)}\}-\{1^{(1)},\cdots,r^{(1)}\}}\ \bigvee_{k_i\in\{1_i^{(3)},\cdots,s_i^{(3)}\}} Out(fR(U_i, Dual(V_{k_i}))).$$

The use of $\{1^{(3)}, \cdots, r^{(3)}\} - \{1^{(1)}, \cdots, r^{(1)}\}$ when defining E is somehow ambiguous. Sainz didn't mention how to calculate $\{1^{(3)}, \cdots, r^{(3)}\} - \{1^{(1)}, \cdots, r^{(1)}\}$ in their paper. Instead, we claim that

$$E = \bigvee_{i \in \{1^{(3)}, \cdots, r^{(3)}\}} \bigvee_{k_i \in \{1_i^{(3)}, \cdots, s_i^{(3)}\}} Out(fR(U_i, Dual(V_{k_i})))$$

and test with numerical experiments.

6.2 Numerical Results

The deletion conditions proposed in section 3.4.2 are not available for constrained minimax algorithm because of the feasible region. So the only deletion condition used is feasibility test. If a cell is in region Π_2, it must be eliminated. If a cell is in region Π_3, it must be kept for subsequent partitions. The algorithm will terminate when one of the termination conditions mentioned in section 3.4.2 is satisfied.

Problem P_{31} [16]:

$$\min_{x \in [0,6]} \max_{y \in [2,8]} f(x, y) = \min_{x \in [0,6]} \max_{y \in [2,8]} x^2 + y^2 + 2xy - 20x - 20y + 100,$$

subject to

$$g_1(x, y) = -(x-5)^2 - (y-3)^2 + 4 \leq 0,$$

$$g_2(x, y) = (x-5)^2 + (y-3)^2 - 16 \leq 0.$$

With $\epsilon_1 = 10^{-1}$ and $\epsilon_2 = 10^{-1}$, the result is

$$\min_{x \in [0,6]} \max_{y \in [2,8]} f(x, y) \in [-0.779053, 3.10571]$$

with a computation time of 1297 ms.

Problem P_{32} [16]:

$$\min_{x\in[-3.14,3.14]}\max_{y\in[-3.14,3.14]} f(x,y) = \min_{x\in[-3.14,3.14]}\max_{y\in[-3.14,3.14]}(\cos y + \cos(2y+x))^2,$$

subject to

$$g_1(x,y) = y - x(x+6.28) \leq 0,$$

$$g_2(x,y) = y - x(x-6.28) \leq 0.$$

With $\epsilon_1 = 10^{-4}$ and $\epsilon_2 = 10^{-2}$, the result is

$$\min_{x\in[-3.14,3.14]}\max_{y\in[-3.14,3.14]} f(x,y) \in [0.00893922, 0.0102298]$$

with a computation time of 492981 ms.

CHAPTER 7

CONCLUSIONS

Algorithms based on modal interval analysis have been applied to solving relatively simple problems on predictive control, parameter identification and some other engineering problems [15]. Numerical results confirmed that our algorithms have a better performance compared with the existing modal interval method. It is demonstrated to be very efficient for solving minimax problems of several variables. As Saniz pointed out that for some complex minimax problems, the interval method is not very efficient since the complexity significantly increases when the number of variables grows [16]. Our numerical results further confirmed the complexity issue for solving higher dimensional problems. Our future work will be focused on other applications of our minimax algorithms and solving constrained minimax problems.

REFERENCES

[1] Cao, D.X., Li, S.B., Wu, Y.Q.. Interval method for global solutions of continuous minimax problem. Numerical Mathematics A Journal of Chinese Universities, 24(4):359-365, 2002.

[2] Chen, J.C., Lai, H.C.. Optimality Conditions for Minimax Programming of Analytic Functions. Taiwanese Journal of Mathematics, 8(4):673-686, 2004.

[3] Goldsztejn, A.. Modal intervals revisited, part 1: A generalized interval natural extension. Reliable Computing, 16:130-183, 2012.

[4] Goldsztejn, A.. Modal intervals revisited, part 2: A generalized interval mean value extension. Reliable Computing, 16:184-209, 2012.

[5] Hansen, E.. Global Optimization Using Interval Analysis-the multi-dimensional case. Numerische Mathematik, 34(3):247-270, 1980.

[6] Lu, B., Cao, Y., Yuan, M.J., Zhou, J.. Reference Variable Methods of Solving Min-max Optimization Problems. Journal of Global Optimization, 42(1):1-21, 2008.

[7] Krawczyk, R., Nickel, K.. Die Zentrische Form in Der Intervallarithmetik, ihre quadratische Konvergenz und ihre Inklusionsisotonie. Computing, 28(2): 117-137, 1982.

[8] Lai, H.C., Liu, J.C., Tanaka, K.. Necessary and Sufficient Conditions for Minimax Fractional Programming. Journal of Mathematical Analysis and Applications, 230:311-328, 1999.

[9] Lu, S.W., Basar, T.. Genetic Algorithms-based Identification. IEEE International Conference on Systems, Man and Cybernetics, Intelligent Systems for the 21st Century, Oct. 22-25, 1995.

[10] Moore, R.E.. Interval Arithmetic and Automatic Error Analysis in Digital Computation. Ph.D. Thesis, Stanford University, 1962.

[11] Moore, R.E.. Interval Analysis. Prentice-Hall, Englewood Cliffs, NJ, 1966.

[12] Parpas, P., Rustem, B.. An Algorithm for the Global Optimization of a Class of Continuous Minimax Problems. J Optim Theory Appl, 141: 461-473, 2009.

[13] Ratschek, H., Rokne, J.. New Computer Methods for Global Optimization. Ellis Horwood Limited, London, 1988.

[14] Rustem, B., Howe, M.A.. Algorithms for Worst-case Design with Applications to Risk Management. Princeton University Press, Princeton, 2001.

[15] Sainz, M.A., Armengol, J., Calm, R., Herrero, P., Jorba, L., Vehi, J.. Modal Interval Analysis: New Tools for Numerical Information. Springer International Publishing, Switzerland, 2014.

[16] Sainz, M.A., Herrero, P., Armengol, J., Vehí, J.. Continuous minimax optimization using modal intervals. J. Math. Anal. Appl, 339:18-30, 2008.

[17] Schmitendorf, W.E.. Necessary Conditions and Sufficient Conditions for Static Minmax Problem. Journal of Mathematical Analysis and Application, 57:683-693, 1977.

[18] Shen, Z.H., Neumaier, A., Eiermann, M.C.. Solving minimax problems by interval methods. BIT, 30:742-751, 1990.

[19] Shimizu, K., Aiyoshi, E.. Necessary Conditions for Min-Max Problems and Algorithms by a Relaxation Procedure. IEEE Transactions on Automatic Control, 25(1):62-66, 1980.

[20] Sion, M.. On general minimax theorems. Pac. J. Math, 8(1):171-176, 1958.

[21] Storn, R., Price, K.. Differential Evolution-A Simple and Efficient Heuristic for global Optimization over Continuous Spaces. Journal of Global Optimization, 11(4):341-359, 1997.

[22] Wang, J., Cao, D.X.. Interval Algorithm for a Class of Continuous Minimax Problems. Computational Intelligence and Software Engineering (CiSE), Dec 10-12, 2010.