GLOBAL SOLUTION
TO PARAMETRIC COMPLEMENTARITY CONSTRAINED PROGRAMS
AND APPLICATIONS IN OPTIMAL PARAMETER SELECTION

BY

YU-CHING LEE

DISSERTATION
Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Industrial Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2012

Urbana, Illinois

Doctoral Committee:
Professor Jong-Shi Pang, Chair
Associate Professor Angelia Nedich
Associate Professor Ramavarapu S. Sreenivas
Associate Professor Che-Lin Su, The University of Chicago
Abstract

This thesis contains five chapters. The notations, terminologies, definitions and numbering of equations, theorems and algorithms are independent in each chapter. Chapter 1 provides a fundamental introduction and contextual discussions to provide a unified theme for the subsequent chapters into a complete work. Chapters 2, 3 and 4 are arranged for ease of reading and understanding separately. Future research directions are proposed in Chapter 5 based on our findings.

Chapter 1, *Parametric Complementarity Constrained Programs— a Review of Methodologies*, summarizes the basic techniques that are used in the algorithms for solving the mathematical program with complementarity constraints (MPCC), which is also referred to as the mathematical program with equilibrium constraints (MPEC) interchangeably in the chapter. We review the philosophy and main techniques behind the existing algorithms developed for solving MPEC. This background knowledge is followed by a section focusing on the methodologies for solving the specific class of problems that are uni-parametric, bi-parametric, and multi-parametric complementarity constrained. One of the main sources of the parametric complementarity constrained program, inverse optimization, is defined in this chapter.

A linear program with linear complementarity constraints (LPCC) is among the simplest mathematical programs with complementarity constraints. Yet the global solution of the LPCC remains difficult to find and/or verify. In Chapter 2, *Global Solution of Bi-Parametric Linear Complementarity Constrained Linear Programs*, we study a specific type of the LPCC which we term a bi-parametric LPCC. Reformulating the bi-parametric LPCC as a non-convex quadratically constrained program, we develop a domain-partitioning algorithm that solves a series of linear
subprograms and/or convex quadratically constrained subprograms obtained by the relaxations of the complementarity constraint. We control the domain on which the partitioning is done via a pair of scalars that define the slope and intercept of a line in the bi-parametric space. Numerical results of the algorithm are presented.

An important application of bi-parametric LPCC is the Cross-validated Support Vector Machine Regression Parameters Selection Problem. The Support vector machine regression is a robust regression method to minimize the sum of deducted residuals, and thus is less sensitive to changes of data points near the regression hyperplane than the standard regression method. Two design parameters, the insensitive tube size ($\varepsilon_e$) and the weight ($C_e$) assigned to the regression error, are selected by users via a cross validation technique to gain better forecasts. The cross-validated parameter selection procedure can be formulated as a bi-level optimization problem, which then is equivalently reformulated as an LPCC. In Chapter 3, we propose a two-stage global optimization algorithm to solve this LPCC. The algorithm exhausts invariancy regions without explicitly identifying the edges of the regions on the parameter plane ($((C_e, \varepsilon_e)$-plane). This algorithm is tested on synthetic and real-world support vector machine regression problems with up to hundreds of data points and compared with several other approaches. The resulting global optimal parameter is important and can be serve as a benchmark for any other selection of parameter values.

In Chapter 4, we study an inverse optimization problem: Estimation of Pure Characteristics Demand Models. The pure characteristics demand model (PCM) is a discrete-choice model that formulates a consumer’s utility of a product by a linear function on a bundle of quantitative and observed product characteristics, the product price, and one unobserved product characteristic. The estimation of PCM calculates consumer specific coefficients of the observed product characteristics so that the observed market level data, such as market shares, are appropriately reflected. This process also requires an estimation of the unobserved product characteristic. Traditional algorithms used in the economics literature include contraction mapping, element-by-element inverse, and homotopy methods. These methods, however, are time-consuming if an exact solution is required, and are limited to solving specific types of numerical examples. In this chapter, we construct a hi-
erarchical mathematical program to formulate the estimation problem, which is a significantly superior to the conventional methods for estimating PCM. The framework of this mathematical program also allows the extension to deal with broader scopes of market level data. In addition to the observed market share considered in the literature, we introduce a Nash-Bertrand game to reflect the mechanism of firms’ competition and market optimization. The objective function of this hierarchical mathematical program employs the Generalized Method of Moments (GMM) to identify the values of the unobserved product characteristics, so they are least correlated to the observed ones. The resulting mathematical program belongs to the class of quadratic programs with nonlinear complementarity constraints. Three variations of the PCM estimation models are developed and validated by synthetic numerical experiments.
Acknowledgements

I want to express my deepest thanks for the advice, guidance and support from my mentor and master Professor Jong-Shi Pang. I am more than fortunate to have had the chance to work closely with Professor Pang since 2008. He not only has broad knowledge in mathematics, optimization and engineering, but also precise insight toward many research and general topics. During my PhD study, he taught me the methods and attitudes to conduct research and led me to think and debate back and forth. The weekly meeting with him was always motivating and joyful. I am truly grateful for the training he gave to sharpen my research skills and the patience and understanding he showed when I was frustrated.

I am excited and thankful for having Professors Ramavarapu S. Sreenivas, Angelia Nedich and Che-Lin Su on my committee. I have taken two of the courses in optimization from Professor Nedich and two of the courses in control systems from Professor Sreenivas. For me, both are excellent teachers and vivid role models of respectable scholars. I have also been very lucky to discuss research and have a joint work with Professor Che-Lin Su of the University of Chicago. I feel honored that these fine professors could be the first readers of my thesis.

Besides these people, I am grateful for the lectures given by Professors Xin Chen, Hayri Onal, Uday V. Shanbhag, Enlu Zhou and Mattias Polborn. Their lectures and our discussions outside the classroom have granted me abundant professional knowledge that can be applied in my research. Their lectures are among my most precious memories in the University of Illinois at Urbana-Champaign.

The encouragement and support from my mother Wen-Sen Chou, my father Shih-Ter Li, and my aunt Win-Lin Chou have helped me pass through the emotional and technical difficulties I confronted during these years. Although the distance
between us is far, their love is my power. It would not have been possible for me to complete this thesis without their unchanging support.

Special thanks go to the department office administrator Amy Summers, who has been taking care of me in the Department of Industrial and Enterprise Systems Engineering, and my English teacher Ann R. Spear, who constantly pushes me to read and write.

I also want to express thanks for the reliable help and companionship ever given during my graduate study from my friends, many of whom have already graduated from this school. Their names are Hsiao-Mei Wu, Yi-Chun Chen, Amy Lien, Yung-Cheng Rex Lai, Albert Y. Chen, Yan Jiang, Xiaopeng Li, Yinghan Annie Wang, Leslie Wang, Peng Jiang, Sunjae (Liz) Hong, Yun Bai, Mei-Hsiu Lai, I-Ju Chen, Wan-Ting (Grace) Chen, Yae-jin Moon and Hengyi Fu.
Table of Contents

List of Figures ................................................................. x

List of Tables ................................................................. xii

Preface ................................................................. xv

Chapter 1: Parametric Complementarity Constrained Programs--
        a Review of Methodologies .............................................. 1
        1 Mathematical programs with complementarity constraints ... 1
        2 Parametric linear complementarity constrained program .... 15
        3 Inverse optimization ................................................. 22

Chapter 2: Global Solution of Bi-Parametric Linear Complementar-
tarity Constrained Linear Programs ........................................ 23
        1 Introduction ............................................................ 23
        1.1 Bi-parametric LPCC example: Cross-validated Support
             Vector Machine Regression ................................... 30
        2 Solution techniques .................................................... 35
        2.1 Aggregated convex relaxation .................................... 36
        2.2 Depth-first domain-partitioning scheme ....................... 39
        2.3 Changing plane method ............................................ 41
        2.4 Linearization ........................................................ 43
        3 Algorithm ............................................................... 45
        3.1 Subproblems ......................................................... 45
        3.2 Domain partitioning algorithm ................................... 51
        4 Numerical experiment ................................................. 58
## Table of Contents

4.1 M is randomly generated (positive semi-definite) matrix .......................... 58
4.2 M is skew symmetric ................................................. 61
4.3 B is null matrix .................................................. 62
4.4 M has different rank and density .................................. 63
4.5 Effect on different choices of \((s, t)\) .......................... 63
4.6 Effect on approximation error window ......................... 65

5 Conclusion ............................................................ 67

### Chapter 3: Global Resolution of the Cross-validated Support Vector Machine Regression Parameters Selection Problem  . 71

1 Introduction .......................................................... 71

2 Models and Properties ................................................. 75
   2.1 Support Vector Machine Regression ....................... 75
   2.2 Cross-validated SVM Regression LPCC formulation .... 77

3 \((C_e, \varepsilon_e)\)-rectangle search algorithm .................... 83
   3.1 Lower-level problem with fixed \((C_e, \varepsilon_e)\) ............ 84
   3.2 Piece of the complementarity and data point allocation (grouping) in space ................. 90
   3.3 Restricted LP, Reduced Restricted LP, and Restricted QCP ........................................ 96
   3.4 Invariancy interval along a chosen line ................. 100
   3.5 The 1st stage: identify rectangles of the same grouping ................................... 105
   3.6 The 2nd stage: identify the non-vertical-and-non-horizontal boundary .................... 110

4 Integer Program with the Big-Numbers Tightening Procedure  . 118

5 Numerical Experiments ................................................ 124
   5.1 Synthetic data .................................................. 126
   5.2 Real-world data ................................................ 129
   5.3 Performance analysis between methods .................. 131
   5.4 Observation on solutions .................................. 134

6 Conclusion ............................................................ 135
# Chapter 4: Estimation of Pure Characteristics Demand Models Under Firm’s Competition and Market Optimization

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Introduction</td>
<td>146</td>
</tr>
<tr>
<td>2 Utility function of the Pure Characteristics Demand Model</td>
<td>148</td>
</tr>
<tr>
<td>3 Nash-Bertrand game</td>
<td>149</td>
</tr>
<tr>
<td>3.1 Firm’s pricing problem</td>
<td>149</td>
</tr>
<tr>
<td>3.2 Market optimization problem</td>
<td>151</td>
</tr>
<tr>
<td>3.3 Equilibrium of the $F + 1$ players</td>
<td>151</td>
</tr>
<tr>
<td>4 Lowest-utility constrained Nash-Bertrand game</td>
<td>155</td>
</tr>
<tr>
<td>5 PCM Estimation in game</td>
<td>157</td>
</tr>
<tr>
<td>5.1 Observed quantities</td>
<td>158</td>
</tr>
<tr>
<td>5.2 Structural parameters</td>
<td>159</td>
</tr>
<tr>
<td>5.3 Generalized Method of Moments for estimating unobserved characteristics</td>
<td>159</td>
</tr>
<tr>
<td>5.4 Estimation model– quadratic programs with complementarity constraints</td>
<td>160</td>
</tr>
<tr>
<td>5.5 Simplified estimation model</td>
<td>163</td>
</tr>
<tr>
<td>6 Numerical study</td>
<td>164</td>
</tr>
<tr>
<td>6.1 List of parameters, sets and variables</td>
<td>164</td>
</tr>
<tr>
<td>6.2 Selection of the instrumental matrices and weight matrices</td>
<td>165</td>
</tr>
<tr>
<td>6.3 Game-simulated data generation</td>
<td>168</td>
</tr>
<tr>
<td>6.4 Numerical experiments and results</td>
<td>169</td>
</tr>
<tr>
<td>7 Discussion</td>
<td>179</td>
</tr>
</tbody>
</table>

# Chapter 5: Future Directions

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>8 Discussion</td>
<td>182</td>
</tr>
</tbody>
</table>

# Bibliography

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>8 Bibliography</td>
<td>185</td>
</tr>
</tbody>
</table>
List of Figures

In Chapter 2:

1. Geometrical meaning of the Support Vector Machine Regression, where \( \frac{2\varepsilon_e}{\|w_s\|} \) is the margin size, \( \varepsilon_e \) is the tube radius, and \((w_s, b_s)\) defines the regression hyperplane. (Note that what is shown in the figure is not an optimal hyperplane.) ......................................... 29

2. Approximation error from the McCormick bound .......................... 37

3. Approximation error from the quadratic-difference upperbound . . 37

4. A geometrical illustration of the partitioning process and strategy on \((x_1 + z_1, x_2 + z_2)\)-plane. ......................................................... 41

In Chapter 3:

5. Within the SVR context, a single data point can be labeled by its allocation in space. ................................................................. 91

6. Problematic invariancy intervals \( I^B = [C_{min}^B, C_{max}^B] \) subsequent to the interval \( I^A \) on a line with fixed \( \varepsilon_e \) .............................. 103

7. The appropriate invariancy interval \( I^B = [C_{min}^B, C_{max}^B] \) subsequent and adjacent to the interval \( I^A \) on a line with fixed \( \varepsilon_e \) .......... 103

8. An example \((35, 35, 5, 2)\) of searching along the boundaries of a rectangle and finding the invariancy intervals using the Algorithm 3.4 with the add-in. ......................................................... 105

9. The six cases in which the rectangular area is bisected by a straight line into two invariancy regions A1 and A2 on the \((C_e, \varepsilon_e)\)-plane. . 115

10. Example for the Exception 1 in the Step 3 of the 2nd stage. There are three invariancy regions, A1, A2 and A3. .............................. 116
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>$(C_e, \varepsilon_e)$-rectangle search algorithm. The 1st and the 2nd stages are both illustrated.</td>
<td>116</td>
</tr>
<tr>
<td>12</td>
<td>The four quadrants of difficulty that are separated by the number of training data points each fold and the features of a data point of an instance.</td>
<td>128</td>
</tr>
<tr>
<td>13</td>
<td>Running time of the instances belonging in four quadrants separately.</td>
<td>130</td>
</tr>
<tr>
<td>14</td>
<td>Comparisons of convergent capability and convergent speed between the $(C_e, \varepsilon_e)$-rectangle search algorithm and the improved integer program on the same instances.</td>
<td>133</td>
</tr>
</tbody>
</table>
# List of Tables

In Chapter 2:

1. Global solution and algorithm performance on randomly generated problems. ............................................. 60
2. Global solution and algorithm performance on problems with skew symmetric matrix $M$. .......................... 61
4. Global solution and algorithm performance on LPCC in [120]. ................................................................. 64
5. Algorithm performance on different choice of $(s, t)$. ........................................................................... 66
6. Algorithm performance on different approximation error windows. ......................................................... 68

In Chapter 3:

7. Examples of the averaged tightened values $\theta_1, \theta_2, \theta_3, \theta_4, \theta_5$, and $\theta_6$. Each data point has 30 features. .......................................................... 125
8. Result of 1-fold training and 1-fold testing data solved by the $(C_e, \varepsilon_e)$-rectangle search algorithm. $C_e \in [1, 10]$ and $\varepsilon_e \in [0.1, 0.5]$ ........................................ 137
9. Result of 2-fold training and 2-fold testing data solved by the $(C_e, \varepsilon_e)$-rectangle search algorithm. $C_e \in [1, 10]$ and $\varepsilon_e \in [0.1, 0.5]$ ........................................ 138
10. Result of 3-fold training and 3-fold testing data solved by the $(C_e, \varepsilon_e)$-rectangle search algorithm. $C_e \in [1, 10]$ and $\varepsilon_e \in [0.1, 0.5]$ ........................................ 139
11. Profile of the chemoinformatics data. (Table 3.1 in [72]) ................................................................. 139
12. Result of the real-world, 5-fold training, and 1-fold testing data solved by the $(C_e, \varepsilon_e)$-rectangle search algorithm. $C_e \in [0.1, 10]$ and $\varepsilon_e \in [0.01, 1]$ ........................................ 140
<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>13</td>
<td>Comparisons between three methods. Results of solving the cross-validated SVR with 1-fold training and 1-fold testing data. $C_e \in [1, 10]$ and $\varepsilon_e \in [0.1, 0.5]$</td>
</tr>
<tr>
<td>14</td>
<td>Comparisons between three methods. Results of solving the cross-validated SVR with 2-fold training and 2-fold testing data. $C_e \in [1, 10]$ and $\varepsilon_e \in [0.1, 0.5]$</td>
</tr>
<tr>
<td>15</td>
<td>Comparisons between three methods. Results of solving the cross-validated SVR with 3-fold training and 3-fold testing data. $C_e \in [1, 10]$ and $\varepsilon_e \in [0.1, 0.5]$</td>
</tr>
<tr>
<td>16</td>
<td>Results of the chemoinformatics cross-validated SVR with 5-fold training and 1-fold testing data solved by KNITRO.</td>
</tr>
<tr>
<td>17</td>
<td>Results of the chemoinformatics cross-validated SVR with 5-fold training and 1-fold testing data solved by CPLEX as an improved integer program (with the tightened $\theta$).</td>
</tr>
</tbody>
</table>

In Chapter 4:

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>18</td>
<td>Values of $\tilde{\beta}$, $\bar{\alpha}$, $\sigma_{\alpha}$, $\phi$, $\xi$, $\omega$, and $M_t$ in the instances of Nash-Bertrand game.</td>
</tr>
<tr>
<td>19</td>
<td>Results of solving Nash-Bertrand game with an objective function $\sum_{ijt} \pi_{ijt}$.</td>
</tr>
<tr>
<td>20</td>
<td>Results of the parameters estimation. $\bar{\alpha} = 1$. The objective function minimizes the orthogonality in GMM and the sum of squared deviation of parameters.</td>
</tr>
<tr>
<td>21</td>
<td>Results of the parameters estimation. $\bar{\alpha} = 1$. A non-typical size of the instrumental variables: $H &gt; J \times T &gt; K$.</td>
</tr>
<tr>
<td>22</td>
<td>Results of the parameters estimation. $\bar{\alpha} = 1$. No SSD term in the objective function.</td>
</tr>
<tr>
<td>23</td>
<td>Parameters in the instances of lowest-utility constrained Nash-Bertrand game and results.</td>
</tr>
<tr>
<td>24</td>
<td>Consumers’ purchasing behavior in the instances of lowest-utility constrained Nash-Bertrand game.</td>
</tr>
<tr>
<td>Table</td>
<td>Description</td>
</tr>
<tr>
<td>-------</td>
<td>-------------</td>
</tr>
<tr>
<td>25</td>
<td>Results of the parameters estimation under lowest-utility constrained pricing game. $\bar{\alpha} = 1$. The objective function minimizes the orthogonality in GMM and the sum of squared deviation of parameters.</td>
</tr>
<tr>
<td>26</td>
<td>Results of parameters estimation with $\bar{\alpha}$ fixed at values different from the true one.</td>
</tr>
<tr>
<td>27</td>
<td>Results of the simplified parameters estimation.</td>
</tr>
</tbody>
</table>
Preface

The equilibrium problem has been an important problem in formulating scientific and social systems. A system’s equilibrium usually involves the mathematical formulation as a collection of complementarities, and thus makes the complementarity constrained program one of the most useful tools in understanding the balance, competition and strategy chosen among several options in the system.

The core of this thesis is the parametric complementarity constrained programs, especially those arising from the parameter selection problem, which are known as the inverse optimization problem. The complementarity constraints involved in these programs are in fact the optimality Karush-Kuhn-Tucker (KKT) conditions of other convex optimization problems, which are called forward problems. In the two machine learning and economics models we study in this thesis and beyond, the forward problems are characterized by some parameters that are directly related to the effectiveness of the forward models. The number of these parameters determines whether the inverse models, as complementarity constrained programs, are uni-parametric, bi-parametric or multi-parametric.

Therefore, the first emphasis of this thesis is on the suitable and tractable inverse models. They are the parameter selection for the Support Vector Machine Regression (SVR) and the parameter estimation for the Pure Characteristics Demand Model (PCM) in Chapters 3 and 4 respectively. The former inverse model becomes a bi-parametric linear program with linear complementarity constraints (LPCC), while the later inverse model, assuming one of the estimators is known in advance, is formulated as a multi-parametric quadratic program with linear complementarity constraints (QPCC).

The second emphasis is on algorithms for finding the global optimal solution to
the (bi-)parametric complementarity constrained programs. Prevailing solvers for
the mathematical programs with complementarity constraints (MPCC) adapt the
algorithms from nonlinear programming, most of which are sequential quadratic pro-
gramming (SQP) methods, active set methods, and interior point methods. These
algorithms find a stationary point of MPCC, whose definition will be reviewed in
Chapter 1, but lack a certificate of global optimality. The parameters obtained
at a stationary solution of the inverse model simply have a gap towards the best
selection. The algorithms developed in Chapters 2 and 3 aim at obtaining the
global optimality certificate for general bi-parametric complementarity constrained
programs and SVR parameter selection respectively. For PCM parameter estima-
tion in Chapter 4, however, we only point out the condition at which the global
optimality can be verified.

The analysis in this thesis is built upon the foundations of mathematical pro-
grams with equilibrium constraints (MPEC) and linear complementarity problems
(LCP), and their usage is linked with the inverse models of the original SVR and
PCM. Many techniques involved in developing algorithms are borrowed from or
inspired by parametric programming. Besides these, the numerical studies rely
on the maturity of solvers (CPLEX, SNOPT, PATH and KNITRO) in solving var-
ious types of the optimization programs. I gratefully mention a list of books,
papers and PhD theses that motivated the early research stage of this thesis:
[38, 39, 31, 82, 2, 52, 72, 120, 69, 16].
Chapter 1: Parametric Complementarity

Constrained Programs— a Review of Methodologies

1 Mathematical programs with complementarity constraints

Mathematical optimization, or mathematical programming, is a process to find the best selection from a pool of candidates. This process requires that the elements in the pool of candidates are comparable and the sense of optimality is well defined. The sense of optimality is a specific mathematical rule based on which any element is either better, equal, or worse than another in the pool of candidates. The mathematical programming problem is usually presented as an optimization model comprising the sense of optimality, objective function, variables, and constraints. In other cases, it can be shown in a network formulation (featuring nodes, arcs, and paths), or in a dynamic programming model (defined by states, activities, transition function, and objective function). Branches of the optimization problem defined by the models are distinguished by different properties or forms of the objective function, variables and constraints. Most common examples include convex programming, linear programming, integer programming, and nonlinear programming. Under these classifications, prosperous subfields include quadratic programming (where the objective function is quadratic and the constraints are linear),
semidefinite programming [115] (which is constrained by matrix-form equalities or inequalities, and the underlying matrix is positive semidefinite), conic programming (where the feasible regions are defined by some type of cones), stochastic programming (where the uncertain parameters appear in the form of expectation given the probability distribution of the parameters), hierarchical programming (where the constraint set of an optimization model is defined by the optimal solution set of another optimization model) and fractional programming [25, 89] (where the objective function contains fractions). Compared to others, a mathematical program with complementarity constraints (MPCC), which we focus on in this section, features the constraint set being the intersection of a polyhedron and a set of linear complementarities. This model is special enough to be studied in a unique framework.

The general form of a MPCC is as follows:

$$\begin{align*}
\min_y & \quad f(y) \\
\text{subject to} & \quad g_i(y) = 0, \quad \forall i \in \text{SideConstraints}_g, \\
& \quad g_i(y) \geq 0, \quad \forall i \in \text{SideConstraints}_i, \\
& \quad 0 \leq y_j \perp h_j(y) \geq 0, \quad \forall j \in \text{ComplementarityConstraints},
\end{align*}$$

(1)

where \( y_j \perp h_j \) denotes the complementarity \( y_j h_j = 0 \). If \( g \) and \( h \) are linear functions, the constraint set is named as the “linear” complementarity, and the satisfiability problem \( 0 \leq y \perp h(y) \geq 0 \) itself is called the linear complementarity problem (LCP) (See [31]). In the linear case, if \( y \) solves \( 0 \leq y \perp h(y) \geq 0 \), \( y \geq 0 \) also solves the affine variational inequality (VI) (See [38, 39])

$$\begin{align*}
(\Phi - y)^T h(y) \geq 0, \quad \forall \Phi \geq 0,
\end{align*}$$

and the optimization problem

$$\begin{align*}
\min_y & \quad y^T h(y) \\
\text{subject to} & \quad y \geq 0, h(y) \geq 0.
\end{align*}$$

(2)

If the complementarities in (1) are replaced by (2), the resulting formulation becomes a nonlinear bi-level program (See [34]), a class of the hierarchical program-
ming. In the case where \( h(y) \) is the gradient of another function \( q(y) \), i.e., \( h(y) = \nabla q(y) \), problem \( (2) \) is equivalent to \( \min_{y \geq 0} q(y) \).

Next we discuss some terminologies in game theory. The *Stackelberg game* (See [109]) formulation is an example of the mathematical program with complementarity constraints. In the Stackelberg game, the leader chooses an action, then the follower chooses an action. The leader anticipates the follower’s reaction and makes his decision, while the follower can see the leader’s output and makes a rational decision accordingly. The solution to a Stackelberg game is a *subgame perfect Nash equilibrium*, which is the Nash equilibrium of every subgame given the moves within an arbitrary history.

The studies about MPCC are often seen under a more general topic, *mathematical programs with equilibrium constraints* (MPEC) (See [82]). The results reviewed in the following paragraphs fall mostly under the framework of MPEC and MPCC, while the linear program with linear complementary constraints (LPCC) is a special case of them.

The MPEC distinguishes itself from other classes of the optimization problem by the special features of its constraints set:

1. The constraints set in \( (1) \) is not a convex set;

2. The constraints set is disjunctive; this nature makes the problem NP-hard (See relevant work in [84]);

3. Consider the equivalent nonlinear program to \( (1) \):

\[
\begin{align*}
\min_{y} & \quad f(y) \\
\text{subject to} & \quad g_{E_1}(y) = 0, \\
& \quad g_{E_2}(y) \geq 0, \\
& \quad y_j h_j(y) = 0, \\
& \quad h_j(y) \geq 0, \\
& \quad y_j \geq 0,
\end{align*}
\]

and the gradient of the function \( y_j h_j(y), \forall j \). The set \{\( h_j(y) + y_j \nabla y_j h_j(y), \forall j \}\) is linearly dependent at any feasible solution \( y \) (See [29]), and the Mangasarian-
Fromovitz constraint qualification (MFCQ) doesn’t hold. This violation implies the unbounded Lagrangian multipliers [6, 50] in the Karush-Kuhn-Tucker (KKT) system of (3). Therefore, the direct applicability of the KKT-based analysis and the associated algorithms of nonlinear programming is in jeopardy.

To find solutions to an MPEC, the development of solution algorithm may involve one or several techniques: the NCP-function and restricted NCP function, smoothing method, penalization scheme, regularization (or relaxation), binary variables, and implicit program.

**NCP-function and Restricted NCP-function.** This technique restates the complementarities $0 \leq a \perp b \geq 0$, so the solution to the restatement is also a solution to the original problem. A function $\psi$ is a nonlinear complementarity program-function (NCP-function) by definition if

$$\psi(a, b) = 0 \Leftrightarrow [a > 0, b > 0, ab = 0].$$

(Here we follow the terminologies applicable for a general MPEC where the linearity of the complementarity is not specified.) The must-know NCP-functions include

$$\psi_{\text{min}}(a, b) := \min(a, b),$$

$$\psi_{\text{max}} := |a - b| - a - b, \text{ and}$$

$$\psi_{FB} := \sqrt{a^2 + b^2} - (a + b).$$

The last example is called the Fisher-Burmeister function. Besides these, the implicit Lagrangian function $M_\alpha(x)$ proposed in [88] for the complementarity $0 \leq x \perp F(x) \geq 0$ also satisfies the property of an NCP-function.

$$M_\alpha(x) := xF(x) + \frac{1}{2} \{\|(-\alpha F(x) + x)_+\|^2 - \|x\|^2 + \|(-\alpha x + F(x))_+\|^2 - \|F(x)\|^2\},$$

where $(\cdot)_+$ denotes $\max(\cdot, 0)$. Literature on the NCP-functions or equivalent concepts includes [86, 44, 40, 68, 110].
On the other hand, the definition of the restricted nonlinear complementarity program-function (restricted NCP-function) is given in [119]. The function $\phi$ is called a restricted NCP-function if for given $a > 0$, it holds that

$$\phi(a, b) = 0 \iff [ab = 0, b > 0].$$

The functions being raised under the topic of the restricted NCP include

$$\phi_{\min}(a, b) := |\min(a, b)|^2,$$
$$\phi_{FB} := \left[\sqrt{a^2 + b^2} - (a + b)\right]^2$$

(the squared Fisher-Burmeister function),

$$\phi_S := a[(b)_+]^2 + [(-b)_+]^2$$

(the Solodov function),

$$\phi_A := a[(b)_+]^3 + \frac{1}{2}(\sqrt{a^2 + b^2} - (a + b))^2$$

(the augmented squared Fisher-Burmeister function), and

$$\phi_{RG}(a, b) := ab + \frac{1}{2a}\{[(a - ab)_+]^2 - a^2\}$$

(the regularized gap function). Works on the restricted NCP-functions can be found in [79, 119].

**Smoothing method.** This technique employs a continuously differentiable function to approximate the complementarity condition, so the favorable properties around the approximation function such as the smoothness, nonsingularity, limit existence, etc. can augment the convergence result for some algorithms. One widely used smooth function is the $\phi_\mu : \mathbb{R}^2 \to \mathbb{R}$ given by

$$\phi_\mu(a, b) := \sqrt{(a - b)^2 + 4\mu^2} - (a + b),$$

where $\mu \neq 0$ is a parameter. The function $\phi_\mu(a, b) = 0$ if and only if $[a \geq$
0, b ≥ 0, and ab = μ²]. Moreover, lim_{μ→0} φ_μ(a, b) = min(a, b). The studies on the properties and algorithms related to (4) are found in [37, 67].

The following smooth function \( p(y)_α \) with a parameter \( α > 0 \):

\[
p_α(y) := y + \frac{1}{α} \log(1 + e^{-αy}),
\]

is an approximation to the plus function \((y)_+ = \max\{y, 0\}\). This function is derived from the integral of the sigmoid function of neural network. The plus function is related to the complementarity \(0 ≤ y_⊥ b ≥ 0\) due to its equivalent formulation \(y_⊥ (y_⊥ h(y))_+ = 0\). The main results, shown in [26], on properties of \( p_α(y) \) include 1) \( p_α(y) \) is \(k\)-times continually differentiable, 2) \( \lim_{|y|→∞} p_α(y)_⊥ y_+ = 0, \forall α > 0, \) and 3) the inverse function of \( p_α(y) \) is well defined. Relevant works are found in [27, 111].

The entropic regularization function \( φ_ρ [41, 77] \) was originally developed for solving the min-max optimization and semi-infinite programs. The study in [19] applies the smooth and concave entropic regularization function

\[
φ_ρ(a, b) := -\frac{1}{ρ} \log\{e^{-ρa} + e^{-ρb}\},
\]

where parameter \( ρ > 0 \), to solve an MPEC with the complementarities \(0 ≤ a_⊥ b ≥ 0\).

**Penalization scheme.** The penalization scheme solves the following program instead of the MPEC model (1):

\[
\min_y \quad f(y) + ρψ(y)
\]

subject to

\[
g_{ε_i}(y) = 0,
\]

\[
g_{τ_i}(y) ≥ 0,
\]

and \( y_j ≥ 0, h_j(y) ≥ 0 \),

where \( ρ > 0 \) is a given parameter and \( ψ(y) \) is a function such that \( ψ(y) ≠ 0 \) if the complementarities \( y_j h_j(y) = 0 \) are violated at \( y \). Consider a sequence of stationary point \( \{y^k\} \) of (7) and \( ρ = ρ^k \), where \( ρ^k → ∞ \). Under several
conditions (the Conditions A, B and C in [61]), the limit point \( \mathbf{y} \) of the sequence \( \{\mathbf{y}^k\} \) is a stationary point to the MPEC model (1). If the problem satisfies further constraint qualifications (CQ1 and CQ2 in [61]), a limit point \( \mathbf{y} \) guarantees its feasibility to (1). A good choice of \( \psi(\mathbf{y}) \) is \( \sum_j y_j h_j(\mathbf{y}) \) (See [61]). Works related to this technique include [104, 61, 63, 112].

The elastic mode embedded in the sequential quadratic programming (SQP) algorithm in [8] to solve an MPEC is also a penalization technique. Originated in [53], the elastic mode is activated if a nonlinear subproblem proves to be infeasible or unbounded when solving the following nonlinear programming by SQP algorithm.

\[
\begin{align*}
\min_{x,s_N,s_L} & \quad f(x) \\
\text{subject to} & \quad F(x) - s_N = 0, \\
& \quad Gx - s_L = 0, \\
& \quad l \leq x, s_N, s_L \leq u.
\end{align*}
\]  

(8)

In (8), \( s_N \) and \( s_L \) are slack variables for nonlinear and linear slack respectively. The following program is then solved instead of the infeasible or unbounded subproblems:

\[
\begin{align*}
\min_{x,v,w} & \quad f(x) + \rho e^T(v + w) \\
\text{subject to} & \quad F(x) - v + w = 0, \\
& \quad l' \leq Gx \leq u', \\
& \quad l'' \leq x \leq u'', \\
& \quad v, w \geq 0,
\end{align*}
\]  

(9)

where the bounds of constraints in (8) become elastic. In the model (9), \( e \) is a vector of ones, \( v \) and \( w \) are additional variables, \( \rho \) is called the elastic weight, and \( f(x) + \rho e^T(v + w) \) is called the composite objective. It is shown in [8] that the solution obtained from SQP with the elastic mode is an exact solution to the original MPEC.

Regularization (or Relaxation). The regularization technique specifically refers
to introducing a parameter $t > 0$ to reformulate the MPEC model (1) as

$$\min_y f(y)$$

subject to

$$g_{E_i}(y) = 0,$$
$$g_{I_i}(y) \geq 0,$$
$$y_j h_j(y) \leq t,$$
and $$y_j \geq 0, h_j(y) \geq 0.$$  \hspace{1cm} (10)

The nonlinear program (10), which is “regularized,” satisfies the constraints qualifications such as MFCQ. The regularized model (10) doesn’t give an exact solution to the original MPEC. Therefore, in some works, regularization is alternatively referred to as “relaxation.” The amount of the difference between the regularized and the exact solutions yet can be measured. Suppose $y^*$ is a solution to the MPEC, [104] shows that under some conditions (MPEC-SOSC in [104]), the distance between $y^*$ and a local solution $y(t)$ to (10) is within $O(\sqrt{t})$, i.e., the distance is within a constant times $\sqrt{t}$ if $y(t)$ is near $y^*$. Studies relevant to the regularization technique include [107, 104, 66, 91].

**Binary variables.** This technique arises from the disjunctive nature of an MPEC and relates the MPEC to the big-M method of (0-1)-integer programming. With additional binary variables $z_j$ and the valid vector of big numbers $\theta$ (the same function as the big-M), the model (1) is equivalent to

$$\min_{y,z} f(y)$$

subject to

$$g_{E_i}(y) = 0,$$
$$g_{I_i}(y) \geq 0,$$
$$0 \leq y_j \leq \theta_j(1 - z_j),$$
$$0 \leq h_j(y) \leq \theta_j z_j,$$
and $$z_j \in \{0, 1\}.$$  \hspace{1cm} (11)

The studies on this technique can be found under the framework of the bi-level program [55]. The results of employing this technique to an MPEC are
discussed in [59], where a Benders decomposition scheme to solve the model (11) is proposed, and the boundedness of $\theta$ is shown not required.

Implicit program. The implicit programming technique to be employed on an MPEC requires the equivalent formulation of a parametric MPEC. Dividing the vector of variables $y$ in (1) into $[x; v]$, the parametric MPEC $(x)$ is defined as

$$\min_{x, v} f(x, v)$$

subject to

$$g_{E_i}(x, v) = 0,$$

$$g_{I_i}(x, v) \geq 0,$$

and

$$0 \leq v_j \perp h_j(x, v) \geq 0.$$

(12)

Let $S(x)$ be the set of the feasible solution $v$ to the complementarities for every $x$: $0 \leq v_j \perp h_j(x, v) \geq 0, \forall j$. If $S(x)$ is a singleton and is expressed by $v(x)$, the implicit program is formulated as

$$\min_{x} \tilde{f}(x)$$

subject to

$$\tilde{g}_{E_i}(x) = 0,$$

and

$$\tilde{g}_{I_i}(x) \geq 0,$$

(13)

where

$$\tilde{f}(x) := f(x, v(x)) \text{ and } \tilde{g}(x) := g(x, v(x)),$$

(14)

are implicit functions. For any $x$, if $\tilde{h}(y) = h(x, \cdot)$ is a strong monotone function, i.e., there exists a modulus $c > 0$ such that $(y_1 - y_2)^T(\tilde{h}(y_1) - \tilde{h}(y_2)) \geq c\|y_1 - y_2\|^2$, then $S(x)$ is a singleton. The methodology arising from the implicit program can be found in [71].

Besides the aforementioned techniques, which are more or less used in the algorithms proposed to solve an MPEC, the research on MPEC methodologies is analytically concerned with the efficiency and accuracy of the algorithms. Specifically, the contemporary philosophy discussed in the literature includes the MPEC-constraint qualification, stationarity, optimality condition, global or local optimality, global or local convergence, and rate of convergence. These concepts or properties are briefly reviewed in the following.
MPCC-Constraint qualification (of a feasible point). It is known that the Mangasarian Fromovitz constraint qualification (MFCQ) and the linear independence constraint qualification (LICQ) of nonlinear programming cannot be satisfied by any feasible solution to a mathematical problem with complementarity constraints. In the framework of MPCC [106], given a feasible solution $\mathbf{y}$ to the MPCC, the MPCC-constraint qualification MPCC-MFCQ (or MPCC-LICQ) holds if the MFCQ (or LICQ) condition is held at $\mathbf{y}$ for the following tightened program:

$$\begin{align*}
\min_{\mathbf{y}} & \quad f(\mathbf{y}) \\
\text{subject to} & \quad g_{E}(\mathbf{y}) = 0, \\
& \quad g_{I}(\mathbf{y}) \geq 0, \\
& \quad y_{j} = 0, \quad \text{if } \bar{y}_{j} = 0, \\
& \quad y_{j} \geq 0, \quad \text{if } \bar{y}_{j} \geq 0, \\
& \quad h_{j}(\mathbf{y}) = 0, \quad \text{if } h_{j}(\bar{\mathbf{y}}) = 0, \\
\text{and} & \quad h_{j}(\mathbf{y}) \geq 0, \quad \text{if } h_{j}(\bar{\mathbf{y}}) \geq 0.
\end{align*}$$

(15)

In the case where the complementarity constraints are linear, the tightened program is a linear program in which the feasible region is a subset of the feasible region of the original problem (1). The MPCC-constraint qualifications usually appear as assumptions or conditions for other stationarity, optimality, or convergence results.

Stationarity (of a solution). The concept of the stationarity is a geometric view of the feasible set. In general, a stationary point can be defined universally with a representation involving the tangent cone $T(\mathbf{y}; \mathcal{F})$ at a vector $\mathbf{y} \in \mathcal{F}$, where $\mathcal{F}$ denotes the feasible region of the target problem, such as (1). The vector $\mathbf{y}^*$ is said to be a stationary point of the target problem if

$$\mathbf{d} \in T(\mathbf{y}^*; \mathcal{F}) \Rightarrow \nabla f(\mathbf{y}^*)^T \mathbf{d} \geq 0. \quad (16)$$

By definition, the vector $\mathbf{d}$ is in the tangent cone $T(\mathbf{y}^*; \mathcal{F})$ if there is a sequence
of vectors \( \{y^k\} \rightarrow y^* \) and a sequence of scalars \( \{a^k\} > 0 \) such that \( \{\|a^k(y^k - y^*) - d\|\} \rightarrow 0 \). However, other definitions of the stationarity of the KKT-type employed in a nonlinear program requires the assumption of some constraint qualifications, and thus cannot be used directly on an MPEC. The theorems on the stationarity of an MPEC are categorized independently with those of nonlinear programming.

Following the definition in [106], Bouligand stationarity (or \( B \)-stationarity) holds at a feasible \( y^* \) in the MPEC (1) if \( \nabla f(y^*)^T d \geq 0 \) for every \( d \) satisfying

\[
\nabla g_E(y^*)^T d = 0, \\
\nabla g_I_i(y^*)^T d \geq 0, \quad \forall i \text{ such that } g_I_i(y^*) = 0, \\
\min\{d_j, \nabla h_j(y^*)^T d\} = 0, \quad \forall j \in \text{ComplementarityConstraints}.
\]

Other common types of stationarity include the strong stationarity (or \( S \)-stationarity), Mordukhovich stationarity (or \( M \)-stationarity), Clarke stationarity (or \( C \)-stationarity), and weak stationarity, whose definitions can be found in [78, 106, 104]. Among them, strong stationarity at a point is easiest to verify, and it implies the remaining types of the stationarity. Thus in a sufficient optimality condition, strong stationarity is often assumed. It is known [106, 105] that if strict complementarity holds at \( y^* \), the \( B \)-, \( S \)-, \( M \)-, \( C \)-, and weak stationarity are all equivalent.

A stationary point is usually the point of convergence of the MPEC algorithms, especially those extended from the nonlinear programming. Under a variety of assumptions, it is possible to relate a stationary point to a local optimal solution. It is shown in [106] that a local optimal solution to an MPCC is a \( B \)-stationary point if the exact penalty constraint qualification holds at the point.

**Optimality Condition (of a solution).** We review the definition of one of the widely-assumed optimality conditions: the MPCC-second order sufficient condition (MPCC-SOSC) in a representation involving the MPCC-Lagrangian function \( L(y, \lambda, \mu, \tau, \nu) \) and a critical cone \( C(y, \lambda, \mu, \tau, \nu) \) following the deriv-
tion in [106, 104]. The MPCC-Lagrangain function is of the form

\[ L(y, \lambda, \mu, \tau, \nu) = f(y) - \lambda^T g_I(y) - \mu^T g_E(y) - \tau^T y - \nu^T h(y), \]  

(18)

where \( \lambda, \mu, \tau, \) and \( \nu \) are the Lagrangian multipliers. Given a feasible solution \( y^* \) to (1) and the three index sets

\[ P_S = \{ i \in \text{SideConstraints} | g_I(y^*) = 0 \}, \]
\[ P_L = \{ j \in \text{ComplementarityConstraints} | y^*_j = 0 \}, \text{ and} \]
\[ P_R = \{ j \in \text{ComplementarityConstraints} | h_j(y^*) = 0 \} \]

(19)

associated with \( y^* \), the Lagrangian S-stationary system is written as

\[ \nabla f(y^*) - \sum_{i \in P_S} \lambda^*_i \nabla g_I(y^*) - \mu^T \nabla g_E(y^*) - \sum_{j \in P_L} \tau^*_j - \sum_{j \in P_R} \nu^*_j \nabla h_j(y^*) = 0, \]

\[ g_I(y^*) = 0 \text{ and } \lambda^*_i \geq 0 \quad \forall i \in P_S, \]
\[ y^*_j = 0, \quad \forall j \in P_L, \]
\[ h_j(y^*) = 0, \quad \forall j \in P_R, \]
\[ \tau^*_j, \mu^*_j \geq 0, \quad \forall j \in P_L \cap P_R, \]
\[ g_E(y^*) = 0, \quad \forall i \in \text{SideConstraints}, \]
\[ g_I(y^*) \geq 0, \quad \forall i \in \text{SideConstraints}, \]
\[ y^*_j \geq 0 \text{ and } h_j(y^*) \geq 0, \quad \forall j \in \text{ComplementarityConstraints}. \]

(20)

The sign of the feasible \( \lambda^*, \tau^*, \) and \( \nu^* \) in (20) defines the following six index sets

\[ D_{S+} = \{ i \in P_S | \lambda^*_i > 0 \}, \]
\[ D_{S0} = \{ i \in P_S | \lambda^*_i = 0 \}, \]
\[ D_{L+} = \{ j \in P_L \cap P_R | \tau^*_j > 0 \}, \]
\[ D_{L0} = \{ j \in P_L \cap P_R | \tau^*_j = 0 \}, \]
\[ D_{R+} = \{ j \in P_L \cap P_R | \nu^* > 0 \}, \text{ and} \]
\[ D_{R0} = \{ j \in P_L \cap P_R | \nu^* = 0 \}. \]

(21)

Given a strong stationary point \( y^* \) and the tuple \( \{ y^*, \lambda^*, \mu^*, \tau^*, \nu^* \} \) satisfying (20), the MPCC-SOSC holds at \( y^* \) if for every critical direction \( d \in \)
\( \mathbb{C}(\mathbf{y}^*, \lambda^*, \mu^*, \tau^*, \nu^*) \) such that \( \min \{d_j, \nabla h_j(\mathbf{y}^*)^T d \} = 0 \) \( \forall j \in D_{L0} \cap D_{R0} \), there exists \( \sigma > 0 \) and

\[
\sigma \leq d^T \left( \nabla_{yy}^2 L(\mathbf{y}^*, \lambda^*, \mu^*, \tau^*, \nu^*) \right) d.
\] (22)

(See Definition 2.7 in [104]) The critical cone \( \mathbb{C}(\mathbf{y}, \lambda, \mu, \tau, \nu) \) mentioned above is defined as

\[
\mathbb{C}(\mathbf{y}^*, \lambda^*, \mu^*, \tau^*, \nu^*) = \begin{cases} 
|d|_2 = 1, \\
\nabla g_E(\mathbf{y}^*)^T d = 0, \\
\nabla g_I(\mathbf{y}^*)^T d = 0, \forall i \in D_{S+}, \\
\nabla g_I(\mathbf{y}^*)^T d \geq 0, \forall i \in D_{S0}, \\
\nabla h_j(\mathbf{y}^*)^T d = 0, \forall j \in P_L \backslash P_R, \\
\nabla h_j(\mathbf{y}^*)^T d \geq 0, \forall j \in D_{L0}, \\
\nabla h_j(\mathbf{y}^*)^T d = 0, \forall j \in D_{L0}, \\
\n\end{cases}.
\] (23)

The MPCC-strong SOSC (MPCC-SSOSC) requires the condition (22) holding for every piecewisely defined critical direction \( d \). (See Definition 2.8 in [104]). Note that both the MPCC-SOSC and MPCC-SSOSC are sufficient conditions for a local minimizer. On the other hand, varieties of the first- and second-order necessary optimality conditions can be found in [83, 106, 64].

**Stability or Sensitivity (of a solution).** In stability analysis, we study the stationary or optimal solution sensitivity in the view of a parametric program where some parameters are subject to perturbation. Studies about the stability in MPEC have been extended from those of nonlinear programming. Consider the MPEC in the parametric form (12). If \( \mathbf{v}(\mathbf{x}) \) is a weak stationary point of the model \( MPEC(\mathbf{x}) \), it is shown in [106] that under the upper level strict complementarity condition and the nonvanishing determinantal sign condition, there exist open neighborhoods of \( \mathbf{v} \), open neighborhoods of \( \mathbf{x} \), and a piecewise smooth function \( \mathbf{v}(\cdot) \) defined in the space of \( \mathbf{x} \) such that
\( \mathbf{v}(\mathbf{x}) \) gives the unique weak stationary point of \( \text{MPEC}(\mathbf{x}) \). Furthermore, considering the global optimal solution \( \hat{\mathbf{v}}(\hat{\mathbf{x}}) \) of \( \text{MPEC}(\hat{\mathbf{x}}) \) and the global optimal value function \( \mathbb{W}(\mathbf{x}) \), given MPCC-LICQ and MPCC-strong SOSC at \( \hat{\mathbf{x}} \) (or other alternative conditions in [60]), the continuity of \( \mathbb{W}(\mathbf{x}) \) can be shown, and the closed-form of the first- and second- order derivatives, which suggest the rate and curvature of the change of the global optimal value with regard to the deviation of \( \mathbf{x} \) respectively, were established (See [60]).

Moreover, if \( h \) in (12) is a linear function, the results in the framework of \textit{parametric LCP}, \textit{parametric affine VI}, and \textit{parametric quadratic program} [52] (which is related to the model (2) with a linear \( h \)) are important sources for analyzing the stability of a mathematical program with linear complementarity constraints.

**Global vs Local Optimality (of a solution).** A solution \( \mathbf{y}^* \) to (1) is a local optimizer if there is a neighborhood around \( \mathbf{y}^* \) such that \( f(\mathbf{y}^*) \leq f(\mathbf{y}) \) for every feasible \( \mathbf{y} \) in the neighborhood, and it is a global optimizer if \( f(\mathbf{y}^*) \leq f(\mathbf{y}) \) for every feasible \( \mathbf{y} \) to (1). In contrast to a stationary point with desired geometric properties of the constraint set, the desired properties of a global or local optimal point are viewed from the objective value. The algorithms (interior point, active-set, sequential quadratic programming, etc.) that are inherited from nonlinear programming and adapted for an MPCC terminate at a local optimal point, and a global optimum can only be obtained at the smallest among all these local solutions; whereas algorithms of the branch-and-bound types (with the binary variable in (11)) terminate at a global optimal solution. The algorithms that solve for the global optimum of an MPEC are found in [92, 93, 59, 120].

**Remarks:** In Chapter 2 of this thesis a global optimization algorithm for the bi-parametric linear program with linear complementarity constraints (LPCC) is presented. In Chapter 3 of this thesis a bi-parametric LPCC model of a statistics application is solved to global optimum with a variety of methods. The numerical study about an economics model developed in Chapter 4 of this thesis relies on a solver, \textit{SNOPT}, to find a stationary solution to the multi-parametric quadratic program with linear complementarity constraints,
but the global optimality of this problem can be verified only for the case of zero objective value.

Global vs Local Convergence (of an algorithm). For an iterative algorithm, the global convergence holds if the starting point, say, an initial guess of $y$ to (1), can be chosen arbitrarily. For a locally convergent algorithm, however, it is required that the starting point is close enough to the ending point at the convergence. Works addressing the issue of the global convergence of some algorithms are contained in [49, 65, 28, 7, 75, 76].

Rate of convergence (of an algorithm). The rate of convergence being considered in a MPEC-algorithm is defined identically with that in nonlinear programming (See the definition, for example, in [95]). An iterative algorithm is said to converge linearly if the sequence $\{y^k\}$ converges to $y^*$ and there exists $r \in (0, 1)$ such that $\|y^{k+1} - y^*\| \leq r\|y^k - y^*\|$. It is said to converge quadratically if there exists $M > 0$ such that $\|y^{k+1} - y^*\| \leq M\|y^k - y^*\|^2$, and is said to converge superlinearly if $\lim_{k \to \infty} \left(\|y^{k+1} - y^*\|/\|y^k - y^*\|\right) = 0$. Studies about the rate of convergence include the local (and global) quadratic convergence in the active-set methods in [65, 28], the local linear or super-linear convergence in the sequential quadratic programming algorithm in [8] under different conditions, and the superlinear convergence of the interior point algorithm in [102, 75].

2 Parametric linear complementarity constrained program

In this section, we discuss the parametric linear complementarity constrained program and the methodologies for solving it with regard to the number of design parameters. A linearly constrained program with parametric linear complementar-
CHAPTER 1.

...ity constraints is formulated as:

\[
\min_{x, y} \quad f(x, y) \\
\text{subject to} \quad g_i(x, y) = 0, \quad \forall i \in \text{SideConstraints}_E, \\
g_I(x, y) \geq 0, \quad \forall i \in \text{SideConstraints}_I, \\
\text{and} \quad 0 \leq y_j \perp h_j(x, y) \geq 0, \quad \forall j \in \text{ComplementarityConstraints},
\]

where \( g \) and \( h \) are linear. The variable \( x \) is referred to as the design parameter because it is designed by the optimal structure

\[
\min f(x, y) \text{ subject to } g_i(x, y) = 0 \text{ and } g_I(x, y) \geq 0,
\]

and it becomes a fixed parameter for the resulting linear complementarity problem

\[
0 \leq y_j \perp h_j((x), y) \geq 0.
\]

To adapt to the linearity of the side constraints and the complementarity constraints, we let \( x \in \mathbb{R}^n, y \in \mathbb{R}^m \), and the cardinality of the sets \( \text{SideConstraints}_E \) and \( \text{SideConstraints}_I \) be \( k_E \) and \( k_I \) respectively, and use a matrix representation to rewrite the parametric program (25) as:

\[
\begin{align*}
\min_{x, y} & \quad f(x, y) \\
\text{subject to} & \quad A_E x + B_E y + r_E = 0, \\
& \quad A_I x + B_I y + r_I \geq 0, \\
& \quad 0 \leq y_j \perp N x + M y + q \geq 0,
\end{align*}
\]

where \( A_E \in \mathbb{R}^{k_E \times n}, B_E \in \mathbb{R}^{k_E \times m}, r_E \in \mathbb{R}^{k_E}, A_I \in \mathbb{R}^{k_I \times n}, B_I \in \mathbb{R}^{k_I \times m}, r_I \in \mathbb{R}^{k_I}, N \in \mathbb{R}^{m \times n}, M \in \mathbb{R}^{m \times m}, \) and \( q \in \mathbb{R}^m \).

We inspect this problem by parts and first start from the parametric linear complementarity problem (LCP)

\[
0 \leq y \perp N x + M y + q \geq 0.
\]

If \( n = 0 \), i.e., \( N, A_E, A_I \) and \( x \) are null, the feasible problem (26) is affected by the structure of the \( M \) matrix. If \( M \) is positive definite, the solution \( y \) is unique;
if $M$ is positive semi-definite, the set of solutions is a convex polyhedron, whose representation is available. Provided a solution $y$ to the LCP $0 \leq y \perp My + q \geq 0$ is known, one representation for this polyhedron is of the following form as in [31]:

$$
y \geq 0, \ My + q \geq 0,
q^T(y - \bar{y}) = 0,
(M + M^T)(y - \bar{y}) = 0.
$$

(27)

This representation allows us to substitute the complementarity constraints by easier linear constraints. Alternatively, a polyhedral representation for the degenerate variables is shown in [98]:

$$
y_\rho \geq 0, \ M_\bullet_\rho y_\rho + \bar{q} \geq 0,
(M_\rho_\rho + M_\rho_\rho^T)y_\rho = 0.
$$

(28)

where the matrix $\bar{M}$ and vector $\bar{q}$ are the transformation of $M$ and $q$ respectively in the final tableau of Lemke’s algorithm, and the index set $\rho$ collects the indices of the degenerate variables, that is, $i \in \rho$ if and only if $y_i = 0$ and $(My + q)_i = 0$. The algorithm that uses the representation (28) in [98] was proposed to find all the basic solutions to the complementarity when $M$ is positive semidefinite. In fact, the class of matrix such that the set of solutions is convex is referred to as the column sufficient matrix. This is a wider class which contains all positive semidefinite matrices ([31]). For the matrix $M$ violating column sufficiency, the complexity of finding a solution to an LCP is NP-hard, equivalent to a nonconvex quadratic program [20]. Several algorithms are applicable to find at least one solution to (26) regardless of the class of $M$. They are the principle pivoting method [31], the Lemke’s algorithm for an LCP, and algorithm of the Newton types (e.g. PATH [35]).

If $n > 0$ in (26), only the case with a special matrix $M$ is of interest. In other cases of $M$, the parametric LCP (26) can be treated as a non-parametric LCP. Suppose we fix the parameter at $\bar{x}$, and assume $\bar{y}$ be the unique corresponding solution to the fixed-parameter LCP $0 \leq y \perp M\bar{y} + (q + N\bar{x}) \geq 0$, then $(\bar{x}, \bar{y})$ solves the parametric LCP (26). The active sets of the linear complementarity depending on the value of parameter $\bar{x}$ are the index sets $\Lambda_L(\bar{x})$ and $\Lambda_R(\bar{x})$ defined
by:

\[ A_L(\mathbf{x}) = \{ j \in \{1, \ldots, m \} | \gamma_j = 0 \}, \]

\[ A_R(\mathbf{x}) = \{ j \in \{1, \ldots, m \} | (N\mathbf{x} + M\mathbf{y} + q)_j = 0 \}. \]

A region in the space of \( \mathbf{x} \) is called an \textit{invariancy region} or \textit{critical region}, denoted by \( \mathcal{IR} \), such that every \( \mathbf{x} \in \mathcal{IR} \) has the same resulting active set pair \((A_L(\mathbf{x}), A_R(\mathbf{x}))\).

There are existing methodologies to identify the invariancy regions in the parameter space corresponding to every possible combination of active set pairs. When \( n = 1 \), the invariancy region \( \mathcal{IR} \) is an interval \( [\mathbf{x}_l, \mathbf{x}_u] \) due to convexity.

The parametric principle pivoting method described in [31] identifies the invariancy line segments \( [\mathbf{x}_l, \mathbf{x}_u] \) and finds the solution \( \mathbf{y}(\mathbf{x}) \) systematically. This method is specifically good for a uni-parametric LCP. For the multi-parametric LCP with \( n \geq 2 \), the work [30] deals with the LCP with a column sufficient matrix \( \mathbf{M} \) and proposes an algorithm based on the graph search procedure to identify all critical regions. Their algorithm is said to be output-sensitive because the complexity is linear in the size of output, which is the numbers of the invariancy regions being identified.

On the other hand, consider the following quadratic program

\[
\min_{\mathbf{z}} \quad \frac{1}{2} \mathbf{z}^T \mathbf{Q} \mathbf{z} + \mathbf{c}^T \mathbf{z}
\]

subject to \( \mathbf{Gz} \geq \mathbf{h} \),

where \( \mathbf{Q} \) is symmetric and positive definite. It is well known that the sufficient optimality condition, the KKT condition, of this convex quadratic problem is a linear complementarity problem. Therefore, finding the solution and the invariancy regions for the following parametric form of (30):

\[
\min_{\mathbf{z}, \mathbf{d}, \mathbf{t}} \quad \frac{1}{2} \mathbf{z}^T \mathbf{Q} \mathbf{z} + (\mathbf{c} + \mathbf{Rd})^T \mathbf{z}
\]

subject to \( \mathbf{Gz} \geq \mathbf{h} + \mathbf{St} \)

belongs to category of the parametric LCP, where the vector \( \mathbf{u} = [\mathbf{d}, \mathbf{t}] \) is the parameter. The simplest parametric convex quadratic program with \( \mathbf{u} \in \mathbb{R}^2 \), \( \mathbf{d} \in \mathbb{R} \) and \( \mathbf{t} \in \mathbb{R} \) is studied in [52]. In this case, the the invariancy region is on a two-dimensional plane. The algorithm in [52] identifies the edges of the invariancy
regions via a counter-clockwise searching around an interior point in the region, and identifies the graphs of all the invariancy regions on the plane. For the multi-parametric convex quadratic program, the work [113] provides a valid polyhedral representation of the critical region in the “state space” since their motivating model is from the field of predictive control. Other works that contribute to the algorithm for the multi-parametric convex quadratic program include [99, 100]. All the aforementioned works show that the invariancy regions or the critical regions are convex polytopes. It is shown in [113] that the critical region is not full-dimensional if there is a degeneracy. This phenomenon implies that, for an \( IR \subseteq \mathbb{R}^2 \) example, the edge or the vertex of this invariancy region can itself be an invariancy region. In general, we notice that every algorithm that involves the graph traversal procedure is output-sensitive.

We now consider the existence of the side constraints

\[
A_{\mathcal{E}}x + B_{\mathcal{E}}y + r_{\mathcal{E}} = 0 \quad \text{and} \quad A_{\mathcal{I}}x + B_{\mathcal{I}}y + r_{\mathcal{I}} \geq 0
\]

and their effects on the feasible set. For the \( n = 0 \) case, if \( M \) is column sufficient, the feasible set in \( y \) defined by \( 0 \leq y \perp My + q \geq 0 \) and (32) is convex. For \( n > 0 \) and an arbitrary \( IR \), the feasible sets in \((x, y)\) are expressed as the following convex set:

\[
x \in IR,
A_{\mathcal{E}}x + B_{\mathcal{E}}y + r_{\mathcal{E}} = 0,
A_{\mathcal{I}}x + B_{\mathcal{I}}y + r_{\mathcal{I}} \geq 0,
y \geq 0,
Nx + My + q \geq 0,
y_j = 0, \quad \forall j \in A_L(x),
(Nx + My + q)_j = 0, \quad \forall j \in A_R(x).
\]

Thus, the feasible set of the multi-parametric linear complementarity constrained program is piecewise convex, and every invariancy region corresponds to one piece.

It remains to consider the objective function \( f(x, y) \) subject to the feasible re-
gion as a whole, if \( f(x, y) \) is a convex function. If \( n = 0 \), the (non-parametric) linear complementarity constrained program (25) is equivalent to the following problem:

\[
\min_{x, y} \quad f(x, y) \\
\text{subject to} \quad A\epsilon x + B\epsilon y + r\epsilon = 0, \\
A\tau x + B\tau y + r\tau \geq 0, \\
y \geq 0, \\
My + q \geq 0, \\
\text{and} \quad y^T My + q = 0.
\]

We can see that the difficulty of solving the non-parametric linear complementarity constrained program is the same as the quadratically constrained model (34) because they both carry the disjunctive nature. If \( n = 0 \) and \( M \) is positive semidefinite, the problem becomes a convex optimization problem subject to the convex feasible set. This type of problem is easier in the sense that a local optimal solution is the global optimal solution and that the solution set of \( 0 \leq y \perp My + q \geq 0 \) is a valid polyhedron, as in (27) and (28). If \( n = 0 \) but \( M \) is not special enough to form a convex solution set, the techniques (penalization, relaxation, binary variables, etc.) used for an MPEC that are summarized in Section 1 are applicable. Abundant analysis in the methodologies of the convex or nonconvex quadratic constraint programs (34) depending on the class of \( M \) can be found in the literature. Examples include the simplex-type method in [97], the branch-and-bound method in [9, 80], and a relaxation of the semidefinite programming type [70, 81, 11].

For the case \( n > 0 \) in a parametric linear complementarity constrained program (25) with convex \( f(x, y) \), the first way to solve it is by applying any of the invariancy region identifying algorithm in the aforementioned literature [31, 30, 52, 113, 99, 100]. These existing algorithms allow us to enumerate all invariancy regions and solve the following linearly constrained problem

\[
\min_{x, y} \quad f(x, y) \\
\text{subject to} \quad \text{constraints (33)}.
\]
inside each of them. The smallest objective value obtained from problems (35) is the final solution to the parametric linear complementarity constrained program (25). The second way to view and to solve the parametric linear complementarity constrained program (25) is via its equivalent formulation:

\[
\begin{align*}
\min_{x,y} & \quad f(x, y) \\
\text{subject to} & \quad A_E x + B_E y + r_E = 0, \\
& \quad A_I x + B_I y + r_I \geq 0, \\
& \quad y^T N x + y^T M y + y^T q \leq 0, \\
& \quad y \geq 0, \\
\end{align*}
\]

and

\[
\begin{align*}
\text{(36)} & \quad Nx + My + q \geq 0,
\end{align*}
\]

where \(M\) is positive semidefinite. Problem (36) is not a convex problem because of the existence of the \(y^T N x = \sum_i \sum_j (y_i N_{ij}) x_j\) term. Let \(z_j = \sum_i y_i N_{ij}\). We have \(y^T N x = \sum_j z_j x_j\) where \(z_j x_j\) are called bilinear terms. Some researchers proposed the algorithm for solving the nonconvex quadratic constraint by replacing the bilinear terms with some kind of the concave or convex envelope to form a convex approximation problem. This approximation problem is then embedded in a branch-and-bound scheme so the error of the convex approximation problem goes to zero along with the branching. Works applying some convexification techniques on nonconvex quadratic constraints in (36) while maintaining a tree of branching are found in [101, 9, 80].

Remarks: In Chapter 2-Section 3, Chapter 3-Section 3, and Chapter 3-Section 4, we propose three different algorithms for solving a biparametric linear complementarity constrained program. The algorithm proposed in Chapter 2-Section 3 can be categorized in the type of methodologies derived from the formulation (36). In this approach, we use the difference of convex trick to convexify the bilinear term and further make the approximations linear in a depth-first branch-and-bound scheme. The users are allowed to change the path of branching via the selection of a parameter pair \((s, t)\), described in Chapter 2-Section 2.3. The algorithm proposed in Chapter 3-Section 3 makes use of the convexity of the invariancy regions in the parameter plane, yet doesn’t aim at obtaining the exact invariancy region graphi-
cally; instead, the parameter plane is maintained as a series of rectangles (at the 1st stage of the algorithm) until signs for the existence of an invariancy region edge segment appear (the 2nd stage of the algorithm is then activated). Our numerical results confirm the output-sensitivity of this type of graph-searching algorithm.

3 Inverse optimization

One source of applications of a parametric complementarity constrained program is inverse optimization. The inverse problem in optimization is the parameter estimation or the model selection problem for a mathematical programming model. Let the mathematical model be $\mathcal{M}(\mathbf{p}, \mathbf{v}, s)$ where the attribute $\mathbf{p}$ is a vector of necessary parameters to define the model $\mathcal{M}$; $\mathbf{v}$ is the variable, and $s$ is the sense of optimization. We formulate the inverse optimization of $\mathcal{M}(\mathbf{p}, \mathbf{v}, s)$ as

$$\min_{\mathbf{p}, \mathbf{v}} \Principle(\mathbf{p}) \quad \text{subject to} \quad \mathcal{M}(\mathbf{p}, \mathbf{v}, s),$$

(37)

where the $\Principle(\mathbf{p})$ is a function based on which the parameter $\mathbf{p}$ is selected. The inverse problem (37) is a parametric complementarity constrained program if the optimality condition of $\mathcal{M}(\mathbf{p}, \mathbf{v}, s)$ can be expressed by the complementarity. In contrast to the inverse problem, solving the problem $\mathcal{M}(\mathbf{p}, \mathbf{v}, s)$ with assigned $\mathbf{p}$ and $s$ is referred to as the forward problem. We notice that several inverse problems have been formulated as an MPEC including applications in fracture mechanics [85], American options and pricing [62, 58], traffic and telecommunication networks [103], optimal cost functions for human arm movements [5], and support vector machine [72, 120].

Remarks: We study the inverse problem of the support vector machine regression in Chapter 3 and the inverse problem of pricing and purchasing under a pure characteristics demand framework in Chapter 4.
Chapter 2: Global Solution of Bi-Parametric Linear Complementarity Constrained Linear Programs

**Keywords:** linear complementarity constraints, bi-parametric program, global optimization

1 Introduction

The general formulation of a Linear Program with Linear Complementarity Constraint (LPCC) is of the form:

\[
\begin{align*}
\min_{y_{all}} & \quad c(y_{all}) \\
\text{subject to} & \quad g(y_{all}) \geq 0, \\
& \quad 0 \leq l(y_{all}) \perp r(y_{all}) \geq 0,
\end{align*}
\]

where the objective function \(c(y_{all})\), the constraints \(g(y_{all})\), and the functions \(l(y_{all})\) and \(r(y_{all})\) that are involved in the complementarity constraint are all linear functions. To represent the linearity, we consider the matrix representation of the para-
metric LPCC specified as follows

\[
\min_{w,x,y} \quad c^T x + d^T y,
\]

subject to \( Ax + By \geq f, \)

and \( 0 \leq y \perp w := Nx + My + q \geq 0, \)

where \( x \in \mathbb{R}^n, y \in \mathbb{R}^m, \) and \( w \in \mathbb{R}^m \) are variables; and \( c \in \mathbb{R}^n, d \in \mathbb{R}^m, A \in \mathbb{R}^{\ell \times n}, B \in \mathbb{R}^{\ell \times m}, f \in \mathbb{R}^\ell, N \in \mathbb{R}^{m \times n}, M \in \mathbb{R}^{m \times m}, \) and \( q \in \mathbb{R}^m \) are the given coefficients.

In this chapter, we aim at the problem (2) with \( x \in \mathbb{R}^2 \) and \( M \) positive semidefinite, though sometimes with specification in this introductory section we view the problem (2) in a more general way.

From the parameter-analytic point of view, \( x \) is considered the amount of disturbance occurring in the objective, the linear constraints, and the linear complementarity constraints, with the disturbance directions vectors or matrices \( c, A \) and \( N \) respectively. Previous studies regarding the parameter analysis for a general optimization problem have provided results on the relation between perturbation amount and optimal solution for many optimization problem types. The region in which the optimal partition, which is a partition of the indices at the optimality, remains unchanged when disturbance occurs, (e.g. the index sets of basic and non-basic variables of a linear program,) is referred to as invariancy region or critical region, whose dimension is decided by the number of independent disturbances. The simplest form of the invariancy region is an interval obtained from the sensitivity analysis on a linear program with only one perturbation at either the objective function or the constraints. Other parameter analysis can be found in [51], which studies the linear program with the independent disturbance occurring in the objective and the constraints, showing that the invariancy regions for this problem are mesh-like areas separated by vertical and horizontal lines in a two-dimensional region; the work [52], which studies the bi-parametric convex quadratic program and the optimal partition being investigated divides the indices into three sets, shows that the invariancy regions are convex and providing an algorithm with illustrative results to detect the boundary of an invariancy region and to transit to the adjacent regions.

One significant part in the target problem (2) is the parametric linear com-
The non-parametric LPCC:

\[
\begin{align*}
\min_{\mathbf{w}', \mathbf{y}} \quad & \mathbf{d}^T \mathbf{y} \\
\text{subject to} \quad & \mathbf{B} \mathbf{y} \geq \mathbf{f}, \\
\text{and} \quad & 0 \leq \mathbf{y} \perp \mathbf{w}' := \mathbf{M} \mathbf{y} + \mathbf{q} \geq 0
\end{align*}
\]

is potentially easier to solve for a local solution is also a global solution when \( \mathbf{M} \) is positive semi-definite. However, a parameter analysis that investigates the correspondence between the disturbance amount and the new solutions to the LPCC, which allows the user to extend the solution to the original problem (3) to the parametric problem (2), to our knowledge, is still lacking in the literature. Therefore, the algorithm to find the global optimal solution to the disturbed problem (2) can involve a time-consuming process of region-wise searching and comparing.

Obtaining or verifying the global optimality of the mathematical program with complementary constraint (MPEC) is not yet a widespread technique contained in any prevailing solvers. Methods that have been developed to solve the general LPCC to global optimal can be divided into two categories: integer-programming-based algorithms and global-optimization-based algorithms. The method which belongs to the former category relies on an equivalent integer-program-reformulation of (2)
as follows:

$$\min_{x, y, u, z} c^T x + d^T y$$

subject to

$$A x + B y \geq f,$$

$$\theta z \geq N x + M y + q \geq 0,$$

$$\theta (1 - z) \geq y \geq 0,$$

and

$$z \in \{0, 1\}^m,$$

where $\theta \in \mathbb{R}^{m \times m}$ is a diagonal matrix of large-valued parameters, and $z$ is a vector of binary variables. The branch-and-bound scheme in which the branching is done on the complementarity can be at least traced back to [12, 57]. Variations of the branch-and-bound method and applications of decomposition techniques can be found in recent research such as [10, 59, 120]. This type of methodologies usually involve different preprocessing techniques to shrink the gap between the upper and lower bound of the objective value by alternatively generating valid cuts and adding them back to re-solve the relaxed problem. Ways to enumerate the candidate solutions is a concern in this type of methodologies too. The thesis [120] proposed a McCormick-bound refinement in the preprocessing stage to improve valid lower bound in a two-stage branch-and-cut based algorithm for the LPCC. Their numerical results displayed decent convergent speed in the LPCC with up to 200 complementarity constraints. Applying the Benders decomposition in [59], an algorithm that doesn’t require the knowledge of $\theta$ is proven to terminate finitely with the global optimum, if it exists. In their implementation, a series of integer-satisfiability subproblems and linear-programming master problems are solved alternatively to generate and add ray cuts and point cuts on the fly.

On the other hand, the method which belongs to the category of global-optimization-based algorithms considers solving a general LPCC equivalent to a quadratically constrained problem (QCP). For the non-parametric LPCC, this equiv-
alent reformulation is:

\[
\begin{align*}
\min_{y,u} & \quad d^T y \\
\text{subject to} & \quad By \geq f, \\
& \quad 0 \leq y, My + q \geq 0, \\
& \quad y^T My + y^T q = 0.
\end{align*}
\] (5)

If \( M \) is positive semi-definite, the solution set of the QCP (5) is convex; and the quadratic equality \( y^T My + y^T q = 0 \) can be replaced by valid polyhedral representations ([98, 31]). If \( M \) is not of special class to form a convex solution set, branch-and-bound method and semi-definite programming relaxations are among the suitable methods to solve this problem. For the parametric LPCC, its equivalent QCP is the following:

\[
\begin{align*}
\min_{x,y,u} & \quad c^T x + d^T y \\
\text{subject to} & \quad Ax + By \geq f, \\
& \quad 0 \leq y, Nx + My + q \geq 0, \\
& \quad y^T Nx + y^T My + y^T q = 0.
\end{align*}
\] (6)

Usually the matrix \( M \) in the formulation (6) belongs to a special class. Otherwise, the problem can be formulated as the form (5) without parameters. We investigate model (6) with a positive semi-definite matrix \( M \). With this assumption, \( y^T Nx \) is the only term that disturbs the convexity. To solve this non-convex problem, the domain partitioning method [2, 80], the primal-dual method [48], and the difference of convex method [114] are some of the most widely-used techniques. The key ideas are twofold. One is to construct a sequence of upper and lower bounds of the global optimum through different types of subproblems. The other is to underestimate the non-convex term by a valid convex underestimator.

In this chapter, we present an algorithm belonging to class of global optimization and domain partitioning methods involving the solution of a sequence of the convex QCP or LP to find the global optimum of (6), which is equivalent to the global optimum of the parametric LPCC. We show that for a feasible bi-parametric
LPCC, when the algorithm terminates, the global optimality is verified.

In the implementation, the algorithm we are proposing relies on existing solvers to solve various types of subproblems. We use CPLEX [1] to do the job. CPLEX is known to robustly solve the linear program; however, from our numerical experiments, subproblems of convex QCP are not always solvable by CPLEX. Those difficult quadratic constraints are then all substituted by the linear approximation. On the other hand, nonlinear programming (NLP) algorithms that gained significant robustness and efficiency in finding a stationary point, including LOQO [116], KNITRO [22], FILTER [46], SNOPT [54], and many other advances [8, 28, 94], have been extended to adapt to the nature (specifically, the violation of the constraint qualifications) of a mathematical programs with complementarity constraints (MPCC). For example, [14] has extended LOQO with penalty method to treat MPCC as a special case of NLP; [45, 118] have equipped FILTER and KNITRO respectively with additional features in the interface to solve MPCC without guaranteeing global optimality. Among the aforementioned solvers, we use the solution provided by KNITRO as the initial upper bound in our algorithm. KNITRO is a solver that incorporate sequential quadratic programming and trust region techniques, and it provide feasible stationary solution of a general LPCC without proven optimality. To obtain insights about the gap between a general valid stationary solution and a global optimal solution, the global optimal solutions obtained from our algorithm are compared with local ones from KNITRO. The experiment shows that KNITRO’s upper bound coincides with the true optimal value for some of the instances but not all of them.

The application side of the LPCC often comes from a parameter selection procedure, a hierarchical-decision-making optimization problem, or a leader-follower Stackelberg game. Many results from the field of bi-level programming can be linked with the source problems of the LPCC, such as [17] studying the two-level linear source control problem and [36] studying the parameter estimation problem. [17, 24] have shown that the optimal solution to a linear bi-level program occurs at an extreme point of the constraint set of all variables, yet this nice property doesn’t hold for a nonlinear bi-level problem. Paper [55] proposed an algorithm which relaxes the feasible region by convex underestimator and processes as the
Figure 1: Geometrical meaning of the Support Vector Machine Regression, where $\|w_s\|$ is the margin size, $\varepsilon_e$ is the tube radius, and $(w_s, b_s)$ defines the regression hyperplane. (Note that what is shown in the figure is not an optimal hyperplane.)

framework of $\alpha$BB[2] to solve nonlinear bi-level programs to global optimum. Many bi-level global-optimization algorithms are reviewed in [47].

In this work we focus on the parametric LPCC with two parameters, i.e., $x \in \mathbb{R}^2$, and $N$ is not a null matrix. This is motivated from the cross-validated support vector machine regression parameters selection problem but can be applied to any bi-parametric LPCC. The cross-validated support vector machine regression problem and the relation between it and the bi-parametric LPCC are introduced in the following subsection. In Section 2, the techniques we employed to tackle the bi-parametric LPCC are introduced, including an aggregated convex relaxation of the complementarity constraints which can be embedded in a domain-partitioning (or branch-and-bound) scheme, a pair of artificial constants $(s, t)$ which changes the domain of partitioning and the searching paths, and a linearization trick on the quadratic term to allow subproblems being solved robustly. In Section 3, the whole algorithm is presented with the proof of global optimality. In Section 4, numerical experiments on bi-parametric high-dimensional LPCC with various structures are provided and analyzed.
1.1 Bi-parametric LPCC example: Cross-validated Support Vector Machine Regression

The support vector machine (SVM), often applied in machine learning and data mining, is a statistical method to classify or fit the observed data points with multiple features by a set of estimators. Known for robustness, unlike the standard regression method which geometrically identifies one hyperplane to fit the data points, the SVM identifies a tube (two parallel hyperplanes) to fit the data points, as shown in Figure 1. Every data point inside the tube is treated as having no residual (in regression) or no mis-classification (in classification). Given $\varepsilon_e$, the support $w_s$ and the bias $b_s$ are determined so that the absolute $\varepsilon_e$-residual is minimized and that separation $1/\|w_s\|$ is maximized. A comprehensive introduction on the SVM can be found in [69].

The $C_e$ and $\varepsilon_e$ selection problem for the SVM regression can be formulated as a bi-level optimization problem [72], capturing the cross-validation process of the parameter selection as follows:

$$\min_{C_e, \varepsilon_e, w_s, b_s} \sum_{i=n_d+1}^{n_d+m_d} |w_s^T x_d^i + b_s - y_d^i|$$

subject to $0 \leq C \leq C_e \leq \bar{C}$, $0 \leq \underline{\varepsilon} \leq \varepsilon_e \leq \bar{\varepsilon}$

and $(w_s, b_s) \in \arg \min_{w_s, b_s} \left\{ C_e \sum_{j=1}^{n_d} \max \left( |w_s^T x_d^j + b_s - y_d^j| - \varepsilon_e, 0 \right) + \frac{1}{2} w_s^T w_s \right\}$,

where

$w_s$: the support vector,
$b_s$: the bias,
$C_e$: the regularization parameter (or the penalty parameter),
$\varepsilon_e$: the tube size,
$(x_d^i, y_d^i), i = 1, \ldots, n_d$: training data points;
$i = n_d + 1, \ldots, n_d + m_d$: testing data points,
$\underline{C}, \bar{C}$: the lower and upper limit for $C_e$ respectively, and
$\underline{\varepsilon}, \bar{\varepsilon}$: the lower and upper limit for $\varepsilon_e$ respectively.

The inner optimization problem in (7) is a standard SVM regression problem
that minimizes the weighted absolute $\varepsilon$-residual of the training data and the reverse of the margin size, $\|w_s\|$. At the inner level, the weight $C_e$ and the tube size $\varepsilon_e$ are inherited from the outer optimization problem. The outer optimization problem minimizes the absolute residual of testing data, while $(w_s, b_s)$ minimizes the inner optimization problem, and $C_e$ and $\varepsilon_e$ are selected in the outer optimization problem within a box-restricted range of values. By introducing a variable $e_s$, the inner optimization problem can be linearized as follows:

$$
\begin{array}{ll}
\min_{e_s, b_s, w_s} & C_e \sum_{j=1}^{n_d} e_{sj} + \frac{1}{2} w_s^T w_s \\
\text{subject to} & -(x_d^j)^T w_s - b_s + y_{dj} \geq -e_{sj} - \varepsilon_e, \forall j = 1, \ldots, n_d, \\
& (x_d^j)^T w_s + b_s - y_{dj} \geq -e_{sj} - \varepsilon_e, \forall j = 1, \ldots, n_d, \\
& e_{sj} \geq 0, \forall j = 1, \ldots, n_d.
\end{array}
$$

(8)

Since the objective function in (8) is convex and the constraints are linear, the Karush-Kuhn-Tucker (KKT) condition is necessary and sufficient for optimality. The inner problem can be replaced by the KKT condition:

$$
\begin{array}{ll}
C_e - \alpha_j - \beta_j - \gamma_j = 0, \forall j = 1, \ldots, n_d, \\
w_s - \sum_{j=1}^{n_d} (\beta_j - \alpha_j)x_d^j = 0, \\
\sum_{j=1}^{n_d} \alpha_j - \sum_{j=1}^{n_d} \beta_j = 0, \\
0 \leq e_{sj} + \varepsilon_e - (x_d^j)^T w_s - b_s + y_{dj} \perp \alpha_j \geq 0, \forall j = 1, \ldots, n_d, \\
0 \leq e_{sj} + \varepsilon_e + (x_d^j)^T w_s + b_s - y_{dj} \perp \beta_j \geq 0, \forall j = 1, \ldots, n_d, \\
0 \leq e_{sj} \perp \gamma_j \geq 0, \forall j = 1, \ldots, n_d.
\end{array}
$$

(9)

By further introducing a variable $p \in \mathbb{R}^{md}$ and replacing $\gamma_j$ and $w_s$ by expressions
of $\alpha_j, \beta_j,$ and $C_e$, the bi-level problem (7) is equivalent to:

\[
\begin{align*}
\min_{b_s, C_e, \varepsilon, p_j, \alpha_j, \beta_j} & \sum_{j=n_d+1}^{n_d+m_d} p_j \\
\text{subject to} & \quad 0 < C \leq C_e \leq \bar{C}, \\
& \quad 0 < \varepsilon \leq \varepsilon_e \leq \bar{\varepsilon}, \\
& \quad \forall j = n_d+1, \ldots, n_d + m_d: \\
& \quad (x^j_d)^T \sum_{i=1}^{n_d} (\beta_i - \alpha_i) x^i_d + b_s - y_{dj} \leq p_j, \\
& \quad -((x^j_d)^T \sum_{i=1}^{n_d} (\beta_i - \alpha_i)) x^i_d - b_s + y_{dj} \leq p_j,
\end{align*}
\]

and \forall j = 1, \ldots, n_d :

\[
\begin{align*}
0 \leq e_{s_j} + \varepsilon - (x^j_d)^T \sum_{i=1}^{n_d} (\beta_i - \alpha_i) x^i_d - b_s + y_{dj} & \perp \alpha_j \geq 0, \\
0 \leq e_{s_j} + \varepsilon + (x^j_d)^T \sum_{i=1}^{n_d} (\beta_i - \alpha_i) x^i_d + b_s - y_{dj} & \perp \beta_j \geq 0, \\
0 \leq C_e - \alpha_j - \beta_j & \perp e_{s_j} \geq 0, \\
0 = \sum_{j=1}^{n_d} \alpha_j - \sum_{j=1}^{n_d} \beta_j & \perp b_s \text{ free.}
\end{align*}
\]

Note that we denote the initial index of $p$ by $n_d + 1$ rather than 1 to represent the consistency with the indices of the testing data points. The last constraint has made the feasible region a mixed complementarity problem. Thus, (10) relates
to a bi-parametric LPCC, which is slightly different from (2),

$$\begin{align*}
\min_{w, x, y, u} & \quad c^T x + d^T y + g^T u \\
\text{subject to} & \quad Ax + By + Eu \geq f, \\
& \quad y \perp w := Nx + My + q, \\
& \quad y_j \geq 0, \forall j = 1, \ldots, 3n_d, \\
& \quad w_j \geq 0, \forall j = 1, \ldots, 3n_d, \\
& \quad y_{3n_d+1} : free, \\
\text{and} & \quad w_{3n_d+1} = 0,
\end{align*}$$

where $x \in \mathbb{R}^2$, $y \in \mathbb{R}^{3n_d+1}$, $u \in \mathbb{R}^{m_d}$, $c \in \mathbb{R}^2$, $d \in \mathbb{R}^{3n_d+1}$, $g \in \mathbb{R}^{m_d}$, $A \in \mathbb{R}^{(2m_d+4) \times 2}$, $B \in \mathbb{R}^{(2m_d+4) \times (3n_d+1)}$, $E \in \mathbb{R}^{(2m_d+4) \times m_d}$, $f \in \mathbb{R}^{2m_d+4}$, $N \in \mathbb{R}^{(3n_d+1) \times 2}$, $M \in \mathbb{R}^{(3n_d+1) \times (3n_d+1)}$, and $q \in \mathbb{R}^{3n_d+1}$. These matrices of coefficient are specified as follows:

$$x = [C_c \ v_e]^T,$$

$$y = [\alpha_{j=1}^{n_d} \ | \ \beta_{j=1}^{n_d} \ | \ e_{s=1}^{n_d} \ | \ b_s]^T,$$

$$u = [p_{j=n_d+1}^{n_d+m_d}]^T,$$

$$N = \begin{pmatrix}
0_{n_d \times 1} & 1_{n_d \times 1} \\
0_{n_d \times 1} & 1_{n_d \times 1} \\
1_{n_d \times 1} & 0_{n_d \times 1} \\
0 & 0
\end{pmatrix}.$$
\[ M = \begin{pmatrix}
M_1 & M_2 & I_{n_d \times n_d} & -I_{n_d \times 1} \\
M_3 & M_4 & I_{n_d \times n_d} & 0_{n_d \times 1}
\end{pmatrix}, \]

where \( i, j \): index of row, column of \( M \)

\[ M_1 = (x_d^i x_d^j)_{n_d \times n_d}, \]
\[ M_2 = (-x_d^i x_d^{j-n_d})_{n_d \times n_d}, \]
\[ M_3 = (-x_d^{i-n_d} x_d^j)_{n_d \times n_d}, \]
\[ M_4 = (x_d^{i-n_d} x_d^{j-n_d})_{n_d \times n_d}, \]

\[ q = [y_{d1}^{j=1} \ldots y_{dn_d}^{j=1} \ldots y_{d1}^{j=n_d} \ldots y_{dn_d}^{j=n_d} | 0_{1 \times n_d} | 0]_T, \]

\[ A = \begin{pmatrix}
1 & 0 \\
-1 & 0 \\
0 & 1 \\
0 & -1 \\
0_{2m_d \times 1} & 0_{2m_d \times 1}
\end{pmatrix}, \]

\[ B = \begin{pmatrix}
0_{4 \times n_d} & 0_{4 \times n_d} & 0_{4 \times n_d} & 0_{4 \times 1} \\
B_1 & B_2 & 0_{m_d \times n_d} & -1_{m_d \times 1} \\
B_3 & B_4 & 0_{m_d \times n_d} & 1_{m_d \times 1}
\end{pmatrix}, \]

where \( i, j \): index of row, column of \( B \)

\[ B_1 = (x_d^{i-4+n_d} x_d^j)_{m_d \times n_d}, \]
\[ B_2 = (-x_d^{i-4+n_d} x_d^{j-n_d})_{m_d \times n_d}, \]
\[ B_3 = (-x_d^{i-4} x_d^j)_{m_d \times n_d}, \]
\[ B_4 = (x_d^{i-4} x_d^{j-n_d})_{m_d \times n_d}, \]
\[ E = \begin{pmatrix} 0_{4 \times m_d} \\ I_{m_d \times m_d} \\ I_{m_d \times m_d} \end{pmatrix} , \]

\[ f = \left[ C - \bar{C} \bar{\varepsilon} - \bar{\varepsilon} \mid -y_d^{i=n_d+1} \ldots -y_d^{i=n_d+m_d} \mid y_d^{i=n_d+1} \ldots y_d^{i=n_d+m_d} \right]^T , \]

\[ c = [0_{1 \times 2}]^T , \]

\[ d = [0_{1 \times (3n_d+1)}]^T , \text{ and} \]

\[ g = [1_{1 \times m_d}]^T . \]

The vector \( y \) is mainly a stack of the multipliers and the \( x \) vector is the design variable in the upper level optimization problem. The extra variable \( u \) in this example is resulting from the absolute-valued objective function of the upper level optimization, and \( u \) only involves in the side constraints. Even though the model (11) has a mixed complementarity portion and extra variables \( u \) to differ from the model (2), the structure of the equality \( y^T N x + y^T M y + y^T q = 0 \) is not harmed. Thus, the convex relaxations on \( y^T N x \), which will be seen in the following sections, are still applicable to model (11).

2 Solution techniques

The main techniques used in developing the algorithm discussed in this section are aggregated convex relaxation, depth-first domain partitioning scheme, changing plane method, and linearization.
2.1 Aggregated convex relaxation

Consider the parametric LPCC in (2) with \( x \in \mathbb{R}^2 \). By aggregating the component-wise complementarities and relaxing the equality, (2) becomes

\[
\begin{align*}
\min_{x, y, z} & \quad c^T x + d^T y \\
\text{subject to} & \quad A x + B y \geq f, \\
& \quad 0 \leq y, N x + M y + q \geq 0, \\
& \quad y^T M y + y^T q \leq x^T z, \\
& \quad z^T = -y^T N,
\end{align*}
\]

where \( z \in \mathbb{R}^2 \), and \( M \) is assumed to be positive semi-definite in our framework. The challenge lies with the non-convex term \( x^T z \). Suppose the bounds on \( x \) and \( z \) are available- \( x \in [l_x, u_x] \) and \( z \in [l_z, u_z] \). An upper bound that can be imposed on the bilinear term \( x_i z_i \) is the McCormick bound, which is a pair of linear inequalities obtained from the Taylor expansion around \((u_x, l_z)\) and \((l_x, u_z)\):

\[
\begin{align*}
x_i z_i & \leq x_i l_z + z_i u_x - l_z u_x, \quad x_i z_i \leq x_i u_z + z_i l_x - u_z l_x, \\
\end{align*}
\]

This upper bound is proven to be a concave envelope of the bilinear term ([4, 3]).

Another upper bound on the bilinear term is the following. \( \forall i = 1, 2 \):

\[
x_i z_i = \frac{1}{4} (x_i + z_i)^2 - \frac{1}{4} (x_i - z_i)^2 \\
\leq \frac{1}{4} [(u_{x_i+z_i} + l_{x_i+z_i})(x_i + z_i) - (l_{x_i+z_i}, u_{x_i+z_i})] - \frac{1}{4} (x_i - z_i)^2. \tag{14}
\]

This upper bound is obtained by rewriting the bilinear term as the difference of two squared terms and replacing the first squared term by an upper linear approximation to ensure the convexity in (12), while no changes are made on the second squared term. Geometrically, this upper linear approximation is the secant line of the squared function \((x_i + z_i)^2\) with two intersection points at \( x_i + z_i = u_{x_i+z_i} \) and \( x_i + z_i = l_{x_i+z_i} \).

The two upper bounds (13) and (14) perform differently on different values of \((x_i, z_i)\). As a demonstration, let \( 10 \leq x_1 \leq 20 \) and \( 30 \leq z_1 \leq 40 \), the differences
between the true value $x_1z_1$ and the two types of approximations are shown in Figures 2 and 3.

Figure 2 plots 10000 values of $\min(x_1l_{z_1} + z_1u_{x_1} - l_{z_1}u_{x_1}, x_1u_{z_1} + z_1l_{x_1} - u_{z_1}l_{x_1}) - x_1z_1$ at the corresponding coordinates of $(x_1, z_1)$ that compose a $100 \times 100$ grid on the plane of $[l_{x_1}, u_{x_1}] \times [l_{z_1}, u_{z_1}]$. Similarly, Figure 3 plots 10000 values of $\{\frac{1}{4}[(u_{x_1+z_1} + l_{x_1+z_1})(x_1 + z_1) - (l_{x_1+z_1}u_{x_1+z_1})] - \frac{1}{4}(x_1 - z_1)^2\} - x_1z_1$ at the same coordinates of $(x_1, z_1)$. The condition under which the approximation (13) exactly equals to the bilinear term is:

$$\{x_i = l_{x_i}\} \cup \{x_i = u_{x_i}\} \cup \{z_i = l_{z_i}\} \cup \{z_i = u_{z_i}\}, \quad (15)$$

whereas the condition under which the approximation (14) exactly equals the bilin-
ear term is:
\[(x_i + z_i = l_{x_i+z_i}) \cup (x_z + z_i = u_{x_i+z_i}).\]
\[(16)\]

The satisfiable solution set to (16) is a subset of the to (15). The largest approximation error of (13) in Figure 2 is 25, and it occurs at
\[
\{(x_1, z_1) \mid x_1 = 0.5(u_{x_1} + l_{x_1}), \ z_1 = 0.5(u_{z_1} + l_{z_1})\};
\]

the approximation error of (14) in Figure 3 is also 25, which occurs at
\[
\{(x_1, z_1) \mid x_1 + z_1 = 0.5(l_{x_1+z_1} + u_{x_1+z_1})\}.
\]

For these 10000 simulated data points, the average errors for the approximations (13) and (14) are 8.1667 and 20.7492 respectively.

Since there is only one inequality involved, we choose to employ the upper bound (14) rather than (13) to replace the bilinear term in (12) in the domain-partitioning algorithm. Namely, by aggregating variables, the dimension of the domain in which the partition is done is reduced from four \((x_1, z_1, x_2, z_2)\)-axis) to two \((x_1 + z_1, x_2 + z_2)\)-axis). Also reducing the dimension of the domain, using only one of the inequalities in (13), however, is not preferred because the maximum error is much larger than that obtained from (14). The drawback is that the approximation error doesn’t reduce to zero at all limit points. For example, if the second inequality in (13) is chosen, the discrepancy is the greatest at \((x_i, z_i) = (u_{x_i}, l_{z_i})\).

**Proposition 1.** The maximal approximation error for (14) is:
\[
\frac{(u_{x_i+z_i} - l_{x_i+z_i})^2}{16}, \forall i = 1, 2.
\]
\[(17)\]

**Proof.** The maximum value occurs at the mid point. By replacing \(x_i + z_i\) with \((u_{x_i+z_i} + l_{x_i+z_i})/2\), we get
\[
\frac{1}{4}[(u_{x_i+z_i} + l_{x_i+z_i})^2 - l_{x_i+z_i}u_{x_i+z_i}] - \frac{1}{4}[(u_{x_i+z_i} + l_{x_i+z_i})^2 - l_{x_i+z_i}u_{x_i+z_i}]
\]
\[= \frac{(u_{x_i+z_i} - l_{x_i+z_i})^2}{16}.
\]
\[\square\]
2.2 Depth-first domain-partitioning scheme

In Proposition 1, we have shown that the approximation error is related to the bounds of the aggregated variables. This motivates the domain-partitioning scheme, which partitions the two dimensional \((x_1 + z_1\)-axis and \(x_2 + z_2\)-axis) plane to mesh-like areas. Each of the areas is defined by a set of bounds \((u_{x_1 + z_1}, l_{x_1 + z_1}, u_{x_2 + z_2}, l_{x_2 + z_2})\). In each area, a subproblem of (12) is formed by replacing the bilinear term with that of (14). When the areas become smaller and smaller, the approximation error in Proposition 1 approaches zero. There are three possible results of solving the subproblems.

1) The subproblem is infeasible. For this situation, the LPCC is also infeasible, and we no longer consider the corresponding range of the values as the possible solution.

2) The subproblem has an optimal solution, but the complementarity constraint \(w \perp y\) is not satisfied. For this case, the solution gives us a lower bound on the global optimum of the LPCC, and we keep partition this region.

3) The subproblem has an optimal solution, and the complementarity constraint \(w \perp y\) is satisfied. For this case, the solution gives us an upper bound on the global optimum of the LPCC, and we don’t need to further partition inside this region. The domain-partitioning scheme stops when the solution of result 3 in some region happens to give the smallest objective value among all the regions. More precisely, the stop criterion is a reasonably small gap between the upper bound and the lower bound of the LPCC.

At each iteration, one area is chosen to be partitioned into two, and two corresponding subproblems are solved. The region we choose is the one containing the solution that gives us the valid lower bound of the LPCC; that is, we choose to go depth first. If some points on the edges turn out to be the solution of the subproblem, the point has no approximation error, and thus a solution that satisfies the complementarity constraint is obtained. If the points on the edges are not feasible to the complementarity constraints, they will not be the solutions of the subproblems. Therefore, if we could cut off some low points, points which minimize the relaxed subproblems, that are non-complementarity-feasible by causing the partition to pass through them, the points will not be possible optimizers of the subsequent subproblems. We propose the partitioning criterion with the following considerations in that order, considering each before proceeding to the next:
1. Avoid the narrow region, i.e., the ratio of the longer side and the shorter side cannot be larger than some threshold. This is to eliminate the possible difficulty the solver may confront in solving QCPs. The threshold is chosen based on experience.

2. Make the point at which the previous solution occurred lie on the boundary. This is to avoid the previous solution point, which is in fact not complementarity feasible, again being the solution of a subproblem, given that consideration 1 has been satisfied.

3. Make the area of the two new regions less different. This is to balance the maximum approximation error as closely as possible for the two regions resulting from the partitioning, given that considerations 1 and 2 have been satisfied.

4. Make the shape of the region more close to a square. This is to avoid the large difference in the approximation errors between the first aggregated variable \((x_1 + z_1)\) and the second aggregated variable \((x_2 + z_2)\), given that the preceding three considerations have been satisfied. In the implementation, this consideration is achieved by cutting at the longer side.

Figure 4 gives a geometrical illustration of the partitioning process and strategy.

In implementation, the equality rarely holds. In practice, any of the following ways can be employed to specify the border line below which the complementarity violation is within an acceptable residual \((\text{precision})\) and thus can be treated as complementarity feasible.

1. Comparative: \(\max_i \min(w_i, y_i) < \text{precision}\)
2. Component-wise: \(\max_{i=1,...,m} w_i y_i < \text{precision}\)
3. Accumulative: \(\mathbf{w}^T \mathbf{y} < \text{precision}\)
4. Relative: \(\max_i \frac{\min(w_i, y_i)}{\max(\max_i(w_i, y_i), 1)} < \text{precision}\)
CHAPTER 2.

41

70

72

73

75

73

Iteration 0: Solve QCP or LP, and get the objective value 70. Check the complementarity at the point where 70 occurs.

Iteration 1: Cut at the previous checked-point. Cut vertically such that the new regions are less different in size. Check comp. at the point where the smaller obj. val. occurs.

Iteration 2: Choose the region with the smallest obj. to partition. Cut at the previous checked-point 72 vertically such that the new regions are less different in size. Check comp. at the point with smaller obj. val. 74.

Iteration 3: Go to Region 73. The previous checked-point doesn’t lie in this region, so we cut evenly at the longer side. Throw away the infeasible region.

Iteration 4: Go to Region 74. Cut evenly at longer side. Check comp. at Point 74.5.

Iteration 5: Go to Region 74.5. To avoid the narrow region, we cut horizontally.

Figure 4: A geometrical illustration of the partitioning process and strategy on $(x_1 + z_1, x_2 + z_2)$-plane.

2.3 Changing plane method

Given arbitrary scalars $s_i, t_i \in \mathbb{R} \forall i = 1, 2$ such that $s_i t_i > 0$, an upper bound for the bilinear term is the following:

\[
x_i z_i = \frac{1}{4s_i t_i} (s_i x_i + t_i z_i)^2 - \frac{1}{4s_i t_i} (s_i x_i - t_i z_i)^2 \\
\leq \frac{1}{4s_i t_i} [(u_i + l_i)(s_i x_i + t_i z_i) - (l_i u_i)] - \frac{1}{4s_i t_i} (s_i x_i - t_i z_i)^2,
\]

where $u_i$ and $l_i$ denote the upper and lower bound for $s_i x_i + t_i z_i$ respectively. Geometrically, employing (18) instead of (14) changes the plane in which the partition is done from the plane of $(x_1 + z_1, x_2 + z_2)$ to $(s_1 x_1 + t_1 z_1, s_2 x_2 + t_2 z_2)$. The maximum approximation error discussed in Proposition 1 becomes

\[
err_{(s_i x_i + t_i z_i)}^2 = \frac{(u_i - l_i)^2}{16s_i t_i}, \forall i = 1, 2.
\]

The constants $s_i$ and $t_i$ can be interpreted as the weight assigned to the variables $x_i$ and $z_i$ respectively. From experiments, we have found that the setting
of the pairs \((s_i, t_i)\) could affect the total runtime dramatically. Except for the basic setting of \((s_i, t_i) = (1, 1)\), one could set \(s_i > t_i\) with \(t_i = 1/s_i\), such as \((s_i, t_i) = (100, 0.01)\) or \((s_i, t_i) = (1000, 0.001)\). These two examples assign a large weight to the \(x\) variables. One could also set \(s_i = t_i\) but both \(s_i\) and \(t_i\) are larger than 1, such as \((s_i, t_i) = (100, 100)\) or \((s_i, t_i) = (1000, 1000)\), to form a large value of the denominator of term \((18)\). The term \((18)\) is an important factor to be monitored during the process of the algorithm for a reasonable small value of the term \((18)\) is a necessary condition at the termination. The effect of different pairs of \((s_i, t_i)\) can be seen from the error term \((18)\). For example, suppose \(s_i t_i = 1\), and without loss of generality, assume that the regions are partitioned evenly for the selected index, say \(i = 1\). Let the original difference of \(u_1 - l_1\) be the unit length. Then the error term evolves as 0.625, 0.015625, 0.0003906, 0.0000976563, 0.0000244141, 0.000006104, 0.000001526. In order to let \(err_{(sx+tz)}^2\) be reduced to less than 0.00005, it would require effort of solving \(1 + 2 + 4 + 8 + 16 + 32 + 64 = 127\) subproblems, assuming there is no infeasible region identified during the process. Suppose \(s_i t_i = 100\). If its corresponding length of \(u_1 - l_1\) is 10 units, the error evolution does not change, and we still need to solve 127 subproblems to reduce \(err_{(sx+tz)}^2\) to be smaller than 0.00005. (In fact, in experiments, the instances with the same initial errors but different scaling could affect the path of branching or bounding because we rely on the solver to generate solutions to subproblems. Here we only analyze at the expected value of \(err_{(sx+tz)}^2\) assuming there is only one path.) However, for some \((s_i, t_i)\) such that \(s_i t_i = 100\), there may be an initial \(u_1 - l_1\) at only 8 units. The error under this choice of \((s_i, t_i)\) progresses as 0.04, 0.01, 0.0025, 0.000625, 0.00015625, 0.00003906, meaning that we only need to solve \(1 + 2 + 4 + 8 + 16 + 32 = 63\) subproblems. The number of rectangles which possess the above approximation errors grows exponentially along with number of iterations.

The reduction of the initial approximation error could significantly speed up the runtime. The best way to select the \((s_i, t_i)\) pair is to assure that \((18)\) is the smallest among all the other choices of \((s_i, t_i)\) before the partition scheme has started. However, there is no way to find such an optimal \((s_i, t_i)\) pair since \(u_i\) and \(l_i\) can only be obtained with fixed \((s_i, t_i)\). Therefore, testing on several different \((s_i, t_i)\) pairs and selecting the best among them is necessary for solving the problem efficiently.
It is noteworthy that the \((s_i, t_i)\) pair which results in a smaller value of the formula (18) does not imply that the constraints \(l_i \leq s_i x_i + t_i z_i \leq u_i\) are tighter.

### 2.4 Linearization

There are some reasons to completely linearize the model (12): First, in the numerical experiments, the solver (CPLEX) has difficulties in solving some QCPs. Second, for cases in which the QCP is solved successfully, it sometimes takes much more time than solving an LP. Given some valid value of \(sx^* - tz^*\) and \(y^*\), the convex terms can be linearized as follows:

\[
y^T M y \geq y^* M y^* + 2(y - y^*)^T \left( \frac{M + M^T}{2} \right) y^*, \forall y^*, (20)
\]

\[
\frac{1}{4s_it_i}(s_i x_i - t_i z_i)^2 \geq \frac{1}{4s_it_i}[2(s_i x_i^* - t_i z_i^*)(s_i x_i - t_i z_i) - (s_i x_i^* - t_i z_i^*)^2], \forall s_i x_i^* - t_i z_i^*, \forall i = 1, 2. (21)
\]

We will consider two sources of valid values of \(sx^* - tz^*\) and \(y^*\): break-even points that divide the feasible range and the historical data of the local solutions obtained from the subproblems. Basically, linearizing at the break-even points controls the approximation errors, and linearizing at the historical data of the solutions avoids a revisit of the infeasible optimizer. Let \(u_{sx_i-tz_i}\) and \(l_{sx_i-tz_i}\) be the upper bound and the lower bound of \(s_i x_i - t_i z_i, \forall i = 1, 2\) respectively, and \(u_y\) and \(l_y\) the upper bound and lower bound of \(y\), the set of break-even points \((BEPSet)\) is defined as:

\[
BEPSet_{sx-tz}(num) = \left\{ (sx - tz) \left| sx - tz = l_{sx-tz} + r \left( \frac{u_{sx-tz} - l_{sx-tz}}{num} \right) \right., \right. \\
\left. \left. r = 0, 1, ..., num, num \geq 1 \right\}, (22) \right.
\]

\[
BEPSet_{y}(num) = \left\{ y \left| y = l_y + r \left( \frac{u_y - l_y}{num} \right), r = 0, 1, ..., num, num \geq 1 \right\}. (23)
\]

The \(num\) in (22) and (23) represents the number of intervals resulting from the dividing. The length of the intervals is closely related to the linear approximation performance, whose error is quantified in the following proposition.

**Proposition 2.** Let the set of break-even points be generated as in (22) and (23),
the vector of the maximum errors resulting from the linearization (20) at all $y^* \in BEPSet_\gamma(num)$ is:

$$\text{err}_{y \cdot My}^\gamma = \frac{1}{4} \left( \frac{u_y - l_y}{num} \right)^T \left( \frac{M + M^T}{2} \right) \left( \frac{u_y - l_y}{num} \right)$$

$$+ \left( u_y - \frac{u_y - l_y}{2 \cdot num} \right) \left( \frac{M - M^T}{2} \right) \left( u_y - \frac{u_y - l_y}{2 \cdot num} \right) - u_y \left( \frac{M - M^T}{2} \right) u_y.$$

(24)

The maximum error resulting from the linearization (21) at all $sx^* - tz^* \in BEPSet_{sx-tz}(num)$ is:

$$\text{err}_{(sx-tz)}^\gamma = \frac{1}{16st} \left( \frac{u_{(sx-tz)} - l_{(sx-tz)}}{num} \right)^2, \forall i = 1, 2.$$  

(25)

**Proof.** For (24), consider the break-even points at $u_y$ and $u_y - \frac{u_y - l_y}{2 \cdot num}$. The maximum value occurs at the mid point $u_y - \frac{u_y - l_y}{2 \cdot num}$. Replacing $y^*$ by $u_y$ and replacing $y$ by $u_y - \frac{u_y - l_y}{2 \cdot num}$ in (20), we obtain

$$\left( u_y - \frac{u_y - l_y}{2 \cdot num} \right)^T M \left( u_y - \frac{u_y - l_y}{2 \cdot num} \right) - u_y^T M u_y + 2 \left( u_y - \frac{u_y - l_y}{2 \cdot num} \right) (M + M^T) u_y$$

$$= \left( u_y - \frac{u_y - l_y}{2 \cdot num} \right)^T \left( \frac{M + M^T}{2} + \frac{M - M^T}{2} \right) \left( u_y - \frac{u_y - l_y}{2 \cdot num} \right) - u_y^T \left( \frac{M + M^T}{2} + \frac{M - M^T}{2} \right) u_y$$

$$- \left( u_y - \frac{u_y - l_y}{num} \right)^T \left( \frac{M + M^T}{2} \right) u_y$$

$$= \frac{1}{4} \left( u_y - \frac{u_y - l_y}{num} \right)^T \left( \frac{M + M^T}{2} \right) \left( u_y - \frac{u_y - l_y}{num} \right) + \left( u_y - \frac{u_y - l_y}{num} \right) \left( \frac{M - M^T}{2} \right) u_y - \left( u_y - \frac{u_y - l_y}{num} \right)^T \left( \frac{M + M^T}{2} \right) u_y.$$

For (25), consider the break points at $u_{(sx-tz)}$ and $u_{(sx-tz)} - \frac{u_{(sx-tz)} - l_{(sx-tz)}}{2 \cdot num}$. The maximum value occurs at the mid point $u_{(sx-tz)} - \frac{u_{(sx-tz)} - l_{(sx-tz)}}{2 \cdot num}$. Replacing $sx_i^* - tz_i^*$ by $u_{(sx-tz)}$, and replacing $sx_i - tz_i$ by $u_{(sx-tz)} - \frac{u_{(sx-tz)} - l_{(sx-tz)}}{2 \cdot num}$ in (21), we obtain

$$\frac{1}{4st} \left( u_{(sx-tz)} - \frac{u_{(sx-tz)} - l_{(sx-tz)}}{2 \cdot num} \right)^2$$

$$- \frac{1}{4st} \left[ 2 u_{(sx-tz)} \left( u_{(sx-tz)} - \frac{u_{(sx-tz)} - l_{(sx-tz)}}{2 \cdot num} \right) \right] - u_{(sx-tz)}^2.$$

$$= \frac{1}{4st} \left( \frac{u_{(sx-tz)} - l_{(sx-tz)}}{4 \cdot num} \right)^2.$$

In the implementation of the algorithm, $BEPSet(num)$ can be updated at
every iteration by letting $num$ be an increasing function, such as the positive linear or the exponential function, of the count of the iterate. It is noteworthy that more the valid point of $sx^* - tz^*$ and $y^*$ does not imply better performance in identifying the global optimum, though theoretically the accuracy of the approximation is increased. Yet every extra break point contributes one additional linear constraint to the model, which may become a burden when there are already too many constraints. For this reason, the size of $BEPS(num)$ cannot be expanded without restriction.

The second source of the valid data points is the set which collects all the solutions that have been obtained from solving one of the subproblems and that have been shown not complementarity-feasible. The approximation error of the points around which the linearizing is done becomes zero. Therefore, getting stuck at a non-global optimal solution can be avoided.

## 3 Algorithm

Within each area of the partitioning scheme, a relaxed subproblem is defined by the boundary of the area and is solved to update the lower bound of the objective. The subproblem is either a quadratic constrained program (QCP) or a linear program (LP) whose formulation will be shown in Section 3.1. The complete algorithm will be shown in Section 3.2.

### 3.1 Subproblems

We define a parallelogram by a triple $(S_1, S_2, \theta)$, where $S_i$ is a tuple $(i, s_i, t_i, l_i, u_i)$ for $i = 1, 2$ and $\theta$ is the objective value obtained from solving the subproblem QCP or LP whose feasible region is defined by $S_1$ and $S_2$. The subproblems of QCP and LP that are used in the domain-partitioning algorithm are formally introduced in this subsection.

The subproblem of QCP is the following:
QCP\(\left(S_1, S_2\right)\):

\[
\theta = \min_{x, y, z, w, \mu} \ c^T x + d^T y
\]

subject to \(Ax + By \geq f\),

\(0 \leq y, w = q + Nx + My \geq 0\),

\(z^T = -y^T N\),

\[
\text{validLB} \leq c^T x + d^T y \leq \text{validUB},
\]

\[
0 \geq q^T y + y^T My + \sum_{i=1}^{2} \frac{1}{4s_i t_i} \left\{ \left( s_i x_i - t_i z_i \right)^2 \right. \\
\left. - \left[ \left( l_i + u_i \right) \left( s_i x_i + t_i z_i \right) - l_i u_i \right] \right\}, \forall i = 1, 2,
\]

and \(l_i \leq s_i x_i + t_i z_i \leq u_i, \forall i = 1, 2\),

where

\(l_i, u_i\): the lower and upper bound of \(s_i x_i + t_i z_i\), and

\(\text{validLB}, \text{validUB}\): the valid lower and the valid upper bound of the objective function \(c^T x + d^T y\).

In the implementation, if the solver fails to solve a subproblem of QCP, we solve an LP instead. The experiments also show that to reduce the time for the solver to try to solve difficult QCP, it is sometimes preferred to employ only the subproblems of LP after a certain number of iterations. The linearized subproblem obtained by the technique described in Section 2.4 is as follows:
\[ \theta = \min_{x, y, z, w, L_0, \nu, \omega} c^T x + d^T y \]

subject to \( Ax + By \geq f \),

\( 0 \leq y, w = q + Nx + My \geq 0 \),

\( z^T = -y^T N \),

\( l_i \leq s_i x_i + t_i z_i \leq u_i, \forall i = 1, 2 \),

\( validLB \leq c^T x + d^T y \leq validUB \),

\( 0 \geq L_0 + \sum_{i=1}^{2} (\omega_i + \nu_i) \),

\( L_0 - q^T y \geq 0 \),

\( L_0 \geq q^T y + y^* My^* + 2(y - y^*)^T \left( \frac{M + M^T}{2} \right) y^* \),

\( \forall y^* \in BEPSet_y(num) \cup PastSolutionSet \)

\( 0 \geq \nu_i, \forall i = 1, 2 \),

\( \nu_i \geq -\frac{1}{4s_i t_i} \left[ (l_i + u_i)(s_i x_i + t_i z_i) - l_i u_i \right], \forall i = 1, 2 \),

\( \omega_i \geq 0, \forall i = 1, 2 \),

\( \omega_i \geq \frac{1}{4s_i t_i} \left[ 2(s_i x_i^* - t_i z_i^*)^2(2(s_i x_i^* - t_i z_i^*)) - (s_i x_i^* - t_i z_i^*)^2 \right] \),

\( \forall (s_i x_i^* - t_i z_i^*) \in BEPSet_{sx-tz}(num) \cup PastSolutionSet, \forall i = 1, 2 \) \( (27) \)

We define a \textit{configuration} by a set of non-intersecting parallellograms such that the overall feasible region

\( l_i \leq s_i x_i + t_i z_i \leq u_i, \forall i = 1, 2 \) \( (28) \)

is covered by these parallellograms, and that the union of these parallellograms is equivalent to the region \( (28) \). The process of the domain-partitioning can be interpreted as generating a sequence of \textit{configurations} by eliminating one parallellogram.
from and adding two new ones to the configuration of the previous iteration.

To help reduce the feasible region, the following are some valid cuts that can be added to (26) and (27).

1. The McCormick bound on $y \cdot w$.

Considering the relaxation $y \cdot w \leq 0$, the following constraints are valid.

$$
\begin{align*}
1_y \cdot w + 1_w \cdot y - 1_y \cdot 1_w (\leq y \cdot w) & \leq 0, \\
u_y \cdot w + u_w \cdot y - u_y \cdot u_w (\leq y \cdot w) & \leq 0,
\end{align*}
$$

where $1_y, 1_w, u_y,$ and $u_w$ are known lower and upper bounds for $y$ and $w$. This natural cut has little effect to tightening the feasible region in our experimental instances.

2. Multiple $(s, t)$-cuts.

The scalars $s_i$ and $t_i$ which act for changing the plane where the partitioning is done usually can not be changed, once it is chosen at the beginning of the partitioning. However, various scalars $s'_i \neq s_i, t'_i \neq t_i$ and the corresponding bounds such that $l'_i \leq s'_i x_i + t'_i z_i \leq u'_i$ can define the following cuts:

$$
\mu_i \leq \frac{1}{4s'_i t'_i} \left\{ (s'_i x_i - t'_i z_i)^2 - [(l'_i + u'_i)(s'_i x_i + t'_i z_i) - l'_i u'_i] \right\} \forall i = 1, 2. \quad (30)
$$

The parameters $l'_i$ and $u'_i$ in these cuts can still be updated according to the current parallelogram $[l_1, u_1] \times [l_2, u_2]$, yet they do not necessarily define a series of configurations on the $s'_i x_i + t'_i z_i$ plane.

3. The McCormick bound on $x \cdot z$. (Valid when the sign information is known.)

If $x \cdot z \geq 0$ is known, the following constraints are valid.

$$
\begin{align*}
&u_x \cdot z + l_z \cdot x - u_x \cdot l_z \geq 0, \\
&1_x \cdot z + u_z \cdot x - 1_x \cdot u_z \geq 0.
\end{align*}
$$

If $x \cdot z \leq 0$ is known, the following constraints are valid.

$$
\begin{align*}
&1_x \cdot z + l_z \cdot x - 1_x \cdot l_z \leq 0, \\
&u_x \cdot z + u_z \cdot x - u_x \cdot u_z \leq 0.
\end{align*}
$$

$$
\begin{align*}
1_y \cdot w + 1_w \cdot y - 1_y \cdot 1_w (\leq y \cdot w) & \leq 0, \\
u_y \cdot w + u_w \cdot y - u_y \cdot u_w (\leq y \cdot w) & \leq 0,
\end{align*}
$$

where $1_y, 1_w, u_y,$ and $u_w$ are known lower and upper bounds for $y$ and $w$. This natural cut has little effect to tightening the feasible region in our experimental instances.
4. Refine valid objective lower bound using the McCormick bound ([120]).

This is a preprocessing technique. Starting with initial values of \( u_x, u_z, l_x, \) and \( l_z \), the following QCP is firstly solved.

\[
\begin{align*}
\bar{\theta} &= \min_{x, y, z, w, \mu} \ c^T x + d^T y \\
\text{subject to} & \quad A x + B y \geq f, \\
& \quad 0 \leq y, w = q + N x + M y \geq 0, \\
& \quad z^T = -y^T N, \\
& \quad \text{validLB} \leq c^T x + d^T y \leq \text{validUB}, \\
& \quad 0 \geq q^T y + y^T M y + \sum_{i=1}^{2} \mu_i, \\
& \quad l_{x_i} \leq x_i \leq u_{x_i}, \forall i = 1, 2, \\
& \quad l_{z_i} \leq z_i \leq u_{z_i}, \forall i = 1, 2, \\
& \quad 0 \leq l_{z_i} z_i + u_{z_i} x_i - l_{z_i} u_{z_i} + \mu_i, \forall i = 1, 2, \\
& \quad 0 \leq u_{x_i} z_i + l_{z_i} x_i - u_{x_i} l_{z_i} + \mu_i, \forall i = 1, 2, \\
& \quad 0 \geq l_{x_i} z_i + l_{z_i} x_i - l_{x_i} l_{z_i} + \mu_i, \forall i = 1, 2, \\
& \quad \text{and} \quad 0 \geq u_{x_i} z_i + u_{z_i} x_i - u_{x_i} u_{z_i} + \mu_i, \forall i = 1, 2.
\end{align*}
\]

If \( \bar{\theta} < \text{validUB} \), let \( \text{validUB} \leftarrow \bar{\theta} \) and update the corresponding constraint in (33). Now solve for the objective which minimizes and maximizes \( x_i \) and \( z_i \) subject to the updated constraints set in (33). This \( x - z \)-bound-updating step requires efforts of solving 8 QCPs. If \( \bar{u}_x, \bar{u}_z, \bar{l}_x, \) and \( \bar{l}_z \) improves, let \( u_x \leftarrow \bar{u}_x, u_z \leftarrow \bar{u}_z, l_x \leftarrow \bar{l}_x, \) and \( l_z \leftarrow \bar{l}_z \). The process continues by alternatively solving for updated bounds on \( x, z, \) and \( \text{validLB} \). Usually the first round of the cross-updating can improve the most, and then marginal effect declines in the subsequent rounds. We found this preprocessing technique could improve valid lower bound on objective function significantly, yet a improved lower bound helps little to expedite the algorithm, as latter shown in Section 4.6.

5. Refine the interval of \( (u_i - l_i) \) using the disjunctive cut.

This is also a preprocessing technique. We explain this technique with the
2-indices \((k \text{ and } k', \text{ where } k, k' \in \{1, \ldots, m\} \text{ and } k \neq k')\) and 4-pieces case which generates the \(\mathbb{QCP}_{\text{bounds}}(k, k', \text{pieceIndex}) \forall \text{pieceIndex } = 1, \ldots, 4\) of the following form:

\[
\mathbb{QCP}_{\text{bounds}}(k, k', \text{pieceIndex}):
\]

\[
\begin{align*}
& \min_{x,y,z,w} \max_{\phi} s_i x_i + t_i z_i \\
& \text{subject to } A x + B y \geq f, \\
& \phantom{\text{subject to }} 0 \leq y, w = q + N x + M y \geq 0, \\
& \phantom{\text{subject to }} z^T = -y^T N, \\
& \phantom{\text{subject to }} \text{validLB} \leq c^T x + d^T y \leq \text{validUB}, \\
& \phantom{\text{subject to }} 0 \geq q^T y + y^T M y + \sum_{j=1}^2 \frac{1}{4} \left\{ (x_j - z_j)^2 \\
& \phantom{\text{subject to }} \phantom{\frac{1}{4}} \cdot \left[ (l_{x_j+z_j} + u_{x_j+z_j})(x_j + z_j) - l_{x_j+z_j} u_{x_j+z_j} \right] \right\}, \forall j = 1, 2, \\
& \phantom{\text{subject to }} l_{x_j+z_j} \leq x_j + z_j \leq u_{x_j+z_j}, \forall j = 1, 2, \\
& \phantom{\text{subject to }} \text{and linear constraints } \in \text{piece}_{\text{pieceIndex}}(k, k').
\end{align*}
\]

(34)

The concept of “pieces” of two complementarity constraints \(0 \leq y_k \perp w_k \geq 0\) and \(0 \leq y'_k \perp w'_k \geq 0\) are sets of linear constraints defined as

\[
\begin{align*}
\text{piece}^1(k, k') & = \{y_k = 0, y'_k = 0, w_k \geq 0, w'_k \geq 0\}, \\
\text{piece}^2(k, k') & = \{y_k \geq 0, y'_k = 0, w_k = 0, w'_k \geq 0\}, \\
\text{piece}^3(k, k') & = \{y_k = 0, y'_k \geq 0, w_k \geq 0, w'_k = 0\}, \text{and} \\
\text{piece}^4(k, k') & = \{y_k \geq 0, y'_k \geq 0, w_k = 0, w'_k = 0\}.
\end{align*}
\]

We need to solve \(\frac{8 m (m-1)}{2}\) many problems of \(\mathbb{QCP}_{\text{bounds}}(k, k', \text{pieceIndex})\) to find all \(l^{k,k',\text{pieceIndex}}_i\) and \(u^{k,k',\text{pieceIndex}}_i\), and the refined \(l_i\) and \(u_i\) are determined by the largest \(l^{k,k',\text{pieceIndex}}_i\) and the smallest \(u^{k,k',\text{pieceIndex}}_i\) respectively. This step can reduce around one percent of the original \((u_i - l_i)\) in some instances, yet is very time-consuming.

The cuts mentioned above are optional in the implementation of the algorithm.
For some instances, the total number of required iteration is reduced and along with the overall run time. However, there are also some instances that require longer solving time because of the extra steps needed to solve for the bounds used in (29), (30), (31) or (32), (33), and (34).

### 3.2 Domain partitioning algorithm

The complete algorithm is the following.

**0. Initialize.**

Set $PastSolutionSet = \emptyset$.

Set $BEPSet = \emptyset$.

Set $FeasibleParallelogram = \emptyset$.

Set $CompFeaLowerBound = -\infty$.

Set $validLB = -\infty$.

Set $validUB$ = the objective value provided by KNITRO. If the KNITRO solution is not available, set it to be $\infty$.

Set $l_{x+z}$ and $u_{x+z}$ = lower and upper bound of $x+z$ respectively.

Set the following parameters$^2$:

- $Precision = 10^{-4}$,
- $FrontIteration = 10$,
- $maxRatio = 10$,
- $BEPSetLimit = 50$,
- $s_j = 100, t_j = 0.01, \forall j = 1, 2$
- $D$ = dimension of $w$ = dimension of $y (= 100)$,
- $J$ = dimension of $x$ = dimension of $z = 2$, and
- $N$ = dimension of $f (= 100)$.

optional: Set $l_y, u_y, l_w, u_w = lower bound of y, upper bound of y, lower bound of w, and upper bound of w$ respectively.

**1. Generate $S-T$ cuts.**

$^2$We mark the initial setting used in solving the instance R100_1 in the adjacent bracket.
For \( k = 1, 2 \), solve

\[
\frac{l_k}{u_k} = \min \frac{\max}{x, y, x, w, \phi} s_k x_k + t_k z_k
\]

subject to

\[
Ax + By \geq f,
\]

\[
0 \leq y, w = q + Nx + My \geq 0,
\]

\[
z^T = -y^TN,
\]

\[
validLB \leq c^T x + d^T y \leq validUB,
\]

\[
0 \geq q^T y + y^T My + \sum_{j=1}^{2} \frac{1}{4} \left\{ (x_j - z_j)^2 - \left[ (l_j x_j + u_j x_j) x_j + z_j - l_j x_j + z_j \right] \right\},
\]

\[
\forall j = 1, 2,
\]

\[
l_j x_j + z_j \leq x_j + z_j \leq u_j x_j + z_j, \quad \forall j = 1, 2,
\]

and optional : (29) – (32).

Let \( S_1 \leftarrow \{(1, s_1, t_1, l_1, u_1)\}, S_2 \leftarrow \{(2, s_2, t_2, l_2, u_2)\} \).

2. **Solve First QCP.**

Solve subproblem QCP(\( S_1, S_2 \)). Let the objective value be \( \theta \).

Check the complementarity (\( \max_{d \leq D} w_d y_d < Precision \)). If satisfied, go to 7.

If not satisfied, collect the solution \( x^{0*}, y^{0*}, z^{0*} \) to the set \( PastSolutionSet \).

Let validLB \leftarrow \theta.

Let \( FeasibleParallelogram \leftarrow FeasibleParallelogram \cup \{(S_1, S_1, \theta)\} \).

Loop for \( i = 1 \ldots T \):

3. **Partition Into Two.**

(a) Find \( Parallelogram P_{ib} = (S_{ib}, S_{ib}, \theta_{ib}) \in FeasibleParallelogram \) whose 3rd component is the smallest.

(b) If \( CompFeaLowerBound \neq \emptyset \), find \( \theta_j^* = \min_{\theta_j \in CompFeaLowerBound} \theta_j \). If \( \theta_j^* = \theta_{ib} \), go to 7.

(c) For \( P_{ib} \):

If \( l_j \leq s_j x_j^{i-1} + t_j z_j^{i-1} \leq u_j, \forall (j, s_j, t_j, l_j, u_j) \in S_{jib} \),

select \( j^* \) such that the ratio of errors between two branches is the smaller, i.e.,
\[
\begin{align*}
&\max \left\{ u_{j^*} - s_{j^*} x_{j^*}^{(i-1)*} - t_{j^*} z_{j^*}^{(i-1)*}, s_{j^*} x_{j^*}^{(i-1)*} + t_{j^*} z_{j^*}^{(i-1)*} - l_{j^*} \right\} \\
&\min \left\{ u_{j^*} - s_{j^*} x_{j^*}^{(i-1)*} - t_{j^*} z_{j^*}^{(i-1)*}, s_{j^*} x_{j^*}^{(i-1)*} + t_{j^*} z_{j^*}^{(i-1)*} - l_{j^*} \right\} \\
&< \max \left\{ u_j - s_j x_j^{(i-1)*} - t_j z_j^{(i-1)*}, s_j x_j^{(i-1)*} + t_j z_j^{(i-1)*} - l_j \right\} \\
&\min \left\{ u_j - s_j x_j^{(i-1)*} - t_j z_j^{(i-1)*}, s_j x_j^{(i-1)*} + t_j z_j^{(i-1)*} - l_j \right\}, \forall j \neq j^*.
\end{align*}
\]

For \( j = j^* \), let
\[
S_{j_{part1}}^i = \left( j^*, s_{j^*}, t_{j^*}, l_{j^*}, s_{j^*} x_{j^*}^{(i-1)*} + t_{j^*} z_{j^*}^{(i-1)*} \right),
\]
\[
S_{j_{part2}}^i = \left( j^*, s_{j^*}, t_{j^*}, l_{j^*} + u_{j^*}, u_{j^*} \right).
\]

For \( j \neq j^* \), let
\[
S_{j_{part1}}^i = S_{j_{part2}}^i = S_{jl^*}.
\]

Otherwise

Select \( j^* \) such that \( u_{j^*} - l_{j^*} \geq u_j - l_j, \forall j \neq j^* \).

For \( j = j^* \), let
\[
S_{j_{part1}}^i = \left( j^*, s_{j^*}, t_{j^*}, l_{j^*} + u_{j^*}, \frac{l_{j^*} + u_{j^*}}{2} \right),
\]
\[
S_{j_{part2}}^i = \left( j^*, s_{j^*}, t_{j^*}, \frac{l_{j^*} + u_{j^*}}{2}, u_{j^*} \right).
\]

For \( j \neq j^* \), let
\[
S_{j_{part1}}^i = S_{j_{part2}}^i = S_{jl^*}.
\]

(d) Avoid narrow regions.

If
\[
\frac{\max_{j=1,2} u_j - l_j}{\min_{j=1,2} u_j - l_j} > \text{maxRatio}, \forall l_j, u_j \in S_{j_{part1}}.
\]
CHAPTER 2.

or if
\[
\frac{\max_{j=1,2} u_j - l_j}{\min_{j=1,2} u_j - l_j} > \text{maxRatio}, \forall l_j, u_j \in S_{j_{part2}},
\]

select \(j^*\) such that \(u_{j^*} - l_{j^*} \geq u_j - l_j, \forall l_j, u_j \in S_{j_{part1}}\).

For \(j = j^*\),
\[
S^i_{j_{part1}} \leftarrow (j^*, s_{j^*}, t_{j^*}, l_{j^*}, \frac{l_{j^*} + u_{j^*}}{2}),
\]
\[
S^i_{j_{part2}} \leftarrow (j^*, s_{j^*}, t_{j^*}, \frac{l_{j^*} + u_{j^*}}{2}, u_{j^*}).
\]

For \(j \neq j^*\),
\[
S^i_{j_{part1}} = S^i_{j_{part2}} \leftarrow S_{j_{lb}}.
\]

4. Solve for Parallelogram \(P_i^k\).

For \(k = \text{part1}, \text{part2}:\)

If \(i < \text{FrontIteration},\)
solve subproblem \(\text{QCP}^i_k(S_{1_k}^i, S_{2_k}^i)\).
(If \(\text{QCP}^i_k(S_{1_k}^i, S_{2_k}^i)\) is not solvable, solve subproblem \(\text{LP}^i_k(S_{1_k}^i, S_{2_k}^i)\) instead.)

Otherwise,
solve subproblem \(\text{LP}^i_k(S_{1_k}^i, S_{2_k}^i)\).

If feasible, let the objective value obtained be \(\hat{\theta}^i_k\).

5. Record.

\(\forall k = \text{part1}, \text{part2}:\)
(a) If \(\text{LP}^i_k(S_{1_k}^i, S_{2_k}^i)\) or \(\text{QCP}^i_k(S_{1_k}^i, S_{2_k}^i)\) is feasible,

let \(\text{FeasibleParallelogram} \leftarrow \text{FeasibleParallelogram} \cup \{(S_{1_k}^i, S_{2_k}^i, \hat{\theta}^i_k)\}\.\)

Remove \(P_{lb}\) from \(\text{FeasibleParallelogram}\).

(b) Select \(k^*\) such that
\[
\hat{\theta}^{i}_{k^*} < \hat{\theta}^{i}_k, \forall k \neq k^*.
\]

Let \(x^{i^*}, y^{i^*}, z^{i^*}\) be the solution at which the objective value \(\hat{\theta}^{i}_{k^*}\) is obtained.

Let \(\text{PastSolutionSet} \leftarrow \text{PastSolutionSet} \cup \{(x^{i^*}, y^{i^*}, z^{i^*})\}.\)
55

Let validLB ← smallest θ among the FeasibleParallelogram

6. **Check complementarity.**

   (a) If \( \max_{d\leq D} w_d^* y_d^* < \text{Precision} \), let

   \[
   \text{CompFeaUpperBound} \leftarrow \text{CompFeaUpperBound} \cup \{ \hat{\theta}_{k*}' \}.
   \]

   (b) Optional: If size of \( \text{BEPSet} < \text{BEPSetLimit} \), let

   \[
   \text{BEPSet} \leftarrow \text{BEPSet}(i + 1) \text{ or } \text{BEPSet} \leftarrow \text{BEPSet}(2^i)
   \]

   as defined in (22) and (23).

   (c) Go to 3. continue the loop.

7. **Terminate- solve purified LP.**

   Create \( \text{ActiveSet} \) such that: for \( d = 1 \ldots D \), if \( y_d^* < w_d^* \), \( d \in \text{ActiveSet} \).

   Solve the following purified LP.

   \[
   \theta_p = \min_{x,y,w} \quad c^T x + d^T y
   \]

   subject to \( Ax + By \geq f, \)

   \[
   y_d = 0, \quad \forall d \in \text{ActiveSet},
   \]

   \[
   w_d = (q + Nx + My)_d = 0, \quad \forall d \notin \text{ActiveSet}.
   \]

   \( \theta_p \) is the global optimum of the LPCC (2).

The following theorem shows that the above algorithm finds a global optimal solution if it exists.

**Theorem 3.** If there exists a global optimum for the LPCC (2), the algorithm is convergent to the global optimum of the LPCC in a finite number of iterations with the specified precision on the complementarity feasibility. Precisely, the following three properties hold:

1. The solution obtained from this algorithm is the global optimum.

2. There exists a configuration such that the solution where the smallest objec-
active value among all parallelograms meets the global optimal solution of the LPCC.

3. This method of partitioning finds a configuration satisfying the condition in 2 in a finite number of iterations.

Proof.

1. If $y, w, x$ and $z$ obtained from solving any subproblems satisfy the complementarity condition $\max_{d=1...D} w^*_d y^*_d$ (or any of the conditions specified in Section 2.2), the solution is feasible to the LPCC (2), and thus it gives us a valid upper bound since the feasible region of subproblems is restricted. Among all the valid upper bounds, the smallest one is the global upper bound for the LPCC (2). On the other hand, among all the objective value obtained in each parallelogram for a configuration, the smallest objective value is the valid lower bound for the current relaxed problem defined by the current configuration. If this valid lower bound satisfies the complementarity condition, the valid lower bound is also the global upper bound, and it gives us the global optimal solution.

2. Suppose precision is set at $\varepsilon$. The total error of a fixed subproblem of LP for the approximation to the equality $y^T M_y + y^T q - x^T z = 0$ can be expressed as:

$$
err_{total} = err_{y^T M_y} + \sum_{i=1}^{2} err_{(sx-tz)^2_i} + \sum_{i=1}^{2} err_{(sx+tz)^2_i}.
$$

The terms $err_{y^T M_y}$ and $err_{(sx-tz)^2_i}, \forall i = 1, 2$ are related to the exact value of the matrix $M$ and $u_y$, and $M$ and $u_y$ are finite. In Step 6 of the algorithm, the break points of the linear approximation of the first two error terms are generated. There exists an iterate index $i*$ such that after iteration $i*$, $err_{y^T M_y} < \varepsilon/5$ and $err_{(sx-tz)^2_i} < \varepsilon/5, \forall i = 1, 2$. Since the break point generation function can be arbitrary, we simply assume that $err_{y^T M_y} < \varepsilon/5$ and $err_{(sx-tz)^2_i} < \varepsilon/5, \forall i = 1, 2$. For the third error term, if

$$
u_i - l_i < 4 \sqrt{\frac{s_i l_i \varepsilon}{5}}
$$

(36)
then \( \text{err}_{(sx+tz)}^2 < \varepsilon/5, \forall i = 1,2 \). Thus, for such a parallelogram, if the subproblem is solved to optimum with the solution \( x^*, y^*, z^* \), then

\[
y^T M y^* + y^T q - x^T z^* < \varepsilon
\]  

(37)

This implies \( w^T y^* < \varepsilon \) and \( \max_{d=1,\ldots,D} w_d^T y_d^* < \varepsilon \). On the other hand, assuming the solution of the subproblem does not satisfy (37), i.e., this parallelogram does not contain global optimum, then there exists an integer \( \delta > \varepsilon \) such that \( 0 \geq L_0 + \sum_{i=1}^{2} (\omega_i + \nu_i) \). But such a solution violates the constraint \( 0 \geq L_0 + \sum_{i=1}^{2} (\omega_i + \nu_i) \) in (27), which implies that the solution does not exist. Regardless of the partitioning strategy, we conclude that if a parallelogram contains the global optimum, it will be identified. If a parallelogram satisfies the condition (36) but it does not contain the global optimum, the subproblem will be infeasible. Suppose the original upper and lower bounds on \( s_i x_i + t_i z_i \) are \( u_i^0 \) and \( l_i^0 \), for all \( i = 1,2 \). There exists \( \lambda_1, \lambda_2 \in \mathbb{R}^+ \) such that \( \frac{u_i^0 - l_i^0}{2^{i-1}} < 4\sqrt{\frac{s_i t_i}{5}} \) and \( \frac{u_i^0 - l_i^0}{2^{i-2}} < 4\sqrt{\frac{s_i t_i}{5}} \). Consider the configuration such that all parallelograms are of equal size \( \frac{u_i^0 - l_i^0}{2^{i-1}} \times \frac{u_i^0 - l_i^0}{2^{i-2}} \).

This configuration is a desired one.

3. Without loss of the generality, we assume that at each iteration the region is partitioned evenly into two, and that the \( s_1 x_1 + t_1 z_1 \)-axis and \( s_2 x_2 + t_2 z_2 \)-axis are cut alternatively. The configuration described in 2. can be reached after \( \sum_{i=0}^{\lambda_1+\lambda_2-1} 2^i \) iteration.

\[ \Box \]

The configuration in the second part of the proof of Theorem 3 describes the finest grid containing the most number of the parallelograms among all configurations that identify the global optimum successfully. The required number of iterations, as shown in the third part the proof of Theorem 3, is bounded by \( \sum_{i=0}^{\lambda_1+\lambda_2-1} 2^i \). In experiments, many fewer iterations are required to find the global optimum. For example, if \( u_i^0 - l_i^0 = 100, s_i t_i = 1, \) and \( \varepsilon = 0.0001 \), then \( \lambda_1, \lambda_2 = 13 \). Theoretically, the worst case for the algorithm to converge requires 67,108,863 iterations. The number of iterations is relevant to the initial bounds on \( s_i x_i + \)
Generating tight bounds on $t_i z_i$, which is problem dependent. Generating tight bounds on $s_i x_i + t_i z_i$ could be significant to the total time spent to identify the global optimum.

4 Numerical experiment

The algorithm is implemented in C++. We use CPLEX 12.2 to solve subproblems of QCP or LP. All the experiments are run on the machine in the ORLab in the department of Industrial and Enterprise Systems Engineering, with the following configuration: Dual Core AMD Opteron(tm) Processor 880, 2.4 GHz CPU, and 16 GB memory. We tested the algorithm on bi-parametric LPCC problems that were randomly generated and that were generated with special structures: $\mathbf{M}$ is skew symmetric; $\mathbf{B}$ is zero matrix; $\mathbf{M}$ has low (or high) rank, and $\mathbf{M}$ has low (or high) density. In addition, the effect of two important factors that affect the performance of the algorithm, the choices of the $(s_i, t_i)$ pairs and the bounds on some terms, are analyzed through the experiments.

4.1 $\mathbf{M}$ is randomly generated (positive semi-definite) matrix

To ensure there exists a feasible solution to the instances and there exists bounds for the variables $x, y,$ and $z$, we generate the bi-parametric LPCC problem data using Matlab by the following process. $m$ represents the dimension of $y$ and $w$, which is 100 and 300 in this class of instances. $\ell$ represents the number of side constraints.

1. Generate $X \sim$ random integer vector $\in [0, 100]^{2 \times 1}$, and $Y \sim$ random real vector $\in [0, 1]^{m \times 1}$.

   for $i = 1 \ldots m$
     
     if $Y(i) < 0.3$
     
     $Y(i) \leftarrow 0$;
     
     end
   

2. Generate $S \sim$ diagonal random real matrix with diagonal entries $\in [0, 1]^{m \times m}$, $P \sim$ random real matrix $\in [0, 1]^{m \times m}$, and $SS \sim$ random real matrix $\in [0, 1]^{m \times m}$. Compute $skewSym = SS - SS^T$ and $\mathbf{M} = PSP^T + skewSym$. 
3. Generate $N \sim \text{random integer matrix} \in [-50, 50]^{m \times 2}$.
   Compute $q = -(NX + MY)$.
   
   for $i = 1 \ldots m$
      if $Y(i) = 0$
         $q(i) \leftarrow q(i) + 10$;
      end
   end

4. Generate $A \sim \text{random integer matrix} \in [-50, 50]^{\ell \times 2}$, $B \sim \text{random real matrix} \in [0, 1]^{\ell \times m}$, $c \sim \text{random integer vector} \in [-50, 50]^{2 \times 1}$, and $d \sim \text{random integer vector} \in [-50, 50]^{m \times 1}$.

5. Compute $f = AX + BY - 1_{\ell \times 1}$.

6. Set $y_{max} = 1000$. Let
   
   $B \leftarrow \begin{bmatrix} B \\ -I_{m \times m} \end{bmatrix}$, $A \leftarrow \begin{bmatrix} A \\ 0_{m \times 2} \end{bmatrix}$, and $f \leftarrow \begin{bmatrix} f \\ -y_{max} \cdot 1_{m \times 1} \end{bmatrix}$.

The above process generates a feasible bi-parametric LPCC in which the $M$ matrix generated at the second step is positive semi-definite, dense, and is the sum of a symmetric matrix and a skew symmetric matrix. Without explicit restrictions in the generating process, in most of the cases, $M$ is full-rank and not symmetric. The $X$ and $Y$ generated at the first step is a feasible solution but not necessarily the global optimum. According to our way of generating, such a feasible solution is comprised of positive $X$ and nonnegative $Y$ (where 30 percent of elements are expected to be zero). For those indices $i$ such that $y_i$ is zero, $w_i > 0$. There is no degeneracy in the generation, but a non-degenerate optimal solution is not guaranteed. At the third step and the fifth step, feasible right-hand-sides corresponding to the complementarity constraints and side constraints are generated respectively. The instances contain an equivalent size of nontrivial side constraints and the complementarity constraints, and $y$ variables are bounded above by $y_{max}$, set at the sixth step.

The result appears in Table 1 with the following information: Column 1- Name of the instance; Column 2- total number of LPs solved throughout the algorithm,
Table 1: Global solution and algorithm performance on randomly generated problems.

<table>
<thead>
<tr>
<th>Prob Name</th>
<th>#LP</th>
<th>#QCP</th>
<th>#partitioning</th>
<th>time to reach 0.001 gap (sec)</th>
<th>total time (sec)</th>
<th>optimal obj val</th>
<th>KNITRO obj val</th>
<th>KNITRO time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>R100_1</td>
<td>380</td>
<td>20</td>
<td>128</td>
<td>59.86</td>
<td>72.03</td>
<td>2442.99</td>
<td>2443.00</td>
<td>9.47</td>
</tr>
<tr>
<td>R100_2</td>
<td>300</td>
<td>20</td>
<td>105</td>
<td>44.27</td>
<td>56.66</td>
<td>-4334.61</td>
<td>-4334.61</td>
<td>8.59</td>
</tr>
<tr>
<td>R100_3</td>
<td>320</td>
<td>18</td>
<td>102</td>
<td>50.59</td>
<td>56.23</td>
<td>-897.42</td>
<td>-897.42</td>
<td>10.13</td>
</tr>
<tr>
<td>R100_4</td>
<td>283</td>
<td>20</td>
<td>103</td>
<td>43.27</td>
<td>53.11</td>
<td>-3525.58</td>
<td>-3525.58</td>
<td>9.58</td>
</tr>
<tr>
<td>R100_5</td>
<td>228</td>
<td>20</td>
<td>80</td>
<td>35.99</td>
<td>41.42</td>
<td>-2347.63</td>
<td>-2347.63</td>
<td>9.32</td>
</tr>
<tr>
<td>average</td>
<td>302.2</td>
<td>19.6</td>
<td>103.6</td>
<td>46.796</td>
<td>55.89</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Prob Name</th>
<th>#LP</th>
<th>#QCP</th>
<th>#partitioning</th>
<th>time to reach 0.001 gap (sec)</th>
<th>total time (sec)</th>
<th>optimal obj val</th>
<th>KNITRO obj val</th>
<th>KNITRO time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>R300_1</td>
<td>281</td>
<td>19</td>
<td>102</td>
<td>941.82</td>
<td>1255.7</td>
<td>-976.27</td>
<td>-976.27</td>
<td>135.76</td>
</tr>
<tr>
<td>R300_2</td>
<td>541</td>
<td>19</td>
<td>178</td>
<td>2259.49</td>
<td>2475.93</td>
<td>307.89</td>
<td>307.90</td>
<td>140.70</td>
</tr>
<tr>
<td>R300_3</td>
<td>442</td>
<td>19</td>
<td>158</td>
<td>1795.55</td>
<td>2047.24</td>
<td>-1616.42</td>
<td>-1616.41</td>
<td>143.96</td>
</tr>
<tr>
<td>R300_4</td>
<td>281</td>
<td>18</td>
<td>110</td>
<td>1102.00</td>
<td>1319.63</td>
<td>-940.78</td>
<td>-940.77</td>
<td>130.92</td>
</tr>
<tr>
<td>R300_5</td>
<td>244</td>
<td>24</td>
<td>96</td>
<td>991.89</td>
<td>1102.75</td>
<td>-614.79</td>
<td>-614.78</td>
<td>138.24</td>
</tr>
<tr>
<td>average</td>
<td>357.8</td>
<td>19.8</td>
<td>128.8</td>
<td>1418.15</td>
<td>1640.25</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

including those in the preprocessing and the partitioning scheme; Column 3- total number of QCPs solved and those attempted but failed to be solved throughout the algorithm; Column 4- total times of partitioning, which is also the total number of iterations; Column 5- total time in seconds to find a valid lower bound that is within 1 percent of the gap compared with the valid upper bound; Column 6- total runtime in seconds to find and to verify the global optimum, including input/output time from/to files; Column 7- the global optimal objective value; Column 8- an upper bound of the problem provided by KNITRO; Column 9- total time for KNITRO to find the upper bound stated in Column 8.

The results in Table 1 are obtained from letting \((s_i, t_i) = (100, 0.01)\), which outperforms the cases where \((s_i, t_i) = (1, 1), (10, 0.01), \) and \((1000, 0.001)\). Moreover, all the optional constraints and steps were turned off.

KNITRO did well in these five instances by providing an upper bound that meets the exact optimal value. The extra run-time taken by our algorithm is considered as the cost of verifying global optimality.
M is 100x100 skew symmetric matrix

<table>
<thead>
<tr>
<th>Prob Name</th>
<th>#LP</th>
<th>#QCP</th>
<th>#partitioning</th>
<th>time to reach 0.001 gap (sec)</th>
<th>total time (sec)</th>
<th>optimal obj val</th>
<th>KNITRO obj val</th>
<th>KNITRO time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>S100_1</td>
<td>195</td>
<td>4</td>
<td>94</td>
<td>N/A</td>
<td>25.61</td>
<td>-7354.50</td>
<td>-4070.57</td>
<td>146.21</td>
</tr>
<tr>
<td>S100_2</td>
<td>277</td>
<td>4</td>
<td>131</td>
<td>N/A</td>
<td>42.47</td>
<td>-733.38</td>
<td>1145.43</td>
<td>139.24</td>
</tr>
<tr>
<td>S100_3</td>
<td>229</td>
<td>4</td>
<td>112</td>
<td>N/A</td>
<td>30.54</td>
<td>-1024.31</td>
<td>-365.09</td>
<td>129.82</td>
</tr>
<tr>
<td>S100_4</td>
<td>413</td>
<td>4</td>
<td>202</td>
<td>N/A</td>
<td>65.76</td>
<td>-2249.99</td>
<td>-1785.94</td>
<td>178.15</td>
</tr>
<tr>
<td>S100_5</td>
<td>273</td>
<td>4</td>
<td>134</td>
<td>N/A</td>
<td>42.73</td>
<td>-4634.73</td>
<td>-4200.31</td>
<td>120.369</td>
</tr>
<tr>
<td><strong>average</strong></td>
<td>277</td>
<td>4</td>
<td>135</td>
<td>N/A</td>
<td><strong>41.42</strong></td>
<td><strong>-1616.91</strong></td>
<td><strong>-4200.31</strong></td>
<td><strong>120.369</strong></td>
</tr>
</tbody>
</table>

M is 300x300 skew symmetric matrix

<table>
<thead>
<tr>
<th>Prob Name</th>
<th>#LP</th>
<th>#QCP</th>
<th>#partitioning</th>
<th>time to reach 0.001 gap (sec)</th>
<th>total time (sec)</th>
<th>optimal obj val</th>
<th>KNITRO obj val</th>
<th>KNITRO time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>S300_1</td>
<td>665</td>
<td>4</td>
<td>311</td>
<td>N/A</td>
<td>4885.53</td>
<td>-3842.27</td>
<td>-1605.37</td>
<td>146.21</td>
</tr>
<tr>
<td>S300_2</td>
<td>849</td>
<td>4</td>
<td>409</td>
<td>N/A</td>
<td>6575.4</td>
<td>1930.75</td>
<td>2715.62</td>
<td>139.24</td>
</tr>
<tr>
<td>S300_3</td>
<td>533</td>
<td>4</td>
<td>259</td>
<td>N/A</td>
<td>2728.25</td>
<td>-1494.92</td>
<td>449.52</td>
<td>129.82</td>
</tr>
<tr>
<td>S300_4</td>
<td>373</td>
<td>4</td>
<td>182</td>
<td>N/A</td>
<td>1915.62</td>
<td>-3915.73</td>
<td>-1616.91</td>
<td>178.15</td>
</tr>
<tr>
<td>S300_5</td>
<td>975</td>
<td>4</td>
<td>443</td>
<td>N/A</td>
<td>4420.05</td>
<td>-3842.27</td>
<td>-1605.38</td>
<td>120.37</td>
</tr>
<tr>
<td><strong>average</strong></td>
<td>679</td>
<td>4</td>
<td>320.8</td>
<td>N/A</td>
<td><strong>4104.97</strong></td>
<td><strong>-1605.38</strong></td>
<td><strong>120.37</strong></td>
<td></td>
</tr>
</tbody>
</table>

Table 2: Global solution and algorithm performance on problems with skew symmetric matrix M.

4.2 M is skew symmetric

The data is generated similarly as the steps in Section 4.1, except that the step 2 is replaced by:

2. Generate \( SS \sim \text{random real matrix} \in [0,1]^{m \times m} \). Compute \( \text{skewSym} = SS - SS^T \), and \( M = \text{skewSym} \).

The \((s_i, t_i)\) pairs being selected here are \((100, 0.01)\), and all optional steps are turned off. The results are shown in Table 2.

When M is skew symmetric, \( y^TMy = 0 \). A result with shorter run time for this set of the LPCC is obtained at the 100-complementarity instances, but longer run time is needed at the 300-complementarity instances, compared with the problems in Section 4.1. Surprisingly, the true optimal values for this set of problems are all far from the upper bound provided by KNITRO. This huge gap suggests that the price of converging to a local optimum can be very worthy.
Table 3: Global solution and algorithm performance on problems with null matrix B.

<table>
<thead>
<tr>
<th>Prob Name</th>
<th>#LP</th>
<th>#QCP</th>
<th>#partitioning</th>
<th>time to reach 0.001 gap (sec)</th>
<th>total time (sec)</th>
<th>optimal obj val</th>
<th>KNITRO upper bound</th>
<th>KNITRO time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>B100_1</td>
<td>479</td>
<td>4</td>
<td>147</td>
<td>74.03</td>
<td>83.22</td>
<td>2417.66</td>
<td>2417.66</td>
<td>4.33</td>
</tr>
<tr>
<td>B100_2</td>
<td>461</td>
<td>4</td>
<td>198</td>
<td>79.91</td>
<td>79.91</td>
<td>-4342.03</td>
<td>-4319.81</td>
<td>2.05</td>
</tr>
<tr>
<td>B100_3</td>
<td>439</td>
<td>4</td>
<td>145</td>
<td>68.87</td>
<td>73.67</td>
<td>-941.01</td>
<td>-941.01</td>
<td>2.56</td>
</tr>
<tr>
<td>B100_4</td>
<td>449</td>
<td>4</td>
<td>151</td>
<td>59.52</td>
<td>77.74</td>
<td>-3560.71</td>
<td>-3560.71</td>
<td>2.61</td>
</tr>
<tr>
<td>B100_5</td>
<td>583</td>
<td>4</td>
<td>186</td>
<td>62.47</td>
<td>101.42</td>
<td>-2368.67</td>
<td>-2368.67</td>
<td>3.80</td>
</tr>
<tr>
<td>average</td>
<td>482.2</td>
<td>4</td>
<td>165.4</td>
<td>68.96</td>
<td>83.192</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Prob Name</th>
<th>#LP</th>
<th>#QCP</th>
<th>#partitioning</th>
<th>time to reach 0.001 gap (sec)</th>
<th>total time (sec)</th>
<th>optimal obj val</th>
<th>KNITRO upper bound</th>
<th>KNITRO time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>B300_1</td>
<td>1317</td>
<td>1424</td>
<td>916</td>
<td>N/A</td>
<td>21698.7</td>
<td>-2008.27</td>
<td>-508.03</td>
<td>38.4338</td>
</tr>
<tr>
<td>B300_2</td>
<td>2233</td>
<td>1394</td>
<td>1274</td>
<td>N/A</td>
<td>21545.8</td>
<td>-900.22</td>
<td>-536.05</td>
<td>56.4395</td>
</tr>
<tr>
<td>B300_3</td>
<td>928</td>
<td>84</td>
<td>389</td>
<td>N/A</td>
<td>3177.17</td>
<td>-2266.66</td>
<td>-2101.32</td>
<td>37.6448</td>
</tr>
<tr>
<td>B300_4</td>
<td>2761</td>
<td>1439</td>
<td>1563</td>
<td>N/A</td>
<td>21346.2</td>
<td>-2435.81</td>
<td>-2268.34</td>
<td>32.2837</td>
</tr>
<tr>
<td>B300_5</td>
<td>3380</td>
<td>1426</td>
<td>1410</td>
<td>N/A</td>
<td>21848.1</td>
<td>-904.33</td>
<td>-836.82</td>
<td>41.9084</td>
</tr>
<tr>
<td>average</td>
<td>2124</td>
<td>1153</td>
<td>1110</td>
<td>N/A</td>
<td>17923.19</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

4.3 B is null matrix

In this subsection, instances have exactly the same c, d, A, N, M, and q with those in Section 4.1, but B is replaced by null matrix, and f is adjusted to make Ax + By ≥ f feasible. The results are shown in Table 3.

This class of problem is the hardest to solve among other problem structures. We are not able to initiate the partitioning process with a very good (s,t) pairs, while (100, 0.01) for 100-complementarity instances and (10000, 5) for 300-complementarity instances are the current choices in obtaining the result. We think that this class of problem causes difficulty to the algorithm because there is no side constraints on y variables, and thus may decrease the percentage of domain identified infeasible and eliminated. With the same reason, we intended to let yTMy term be captured accurately, and solved more subproblems of QCP. KNITRO’s valid upper bounds meet global optimum in 100-complementarity instances, but left a huge gap in 300-complementarity instances.
4.4 \textbf{M} has different rank and density

We test our algorithm on the LPCC problems generated by Yu [120]. This set of LPCCs is comprised of problems with 100, 150, or 200 complementarity constraints and 20 additional linear constraints, where the matrices M have different rank and density. In the experiments, two levels of rank (low and high) and two levels of density (low and high) are considered. For problems with 100, 150, and 200 complementarity constraints, the low level of rank is set to be 30, 30, and 30, and the high level of rank is set to be 60, 100, and 120. For problems with all three possible dimensions of the complementarities, the low level of density is 0.2 (i.e., 80 percent of elements of M is 0), and the high level of density is 0.7 (i.e., 30 percent of elements of M is 0). The results are shown in Table 4.

The results in Table 4 are the best performance among several arbitrary choices of the \((s_i, t_i)\) pair. We didn’t actually run through all possible combinations of \((s_i, t_i)\) since it’s very costly. The averages over the same ranks or the densities showed the trend that the easiest class of instances to solve is of low-rank and low-density.

4.5 Effect on different choices of \((s, t)\)

Table 5 shows collected results for some sample problems that are run multiple times using different choices of the \((s, t)\) pairs. For simplicity, we set \(s_1 = s_2 = s\) and \(t_1 = t_2 = t\). The second column is the name of the problem on which the algorithm is tested, with the setting of the arbitrarily chosen \((s_i, t_i)\) pair being stated in the third column. The initial value of error terms (19) for \(i = 1, 2\), which we found to be the most important factor affecting the total run time, is recorded in the fourth column. As observed from all four sample problems, if the initial value of error terms (19) is relatively small, the algorithm tends to quickly terminate with a global optimal solution. The fifth column is the total number of iterations it takes to terminate, limited by 500. Some problems are not solved to optimum in 500 iterations due to the bad choices of the \((s, t)\)-pairs (STExp2, STExp4, STExp6, STExp8, STExp9, and STExp13). In the sixth column, the number of subproblems identified infeasible during the process is recorded, with the ratio of infeasible/all subproblems appearing in the adjacent bracket. This value
Table 4: Global solution and algorithm performance on LPCC in [120].
is reasonably relevant to the convergence of the algorithm, since as more areas are
eliminated during the partitioning process, fewer iterations are needed to obtain
the very small areas. The seventh, eighth, and ninth columns indicate whether the
problem is solved to optimum or not, the total runtime, and the objective value at
termination before purifying respectively. For those runs that fail to obtain global
optimum, the objective value at termination is merely a lower bound, which we
found can be very loose (STExp2).

Interestingly enough, there are runs that have the same initial errors to start
with, end up with different runtime (STExp1 v.s STExp3, STExp5 v.s STExp7,
STExp10 v.s STExp12, and STExp14 v.s STExp16). Especially for STExp1 and
STExp3, we noticed that the number of iterations are almost the same, yet the
\((s,t)\) pair used in STExp3 seems to cause some difficulty for the solver to solve
each subproblem. Choosing \((s,t)\) at negative values (STExp3, STExp7, STExp11,
and STExp15) caused different results from the runs with positive values (STExp1,
STExp5, STExp10, and STExp14 respectively). For problem R100_1 and R300_3,
\((100, 0.01)\) is the best \((s,t)\) pairs among others, and for problem YM_8 and YM_36,
\((10000, 1000)\) is the best \((s,t)\) pairs among others. Basically, we can’t conclude a
trend of choosing good \((s,t)\) pairs from the large or the small values, but it seems
that data generated from same sources would have same preference at the choice of
\((s,t)\).

4.6 Effect on approximation error window

We are interested in how the initial gap (between valid lower and upper bound), the
range of \([l_i, u_i]\), and the range of \([l_{sx-tz_i}, u_{sx-tz_i}]\) may affect the algorithm
performance. In this set of experiments, \(window1_i, window2_i, \forall i = 1, 2, \) and \(window3, \)
which are defined as the following:

\[
\begin{align*}
    u_i - l_i &= \text{window1}_i, \forall i = 1, 2, \\
    u_{(sx-tz)_i} - l_{(sx-tz)_i} &= \text{window2}_i, \forall i = 1, 2, \\
    \text{validUB} - \text{validLB} &= \text{window3},
\end{align*}
\]

are fixed at some value. To reach this purpose, the values of \(u_i, l_i, u_{(sx-tz)_i}, l_{(sx-tz)_i}, \)
\(\text{validLB}, \) and \(\text{validUB}\) are designed in the experiments to initiate the algorithm.
### Table 5: Algorithm performance on different choice of $(s, t)$.

Assuming the global solutions are known a priori, where the objective value is $\theta^{pre}$ and the solutions are $x^{pre}$, $y^{pre}$, and $z^{pre}$, (which actually can be obtained from the runs in Section 4.1 and 4.4). We then set the initial value of $validUB$ as $\theta^{pre}$, the initial $validLB$ as $\theta^{pre} - window3$, the values of $l_i$ and $u_i$, with which the partitioning process starts, as $(sx_i^{pre} + tz_i^{pre}) \pm window1_i/2$. For linearized terms that only exist

<table>
<thead>
<tr>
<th>Exp. name</th>
<th>Prob name</th>
<th>$(s,t)$</th>
<th>initial error_1, error_2</th>
<th>#partit</th>
<th>#(%)(%)</th>
<th>infeasible LP or QCP</th>
<th>solved in 500 iteration?</th>
<th>total time</th>
<th>validLB before purification</th>
</tr>
</thead>
<tbody>
<tr>
<td>STExp1</td>
<td>R100_1</td>
<td>(100, 0.01)</td>
<td>817707, 944708</td>
<td>127</td>
<td>190</td>
<td>(74.80%)</td>
<td>Yes</td>
<td>72.03</td>
<td>2442.99</td>
</tr>
<tr>
<td>STExp2</td>
<td>R100_1</td>
<td>(1, 1)</td>
<td>158861211, 152084522</td>
<td>500</td>
<td>166</td>
<td>(16.60%)</td>
<td>No</td>
<td>727.8</td>
<td>-574.95</td>
</tr>
<tr>
<td>STExp3</td>
<td>R100_1</td>
<td>(-100, -0.01)</td>
<td>817707, 944708</td>
<td>128</td>
<td>195</td>
<td>(76.17%)</td>
<td>Yes</td>
<td>569.53</td>
<td>2442.99</td>
</tr>
<tr>
<td>STExp4</td>
<td>R100_1</td>
<td>(10000, 100)</td>
<td>15984689, 15311312</td>
<td>500</td>
<td>143</td>
<td>(14.30%)</td>
<td>No</td>
<td>894.22</td>
<td>1449.31</td>
</tr>
<tr>
<td>STExp5</td>
<td>R100_1</td>
<td>(100, 0.01)</td>
<td>99999999, 99999999</td>
<td>158</td>
<td>215</td>
<td>(68.04%)</td>
<td>Yes</td>
<td>3678.62</td>
<td>-1616.42</td>
</tr>
<tr>
<td>STExp6</td>
<td>R100_1</td>
<td>(1, 1)</td>
<td>99999999, 99999999</td>
<td>500</td>
<td>196</td>
<td>(19.60%)</td>
<td>No</td>
<td>7602.56</td>
<td>-7652.71</td>
</tr>
<tr>
<td>STExp7</td>
<td>R100_1</td>
<td>(-100, -0.01)</td>
<td>99999999, 99999999</td>
<td>156</td>
<td>220</td>
<td>(70.51%)</td>
<td>Yes</td>
<td>3858.28</td>
<td>-1616.43</td>
</tr>
<tr>
<td>STExp8</td>
<td>R100_1</td>
<td>(10000, 100)</td>
<td>99999999, 99999999</td>
<td>500</td>
<td>77</td>
<td>(7.70%)</td>
<td>No</td>
<td>7595.23</td>
<td>-4443.35</td>
</tr>
<tr>
<td>STExp9</td>
<td>YM_8</td>
<td>(100, 0.01)</td>
<td>243507, 276819</td>
<td>500</td>
<td>10</td>
<td>(1.00%)</td>
<td>No</td>
<td>627.14</td>
<td>737.85</td>
</tr>
<tr>
<td>STExp10</td>
<td>YM_8</td>
<td>(10000, 100)</td>
<td>674, 613</td>
<td>39</td>
<td>48</td>
<td>(61.54%)</td>
<td>Yes</td>
<td>16.62</td>
<td>770.98</td>
</tr>
<tr>
<td>STExp11</td>
<td>YM_36</td>
<td>(-1000, -100)</td>
<td>617, 561</td>
<td>35</td>
<td>46</td>
<td>(65.71%)</td>
<td>Yes</td>
<td>23.7</td>
<td>771</td>
</tr>
<tr>
<td>STExp12</td>
<td>YM_36</td>
<td>(10000, 100)</td>
<td>674, 613</td>
<td>38</td>
<td>53</td>
<td>(69.74%)</td>
<td>Yes</td>
<td>15</td>
<td>771</td>
</tr>
<tr>
<td>STExp13</td>
<td>YM_36</td>
<td>(100, 0.01)</td>
<td>70135, 109345</td>
<td>500</td>
<td>33</td>
<td>(3.30%)</td>
<td>No</td>
<td>1362.88</td>
<td>909.85</td>
</tr>
<tr>
<td>STExp14</td>
<td>YM_36</td>
<td>(10000, 100)</td>
<td>263, 282</td>
<td>460</td>
<td>125</td>
<td>(13.59%)</td>
<td>Yes</td>
<td>1399.1</td>
<td>921.27</td>
</tr>
<tr>
<td>STExp15</td>
<td>YM_36</td>
<td>(-1000, -100)</td>
<td>170, 230</td>
<td>471</td>
<td>104</td>
<td>(11.04%)</td>
<td>Yes</td>
<td>856.8</td>
<td>921.27</td>
</tr>
<tr>
<td>STExp16</td>
<td>YM_36</td>
<td>(10000, 100)</td>
<td>263, 282</td>
<td>447</td>
<td>125</td>
<td>(13.98%)</td>
<td>Yes</td>
<td>1300.76</td>
<td>921.27</td>
</tr>
</tbody>
</table>
in the subproblems of LP, the initial valid intervals \([l_{(sx-tz)}_{i}, u_{(sx-tz)}_{i}]\) for \(sx_i - tz_i\) are set to be \([sx_i^{pre} - tz_i^{pre} - window2_{i}/2, sx_i^{pre} - tz_i^{pre} + window2_{i}/2]\), for each \(i\). This set of the experiments contains four parts. For the first part, we consider the situation where \(window1_i = window2_i = 0\), and the following constraint is added in the subproblem of LP (27):

\[
L_0 \geq q^T y^{pre} + (y^{pre})^T M y^{pre} + 2(y - y^{pre})^T \left(\frac{M + M^T}{2}\right) y^{pre}. \tag{39}
\]

The results from this part can be interpreted as the runtime needed for the algorithm to verify whether a solution is global optimum or not, when a solution is given. The second part focuses on analyzing the effect of the initial gap, by maintaining a fixed value of \(window1_i\) and \(window2_i\), while \(window3\) varies. In the third part, \(window1_i\) varies while \(window2_i\) and \(window3\) are fixed. In the fourth part, \(window2_i\) varies while \(window1_i\) and \(window3\) are fixed.

The results here suggest that if there exists some preprocessing method that enables us to obtain tighter initial intervals of \([l_i, u_i]\), and \([l_{(sx-tz)}_{i}, u_{(sx-tz)}_{i}]\), potentially, the problem can be solved as fast as in Table 6. From the results in Part 2, we see \(window3\) is not as important as \(window2\) and \(window1\). Although \(window3\) is very small (WExp2_6-WExp2_9, WExp2_15-WExp2_18, and WExp2_24-WExp2_27), total runtime is not reduced while \(window2\) and \(window3\) are not tight. Comparing Part 3 and Part 4, we see that bounding the \(window1\) at some small value can outperform the results obtained from bounding the \(window2\) at the same value, e.g., fixing \(window1\) at 0 (WExp3_21) had a better result than fixing \(window2\) at 0 (WExp4_21). Basically, we can conclude that smaller values of \(window2\) and \(window1\) are always preferred, yet a smaller value of \(window3\) seems to affect the convergence efficiency only to some extent.

5 Conclusion

1. We proposed a domain-partitioning algorithm to solve the bi-parametric LPCC problem with a positive semidefinite \(M\). In this method, the complementarity constraints are aggregated to form a non-convex quadratic inequality. The non-convex part, bi-linear terms, of the quadratic inequality is rewritten as
## Table 6: Algorithm performance on different approximation error windows.

<table>
<thead>
<tr>
<th>Exp index</th>
<th>Prob name</th>
<th>size</th>
<th>win1</th>
<th>win2</th>
<th>win3</th>
<th>ksplit</th>
<th>total time (sec)</th>
<th>obj value at first iteration</th>
<th>validLB before purification</th>
</tr>
</thead>
</table>

### Part 1:

<table>
<thead>
<tr>
<th>Exp index</th>
<th>Prob name</th>
<th>size</th>
<th>win1</th>
<th>win2</th>
<th>win3</th>
<th>ksplit</th>
<th>total time (sec)</th>
<th>obj value at first iteration</th>
<th>validLB before purification</th>
</tr>
</thead>
</table>

### Part 2:

<table>
<thead>
<tr>
<th>Exp index</th>
<th>Prob name</th>
<th>size</th>
<th>win1</th>
<th>win2</th>
<th>win3</th>
<th>ksplit</th>
<th>total time (sec)</th>
<th>obj value at first iteration</th>
<th>validLB before purification</th>
</tr>
</thead>
</table>

### Part 3:

<table>
<thead>
<tr>
<th>Exp index</th>
<th>Prob name</th>
<th>size</th>
<th>win1</th>
<th>win2</th>
<th>win3</th>
<th>ksplit</th>
<th>total time (sec)</th>
<th>obj value at first iteration</th>
<th>validLB before purification</th>
</tr>
</thead>
</table>

### Part 4:

<table>
<thead>
<tr>
<th>Exp index</th>
<th>Prob name</th>
<th>size</th>
<th>win1</th>
<th>win2</th>
<th>win3</th>
<th>ksplit</th>
<th>total time (sec)</th>
<th>obj value at first iteration</th>
<th>validLB before purification</th>
</tr>
</thead>
</table>
difference of two quadratic functions: $\frac{1}{4}(x_i + z_i)^2 - \frac{1}{4}(x_i - z_i)^2$. The first quadratic term is approximated accordingly to form a relaxed convex QCP subproblem which is then solved by CPLEX within a depth-first branch-and-bound scheme. To avoid solution of re-occurrence, we proposed a partitioning strategy that can sometimes cut off the previous solution point. Meanwhile, the sophisticated partitioning strategy can avoid narrow-and-wide feasible domain which may lead to a numerical difficulty. Using a simple linearization technique, a subproblem of LP can used to substitute the subproblem of QCP, and thus it can be robustly solved by CPLEX.

2. A pair of parameters $(s_i, t_i)$ for all $i = 1, 2$ is introduced to change the plane where the partitioning is done. This trick is found to be effective to help the algorithm converge more efficiently. The key is to identify some $(s_i, t_i)$ such that (19) is small. There is no way to theoretically compute the best choice of $(s_i, t_i)$, and we made decision just by experimenting on several different combinations and choosing the best among them. Besides, there are some optional features including the McCormick bounds, multiple s-t-cuts, valid lower bound refinement, and $(u_i - l_i)$ interval refinement, that can be turned on to possibly help to solve the problem. These features consume extra time, but they don’t always help reduce the total running time, so they are not contained in a standard run of the algorithm.

3. We tested the algorithm on a total of 90 bi-parametric LPCC instances, including 10 random generated instances, 10 instances with skew-symmetric $M$ matrix, 10 instances with zero $B$ matrix, and 60 instances with $M$ matrix that has different rank and density. Those instances are of 100, 150, 200, or 300 complementarity constraints. Some instances are solved by KNITRO as a comparison. KNITRO can find local solutions very quickly, and some of them actually meet the global optimum. However, there are also some local solutions are far from the global optimum. We see that finding a global optimum is indeed necessary for some instances. Though spending more time than KNITRO requires, our algorithm finds global optimum for all 90 instances robustly. At the last part of numerical experiments, we conducted the controlled experiments on how sensitive the algorithm is regarding the initial valid bounds
on \((sx + tz)_i\), \((sx - tz)_i\) and on the objective function separately. This result provides a potential improved running speed of the algorithm if some clever methods can be developed in the future to tighten the bounds.
Chapter 3: Global Resolution of the Cross-validated Support Vector Machine Regression Parameters Selection Problem

Keywords: support vector machine regression, optimal parameter, model selection, global optimality, bi-parametric linear program with linear complementarity constraints

1 Introduction

The support vector machine (SVM) method was originally developed as a tool of data classification by Vapnik in 1964 and its use has been extended to regression since 1997 [117]. This method has drawn much attention in the past 20 years because of the good performance it provides on practical applications in data mining and machine learning. It is known that the SVM regression, or SVR, has two design parameters that significantly affect its performance: 1) the size of the insensitivity zone, and 2) the regularization parameter that is assigned to provide a trade-off
between the absolute residual and the separation of the data. Selecting the best parameters through cross-validation between the training data and testing data can be formulated as a bi-level optimization model.

In the SVM regression bi-level parameter selection problem, the inner level is an SVM regression model that determines a mapping featuring the normal vector \((w_s)\) and the bias \((b_s)\), given the size of the insensitivity zone \((\varepsilon_e)\) and the regularization parameter \((C_e)\). The SVM regression model is a strictly convex quadratic problem, and its KKT condition is necessary and sufficient for optimality. The inner problem thus can be equivalently reformulated by the KKT condition as a linear complementarity problem (LCP). Making use of this reformulation, the semismooth method [43], successive overrelaxation method [87], and the interior-point method [42] are algorithms that have been studied for solving SVM problems with large numbers of data points, and have yielded some good results. (Problems with up to 60 million data points and 34 features are solved in [43, 42]. Problems with up to 10 million data points and 32 features are solved in [87].) On the other hand, the outer level problem is a linear regression model subject to the inner optimization problem and box constraints on its design parameters. To solve this SVR parameter selection problem, algorithms have been proposed for solving a general nonlinear bi-level program to global optimum, including the \(\alpha\)BB-type method in [55], the branch-and-bound-type method in [12], and other methods being reviewed in [47].

This work is among a recent series of research about multi-folds cross-selecting and validating the parameters of the SVR, which reformulates the bi-level model as a model of bi-parametric linear complementarity constrained program. Previous works in this series include the studies in [73, 72, 59, 120, 74]. In [73, 72], a cross-validated SVR parameter selection model containing a “feature selection” scheme is considered. The benchmark for parameter selection in their work is the performance of the selected SVR model on a “hold-out” set of data points rather than the quality of the solution to the bi-level program. In [120], a two-stage branch-and-cut algorithm is proposed for solving the linear program with complementarity constraints (LPCC) to global optimum. In the experiments presented in [120], if the lower bound of the objective value can be pushed closer to the upper bound in the pre-processing stage, the global solution can be identified efficiently in the second stage.
The global-optimization algorithms proposed in [59, 74] can be accommodated to
some more general types of the LPCC. The need for new algorithms to solve for
the global optimum of the SVM regression cross-validation problem comes from the
fact that the amount of training data is usually large, and the run time of solving
a large size bi-level problem, which is NP-hard, explodes when the problem size
grows. No one has yet determined whether this solution process can be expedited.

In this chapter, we develop algorithms that solve this parameter selection
model by exploring the structure of cross-validated SVM regression. The main
algorithm we propose is to search the optimal design parameters in the feasible
region \((C_\varepsilon, \varepsilon_\varepsilon)\). To do this, the concept of “invariancy region” for the inner level
problem is crucial. An invariancy region is a region in the parameter space where
the basis remains unchanged. Invariancy regions are convex, and they partition the
whole feasible region without overlapping. We provide a searching and partitioning
method on \((C_\varepsilon, \varepsilon_\varepsilon)\)-plane that initiates the search from a fixed point \((C_\varepsilon, \varepsilon_\varepsilon)\), with
which a lower-level problem with fixed parameters is solved, and proceeds by iden-
tifying the invariancy interval along one chosen direction. A queue of rectangular
\((C_\varepsilon, \varepsilon_\varepsilon)\)-areas, which result from the partitioning, is maintained and searched one
by one. For each rectangular area, we either conclude that all the invariancy re-
gions inside the area have been examined and remove the rectangle from the queue,
or we partition the rectangular areas horizontally and/or vertically, add the new
rectangular subareas to the queue while removing the original one. The algorithm
terminates when all the rectangular areas in the queue are eliminated. The solution
obtained from this algorithm can be verified to be global optimal. Different from
other methods ([52, 30, 113]) which also perform the search on the parameter plane,
the boundaries of invariancy regions are not explicitly identified in our work. Thus,
it is possible to revisit a previously examined invariancy region. Since the algorithm
involves exploring the invariancy regions, we expect the solution time directly re-
lated to the number of feasible invariancy regions. This conjecture is confirmed in
the numerical experiments.

We propose a second algorithm to solve the cross-validated SVM regression
as an improved integer program (IP). The complementarity constrained program
can be formulated as this IP via the big-M technique. For the specific application
in the SVM regression, the valid big-M is derived from finding the upper bound on the multipliers of the lower-level problem of SVM. Moreover, beyond this theoretical bound, we propose a procedure to further tighten the upper bound on the multipliers. The tightened multiplier upper bounds can reduce the feasible regions for the IP and has enabled us to improve the running time for solving the IP by \textsc{Cplex} \cite{1}. By monitoring the process of branch-and-bound, we notice that the gap between the lower and upper bounds reduces very slowly because the lower bound of the objective is hardly improved, whereas the upper bound of the objective is usually tight.

The contribution of this work is in obtaining a certificate of global optimality for the cross-validated SVM regression parameter selection problem. This global optimal parameter and its corresponding regression residual can be used as a benchmark for every other choice of parameters. Two approaches are proposed to solve the problem to global optimum, including a \((C_e, \varepsilon_e)\)-rectangle search algorithm that aim to take advantage of existing efficient methods of solving the inner level problem and an improved integer programming model that relies on an existing IP solver. The algorithms are tested on instances with single- and multi-fold training and testing data sets, including those generated by us and those from the real world. A significant number of numerical experiments are presented to uncover the strength and limit of each algorithm. We will show that the level of difficulty of the instances inputting to the \((C_e, \varepsilon_e)\)-rectangle search algorithm can be classified by a four-quadrant diagram. Comparing the convergent time of the instances between different quadrants is meaningless because they belong to different scales of difficulty. The running time of the instances in the same quadrant is proportional to the size of data. We also compare the global solution to the solution produced by the non-global optimization commercial solver. Although substantially more time is needed by our algorithm, the solution quality is better in most cases.

The remaining part of this paper is organized as follows. Section 2 introduces the SVM classification and regression models and derives the SVM regression parameter selection problem as a linear program with complementarity constraints. Section 3 introduces the \((C_e, \varepsilon_e)\)-rectangle search algorithm in detail, including our definition of the invariancy region, the geometrical allocation of data points, the
procedure to find the invariancy intervals on a chosen line, and details of the two stages of the algorithm. Section 4 describes the big-numbers tightening algorithm that we used to form the modified IP. Section 5 displays the numerical results of the \((C, \varepsilon)\)-rectangle search algorithm, the modified IP, and the local solution of \textsc{KNITRO} \cite{23}. The performance of the algorithms and the difficulty of the instances are depicted and analyzed. Section 6 summarizes the chapter and concludes the findings from this research.

2 Models and Properties

In this section we introduce the models of the typical SVM and a parameter selection model for the SVM regression. The later model is a bi-level program where the former model occurs as the inner-level problem.

In Section 2.1, we describe the difference between the hard margin SVM classification, the soft margin SVM classification, and the SVM regression problem. The hard margin SVM classification has no parameter, the soft margin SVM classification involves one parameter, \(C\), and the SVM regression problem involves two parameters \((C, \varepsilon)\). In Section 2.2, we explain the multi-fold cross-validation technique in selecting the parameters for the SVM regression and clarify the meaning of the “optimal parameters” in our framework. Following this motivation, the bi-level program formulation and the linear program with complementarity constraints reformulation are presented.

2.1 Support Vector Machine Regression

The SVM model for classification is the original version of every SVM-related study. The SVM classifier is a hyperplane \(\mathbf{w}_a^T \mathbf{x}_d^j + b_a = y_{d_j}\) that classifies the data points \((\mathbf{x}_d^j, y_{d_j})\ \forall j \in D\), where \(\mathbf{x}_d^j \in \mathbb{R}^K\), \(y_{d_j} \in \{+1, -1\}\), and \(D\) denotes the set of training data. The dimension of \(\mathbf{x}_d^j\) represents the number of features or characteristics that describe the data point \(j \in D\), and the value of the variable \(y_{d_j}\) indicates the group to which the data point \(j\) belongs. A data point \(j\) is said to be misclassified by the hyperplane \(\mathbf{w}_a^T \mathbf{x}_d^j + b_a = y_{d_j}\) if the predicted \(y_{d_j}\) is +1 yet the true \(y_{d_j}\) is −1, or vice versa. In the basic setting of SVM classification, the misclassification of a training data point is neglected if the misclassified data point is within a tube area around
the target hyperplane $w_s^T x_d^j + b_s = 0$. This tube is defined by the two parallel hyperplanes $w_s^T x_d^j + b_s = +1$ and $w_s^T x_d^j + b_s = -1$.

In a “soft margin” SVM classification, the desired hyperplane $(w_s, b_s)$ is the optimizer of the following program:

$$\min_{w_s, b_s, \xi} C \sum_{j \in D} \xi_j + \frac{1}{2} w_s^T w_s$$

subject to

$$w_s^T x_d^j + b_s \geq 1 - \xi_j, \forall y_{d_j} = +1,$$

$$w_s^T x_d^j + b_s \leq -1 + \xi_j, \forall y_{d_j} = -1,$$

and

$$\xi_j \geq 0.$$  \hspace{1cm} (1)

where $C$ is a parameter trading off the two terms in the objective function, and $\xi$ is the slack. The hyperplane $(w_s, b_s)$ minimizes the sum of the distance between the misclassified observation and the closer bound of the tube, and also maximizes the margin size, $2/\|w_s\|$, between $w_s^T x_d^j + b_s = +1$ and $w_s^T x_d^j + b_s = -1$. In contrast to the soft margin SVM, the “hard margin” SVM does not allow any misclassification beyond the tube, while the misclassification inside the tube is still neglected. To formulate the hard margin SVM, the model in (1) is revised by removing the $C \sum_{j \in D} \xi_j$ term from the objective function and removing the $-\xi_j$ and $+\xi_j$ terms from the two constraints respectively. In [69], the hard margin version is referred to as “linear maximal margin classifier for linearly separable data”, and the soft margin version is referred to as “linear soft margin classifier for overlapping classes.”

Extended from the classification, the SVM regression identifies a hyperplane $y_{d_j} = w_s^T x_d^j + b_s$ where $y_{d_j}$ is a real-valued dependent variable. Besides the $C$ parameter, the other parameter, $\varepsilon$, defines the size of the tube in which the residual is neglected. Given $C$ and $\varepsilon$, the desired hyperplane minimizes the following problem:

$$\min_{w_s, b_s} \left\{ C \sum_{j=1}^n \max \left( |w_s^T x_d^j + b_s - y_{d_j}| - \varepsilon, 0 \right) + \frac{1}{2} w_s^T w_s \right\}. \hspace{1cm} (2)$$

The $\frac{1}{2} w_s^T w_s$ term in the objective of (2) is directly borrowed from its usage in (1). This term is also called a regularization term because it imposes strong convexity
and forces the optimal $\mathbf{w}_s$ to be unique. The other term in the objective is the (least) absolute residual outside the $\varepsilon_c$-insensitive tube.

### 2.2 Cross-validated SVM Regression LPCC formulation

Suppose we have $F$ folds of training and testing data, that is to say, we divide the training observations $(\mathbf{x}_d^j, y_d^j), j = 1, \ldots, n$ into $F$ groups and testing observations $(\mathbf{x}_d^j, y_d^j), j = n + 1, \ldots, n + m$ into another $F$ groups. Given any $(C_e, \varepsilon_e)$ pair, a hyperplane $(\mathbf{w}_f^j, b_f^j)$ minimizing the objective function of SVR can be obtained by solving (2) for each fold $f = 1, \ldots, F$ of the training data. Using the trained hyperplanes, the regression residuals of the testing data in each fold are then calculated. A small regression error from the predicted testing data is desired. This hope may be achieved by choosing a different $(C_e, \varepsilon_e)$ pair and repeating the aforementioned process. The procedure of sequentially choosing $(C_e, \varepsilon_e)$, solving for the optimal hyperplane on the training data, and computing the regression errors on testing data, is in fact the cross-validation technique. There is no guarantee that the $C_e$ or $\varepsilon_e$ with a large value always produces a smaller regression error than that result from smaller values of parameters.

Is there really a best choice of the parameters? A set of good parameters defines a model that forecasts the future with small errors. Since the future is not known, it is impossible to find the best parameters that will yield precise predictions. This study is in fact about the best parameters under the framework of cross-validation with fixed features and fixed divisions of the testing and training data sets. This best parameter is well defined as the global optimal solution to a linear program with complementarity constraints (LPCC), which we will introduce later. Although the computational cost to obtain the global optimum of an LPCC is expensive, the result provides a benchmark to evaluate the parameters produced by any other parameter selection algorithms. Note that the best parameters defined under the cross-validation process might change when the division of training and testing data change. In this work, only one fixed division of the data sets is considered for one instance.

The observations are arbitrarily labeled. We denote the first and the last index of the observations in the $f$-th training data set as $\texttt{front}_f$ and $\texttt{end}_f$ respectively,
and there are \( n_f \) observations in the \( f \)-th training data set. Similarly, the first and the last index of the observations in the \( f \)-th testing data is denoted as \( \text{front}_{ts}^f \) and \( \text{end}_{ts}^f \) respectively, and there are \( m_f \) observations in the \( f \)-th testing data set.

The cross-validation process of the SVR parameter selection is formulated as follows:

\[
\min_{C, \varepsilon, w_1^f, b_1^f, \ldots, w_F^f, b_F^f} \sum_{f=1}^{F} \sum_{i=\text{front}_{ts}^f}^{\text{end}_{ts}^f} |w_s^T x_i^d + b_s^f - y_d_i| \\
\text{subject to } 0 \leq C \leq C_e, \quad 0 \leq \varepsilon \leq \varepsilon_e, \quad 0 \leq \varepsilon \leq \varepsilon_e,
\]

and for all \( f = 1, \ldots, F \), \((w_s^f, b_s^f) \in \text{the solution set of the following problem}

\[
\min_{w_s^f, b_s^f} \left\{ C_e \sum_{j=\text{front}_{ts}^f}^{\text{end}_{ts}^f} \max \left( |w_s^T x_d^j + b_s^f - y_d^j| - \varepsilon, 0 \right) + \frac{1}{2} w_s^T w_s \right\},
\]

(3)

If the multiple folds of the training data all share one single fold of testing data, the objective function in (3) needs to be replaced by:

\[
\min_{C, \varepsilon, w_1^f, b_1^f, \ldots, w_F^f, b_F^f} \sum_{f=1}^{F} \sum_{i=\text{front}_{ts}^f}^{\text{end}_{ts}^f} |w_s^T x_i^d + b_s^f - y_d_i|.
\]

(4)

For example, suppose we have 3 folds of testing data which contain \( m_1, m_2 \) and \( m_3 \) observations and 3 folds of training data which contain \( n_1, n_2 \) and \( n_3 \) observations. \( n_1 + n_2 + n_3 = n \) and \( m_1 + m_2 + m_3 = m \). The model (3) is explicitly
written as:

$$\min_{C_e, \varepsilon, w^1_s, b^1_s, \ldots, w^3_s, b^3_s} \left\{ \sum_{i=1}^{n+m} |w^1_s x^i_d + b^1_s - y_d| + \sum_{i=n+1}^{n+m} |w^2_s x^i_d + b^2_s - y_d| + \sum_{i=n+m+1}^{n+m+m+3} |w^3_s x^i_d + b^3_s - y_d| \right\}$$

subject to $0 \leq C_e \leq C, \quad 0 \leq \varepsilon \leq \varepsilon_c \leq \bar{C}$.

$(w^1_s, b^1_s) \in$ the solution set

$$\arg\min_{w^1_s, b^1_s} \left\{ C_e \sum_{j=1}^{n_1} \max \left( |w^1_s x^j_d + b^1_s - y_d| - \varepsilon_c, 0 \right) + \frac{1}{2} w^1_s^T w^1_s \right\},$$

$(w^2_s, b^2_s) \in$ the solution set

$$\arg\min_{w^2_s, b^2_s} \left\{ C_e \sum_{j=n_1+1}^{n_1+n_2} \max \left( |w^2_s x^j_d + b^2_s - y_d| - \varepsilon_c, 0 \right) + \frac{1}{2} w^2_s^T w^2_s \right\},$$

and

$(w^3_s, b^3_s) \in$ the solution set

$$\arg\min_{w^3_s, b^3_s} \left\{ C_e \sum_{j=n_1+n_2+1}^{n_1+n_2+n_3} \max \left( |w^3_s x^j_d + b^3_s - y_d| - \varepsilon_c, 0 \right) + \frac{1}{2} w^3_s^T w^3_s \right\}.$$

(5)

If there is only 1 fold of testing data, the objective function in (5) is replaced by:

$$\min_{C_e, \varepsilon, w^1_s, b^1_s, \ldots, w^3_s, b^3_s} \left\{ \sum_{i=1}^{n+m} |w^1_s x^i_d + b^1_s - y_d| + \sum_{i=n+1}^{n+m} |w^2_s x^i_d + b^2_s - y_d| + \sum_{i=n+m+1}^{n+m+m+3} |w^3_s x^i_d + b^3_s - y_d| + \sum_{i=n+1}^{n+m} |w^4_s x^i_d + b^4_s - y_d| + \sum_{i=n+1}^{n+m+m+3} |w^5_s x^i_d + b^5_s - y_d| \right\}$$

(6)

It is known that the KKT condition of the inner-level problems in (3) is sufficient for optimality because the inner-level problems are convex. $F$ folds of inner-level problems thus can be replaced by the KKT condition, which is equivalent
to $F$ folds of linear complementarity problem $\text{LCP}^{f}_{SVR}$ as follows:

$$\forall f = 1, \ldots, F :$$

$$\begin{align*}
\text{LCP}^{f}_{SVR} := \\
\forall j = \text{front}^f \ldots \text{end}^f : \\
\begin{cases}
0 \leq e_{s_j} + \epsilon_c - (x_d^f)^T w_s^f - b_s^f + y_{d_j} \perp \alpha_j \geq 0, \\
0 \leq e_{s_j} + \epsilon_c + (x_d^f)^T w_s^f + b_s^f - y_{d_j} \perp \beta_j \geq 0, \\
0 \leq C_e - \alpha_j - \beta_j \perp e_{s_j} \geq 0.
\end{cases}
\end{align*}$$
For example, suppose \( F = 3 \), then \( n = n_1 + n_2 + n_3 \), and (7) is written as:

\[
\begin{align*}
\mathbf{w}_1^s - \sum_{j=1}^{n_1} (\beta_j - \alpha_j) \mathbf{x}_{d}^j &= 0, \\
\sum_{j=1}^{n_1} (\alpha_j - \beta_j) &= 0, \\
0 \leq e_s + \varepsilon - (\mathbf{x}_{d}^1)^T \mathbf{w}_1^s - b_1^s + y_{d_j} \perp \alpha_j &\geq 0, \forall j = 1, \ldots, n_1, \\
0 \leq e_s + \varepsilon + (\mathbf{x}_{d}^1)^T \mathbf{w}_1^s + b_1^s - y_{d_j} \perp \beta_j &\geq 0, \forall j = 1, \ldots, n_1, \\
0 \leq C_e - \alpha_j - \beta_j \perp e_{s_j} &\geq 0, \forall j = 1, \ldots, n_1,
\end{align*}
\]

(8)

Furthermore, by introducing the variable \( p_i, \forall i = n + 1, \ldots, n + m \), to linearize the upper level problem in (3), we have the following linear program with
complementarity constraints (LPCC):

\[
\min_{C_e, \varepsilon_e, w^1_s, b^1_s, p_i, \alpha_j, \beta_j, \varepsilon_j} \sum_{f=1}^{F} \sum_{i=\text{front}_{ts}^f}^{\text{end}_{ts}^f} p_i
\]

subject to
\[
0 \leq C \leq C_e \leq \bar{C},
\]
\[
0 \leq \varepsilon \leq \varepsilon_e \leq \bar{\varepsilon},
\]
\[
(x_d^i)^T w^1_s + b^1_s - y_{d_i} \leq p_i, \quad \forall i = n+1, \ldots, n+m_1,
\]
\[
-(x_d^i)^T w^1_s - b^1_s + y_{d_i} \leq p_i, \quad \forall i = n+1, \ldots, n+m_1,
\]
\[
(x_d^i)^T w^2_s + b^2_s - y_{d_i} \leq p_i, \quad \forall i = n+m_1+1, \ldots, n+m_1+m_2,
\]
\[
-(x_d^i)^T w^2_s - b^2_s + y_{d_i} \leq p_i, \quad \forall i = n+m_1+1, \ldots, n+m_1+m_2,
\]
\[
\vdots
\]
\[
(x_d^i)^T w^F_s + b^F_s - y_{d_i} \leq p_i, \quad \forall i = \text{front}_{ts}^F, \ldots, \text{end}_{ts}^F,
\]
\[
-(x_d^i)^T w^F_s - b^F_s + y_{d_i} \leq p_i, \quad \forall i = \text{front}_{ts}^F, \ldots, \text{end}_{ts}^F,
\]

and constraints in (7).

For the case where there is only 1 fold of testing data containing \(m\) data points, the LPCC that results from linearizing is different from that in (9) and is written.
as follows:

\[
\min_{C, \varepsilon} \sum_{f=1}^{F} \sum_{i=n+1}^{n+m} p_i \\
\text{subject to } 0 \leq C \leq C_e \leq \overline{C}, \\
0 \leq \varepsilon \leq \overline{\varepsilon}, \\
(x^i_d)^T w^k_s + b^k_s - y_d \leq p_i, \quad \forall i = n+1, \ldots, n+m, \\
-(x^i_d)^T w^k_s - b^k_s + y_d \leq p_i, \quad \forall i = n+1, \ldots, n+m, \\
(x^i_d)^T w^2_s + b^2_s - y_d \leq p_i, \quad \forall i = n+1, \ldots, n+m, \\
-(x^i_d)^T w^2_s - b^2_s + y_d \leq p_i, \quad \forall i = n+1, \ldots, n+m, \\
\vdots \\
(x^i_d)^T w^F_s + b^F_s - y_d \leq p_i, \quad \forall i = n+1, \ldots, n+m, \\
-(x^i_d)^T w^F_s - b^F_s + y_d \leq p_i, \quad \forall i = n+1, \ldots, n+m, \\
\text{and constraints in (7).}
\]

3 \quad (C_e, \varepsilon_e)\text{-rectangle search algorithm}

We demonstrate a global optimization algorithm that solves programs (9) and (10) in this section. This algorithm contains the following 8 key tasks:

1. Solving the lower-level problem with a fixed \((C_e, \varepsilon_e)\) by known methodologies.

2. Replacing the complementarity constraints with linear constraints, which restricts the feasible region in a smaller but convex region (invariance region).

3. Solving for the best \((C_e, \varepsilon_e)\) and the optimal solution within the region defined by a piece of \(LCP_{SVR}^f\). This can be done by solving a linear program.

4. Searching for the next \((C_e, \varepsilon_e)\) at which the lower-level problem is solved.

5. Partitioning the initial \([C, \overline{C}] \times [\varepsilon, \overline{\varepsilon}]\) area into small rectangular regions at chosen points. When partitioning, a small margin (around \(10^{-4}\)) along the boundary is cropped out in the new rectangles.
6. Maintaining a list of visited invariancy regions by recording their corresponding data allocation in space (grouping).

7. Maintaining a queue of rectangular areas in \((C_e, \varepsilon_e)\)-plane to be examined.

8. Eliminating the rectangular areas from the queue if 1) the four-corner points of the \((C_e, \varepsilon_e)\)-rectangular area have the same grouping vector, or 2) the \((C_e, \varepsilon_e)\)-rectangular area is bisected into two regions by a straight line, each having one grouping vector.

Task 1 is discussed in Section 3.1; the invariancy region mentioned in task 2 is defined in Section 3.2; the piece mentioned in task 3 is discussed in Section 3.2; the linear program mentioned in task 3 is introduced in Section 3.3; the search strategy in task 4 can be seen in the algorithm described in Section 3.5; the grouping mentioned in task 6 is defined in Section 3.2; the partitioning, recording, and eliminating steps mentioned in tasks 5-8 are shown in Sections 3.5 and 3.6.

### 3.1 Lower-level problem with fixed \((C_e, \varepsilon_e)\)

The lower-level problem with fixed \((C_e, \varepsilon_e)\) is an SVM regression problem for each fold of training data. By writing down the KKT conditions, the lower-level problems are equivalently reformulated as linear complementarity problems (LCP). To solve these LCPs, we employ the semismooth method [32], which involves the semismooth Fischer-Burmeister function.

Consider the general complementarity problem as follows:

\[
0 \leq F_i(z) \perp z_i \geq 0, \quad \forall i \in \mathbb{I},
\]

\[
0 = F_i(z) \perp z_i : \text{free}, \quad \forall i \in \mathbb{E},
\]

where \(\mathbb{I}\) and \(\mathbb{E}\) denotes the nonoverlapping sets of indices for inequalities and equalities respectively. The Fischer-Burmeister function for the LCP (11) is defined as:

\[
\phi(z_i, F_i(z)) := z_i + F_i(z) - \sqrt{z_i^2 + F_i(z)^2}.
\]

It holds that \(\phi(z_i, F_i(z)) = 0\) if and only if \(0 \leq F_i(z) \perp z_i \geq 0\). In the semismooth
method, the merit function being used is of the following form:

\[
\Phi(z) := \begin{bmatrix}
\vdots \\
\phi(z_i, F_i(z)), \forall i \in \mathbb{I} \\
\vdots \\
F_i(z), \forall i \in \mathbb{E} \\
\vdots 
\end{bmatrix},
\]

(13)

The merit function \( \Phi(z) \) has been proven, such as in [40], to have some desired properties including that \( \Phi(z) \) is a semismooth function and that \( g(z) := \frac{1}{2} \| \Phi(z) \|_2^2 \) is continuously differentiable. Most importantly, for any \( H \in \partial_B \Phi(z) \), where \( \partial_B \Phi(z) \) represents the B-subdifferential of \( \Phi(z) \), \( \nabla g(z) = H^T \Phi(z) \). These properties hold under the assumption of continuous differentiability of \( F(z) \), which is satisfied in the case of SVR. By solving the equation \( g(z) = 0 \), a solution to the complementarity problem (11) is found. Theoretical foundations of the semismooth method can be seen in [32, 18, 43]. Below we demonstrate the damped Newton method [40] which we employed to solve the lower-level problem.

**Algorithm 3.1 Damped Newton method** (Algorithm 2 in [43])

**Step 1: Initialization:** Let \( z^0 \in \mathbb{R}^n \), \( \rho \geq 0 \), \( p \geq 2 \), and \( \sigma \in (0, \frac{1}{2}) \) be given. Set \( k = 0 \). Set \( tol^1 \).

**Step 2: Termination:** If \( g(z^k) := \frac{1}{2} \| \Phi(z^k) \|_2^2 \leq tol \), stop.

**Step 3: Direction Generation:** Otherwise, let \( H^k \in \partial_B \Phi(z^k) \), and calculate \( d^k \in \mathbb{R}^n \) solving the Newton system:

\[
H^k d^k = -\Phi(z^k).
\]

(14)

If either (14) is unsolvable or the descent condition

\[
\nabla g(z^k)^T d^k < -\rho \| d^k \|_2^p
\]

(15)

\(^1We use \( tol = 10^{-14} \).
is not satisfied, then set
\[ d^k = -\nabla g(z^k). \]  

**Step 4: Line Search:** Choose \( t^k = 2^{-i_k} \), where \( i_k \) is the smallest integer such that
\[ g(z^k + 2^{-i_k}d^k) \leq g(z^k) + \sigma 2^{-i_k} \nabla g(z^k)^T d^k. \]  

**Step 5: Update:** Let \( z^{k+1} := z^k + t^k d^k \) and \( k := k + 1 \). Go to 3. \( \square \)

The \( B \)-subdifferential used in Step 3 of Algorithm 3.1 is obtained using the following theorem.

**Theorem 4.** (Theorem 5 in [43]) Let \( F : \mathbb{R}^n \to \mathbb{R}^n \) be continuously differentiable. Then
\[ \partial_B \Phi(z) \subseteq \{ D_a + D_b F'(z) \}, \]  

where \( D_a \in \mathbb{R}^{n \times n} \) and \( D_b \in \mathbb{R}^{n \times n} \) are diagonal matrices with entries defined as follows:

1. For all \( i \in I \): If \( \| (z_i, F_i(z)) \| \neq 0 \), then
   \[
   (D_a)_{ii} = 1 - \frac{z_i}{\| z_i, F_i(z) \|}, \\
   (D_b)_{ii} = 1 - \frac{F_i(z)}{\| z_i, F_i(z) \|};
   \]  
   otherwise
   \[
   ((D_a)_{ii}, (D_b)_{ii}) \in \{ (1 - \eta, 1 - \gamma) \in \mathbb{R}^2 \| (\eta, \gamma) \| \leq 1 \}. \]  

2. For all \( i \in E \):
   \[
   (D_a)_{ii} = 0, \\
   (D_b)_{ii} = 1.
   \]  

\( \square \)
In Theorem 4, if \( \| (z_i, F_i(z)) \| \neq 0 \), \( \Phi(z) \) is differentiable at \( z \), and (19) computes the exact Jacobian. If \( \| (z_i, F_i(z)) \| = 0 \), \( \eta \) and \( \gamma \) in (20) are computed as in [39]:
\[
\eta = \frac{v_i}{\sqrt{v_i^2 + (F'v)_i^2}}, \quad \text{and} \quad \gamma = \frac{(F'v)_i}{\sqrt{v_i^2 + (F'v)_i^2}},
\]
where \( v \in \mathbb{R}^n \) is a vector of our choice whose \( i^{th} \) element is nonzero, provided \( \| (z_i, F_i(z)) \| = 0 \).

In the context of each fold of SVR, \( \Phi(z') \in \mathbb{R}^{3n_f+1} \) is of the form in (26) provided the fixed parameters \((\overline{C}, e, \overline{e})\), and the computation of the B-subdifferential \( H \) in (18) requires \( \mathbf{F}'(z') \) of the form:
\[
\mathbf{F}'(z') = \begin{bmatrix}
X^T X & -X^T X & \mathbf{I}_{n_f \times n_f} & \mathbf{-1}_{n_f \times 1} \\
-\mathbf{X}^T \mathbf{X} & \mathbf{X}^T \mathbf{X} & \mathbf{I}_{n_f \times n_f} & \mathbf{1}_{n_f \times 1} \\
-\mathbf{1}_{n_f \times n_f} & -\mathbf{1}_{n_f \times n_f} & \mathbf{0}_{n_f \times n_f} & \mathbf{0}_{n_f \times 1} \\
\mathbf{1}_{1 \times n_f} & -\mathbf{1}_{1 \times n_f} & \mathbf{0}_{1 \times n_f} & \mathbf{0}
\end{bmatrix},
\]
where \( X^T X \) is a \( n_f \times n_f \) matrix comprising elements \( x_i^T x_j \) for all \( i = \text{front}^f, \ldots, \text{end}^f \) and \( j = \text{front}^f, \ldots, \text{end}^f \); \( \mathbf{I}_{n_f \times n_f} \) is the identity matrix that belongs in \( \mathbb{R}^{n_f \times n_f} \); \( \mathbf{1}_{n_f \times 1} \) is a matrix belonging in \( \mathbb{R}^{n_f \times 1} \) that has all 1 entries; \( \mathbf{0}_{n_f \times 1} \) is the matrix belonging in \( \mathbb{R}^{n_f \times 1} \) that has all 0 entries; and the variables vector \( z' \in \mathbb{R}^{(3n_f+1)} \) is written as
\[
z' = [\alpha_{\text{front}^f} \cdots \alpha_{\text{end}^f} | \beta_{\text{front}^f} \cdots \beta_{\text{end}^f} | e_{s_{\text{front}^f}} \cdots e_{s_{\text{end}^f}} | b'_s]^T.
\]
To compute $\eta$ and $\gamma$ in (22), we can choose $v = 1$, and let

$$h v := F'(z^f)v = \begin{bmatrix} 0_{n_f \times 1} \\ \vdots \\ 2_{n_f \times 1} \\ \vdots \\ -2_{n_f \times 1} \\ 0 \end{bmatrix},$$

which doesn’t require update. Then the computation of $\eta$ and $\gamma$ is simplified as

$$\eta = \frac{1}{\sqrt{1 + (hv)_i^2}} \quad \text{and} \quad \gamma = \frac{(hv)_i}{\sqrt{1 + (hv)_i^2}}$$

(25)

where the index $i$ is the same as defined in (22).

Now suppose $\{z^k\}$ is a sequence generated in Algorithm 3.1 and $\{z^k\} \to z^*$, where $z^*$ is the final solution to the system $\Phi(z) = 0$. It is known that if $k$ is sufficiently large, the search direction $d^k$ is always chosen at the Newton step computed in (14) rather than the steepest decent step as in (16). Meanwhile, if $k$ is sufficiently large, $t^k$ is always chosen at 1 [32]. The decent condition in (15) ensures the semismooth method converges globally, i.e., any initial point, not necessarily close to the solution, can lead to convergence. However, in our experiments, we ignore steps (15) and (16) to save running time because the system (14) is always solvable, and the convergence is obtained in most cases with the initial $\{z^0\} = 0$.

For rare cases where the condition $g(z^k) \leq tol$ in Step 2 can not be fulfilled, we try a different initial point to restart.
\[
\Phi(z^f) =
\begin{pmatrix}
\alpha_{\text{front}} + (\varepsilon_{\text{front}} + \varepsilon - (x^{\text{front}}_d)^T \sum_{i=\text{front}}^{\text{end}} (\beta_i - \alpha_i)x^i_d - b^i_d + y_d) - \sqrt{(\alpha_{\text{front}})^2 + (\varepsilon_{\text{front}} + \varepsilon - (x^{\text{front}}_d)^T \sum_{i=\text{front}}^{\text{end}} (\beta_i - \alpha_i)x^i_d - b^i_d + y_d)^2} \\
\alpha_{\text{end}} + (\varepsilon_{\text{end}} + \varepsilon - (x^{\text{end}}_d)^T \sum_{i=\text{front}}^{\text{end}} (\beta_i - \alpha_i)x^i_d - b^i_d + y_d) - \sqrt{(\alpha_{\text{end}})^2 + (\varepsilon_{\text{end}} + \varepsilon - (x^{\text{end}}_d)^T \sum_{i=\text{front}}^{\text{end}} (\beta_i - \alpha_i)x^i_d - b^i_d + y_d)^2} \\
\beta_{\text{front}} + (\varepsilon_{\text{front}} + \varepsilon - (x^{\text{front}}_d)^T \sum_{i=\text{front}}^{\text{end}} (\beta_i - \alpha_i)x^i_d - b^i_d + y_d) - \sqrt{(\beta_{\text{front}})^2 + (\varepsilon_{\text{front}} + \varepsilon - (x^{\text{front}}_d)^T \sum_{i=\text{front}}^{\text{end}} (\beta_i - \alpha_i)x^i_d - b^i_d + y_d)^2} \\
\beta_{\text{end}} + (\varepsilon_{\text{end}} + \varepsilon - (x^{\text{end}}_d)^T \sum_{i=\text{front}}^{\text{end}} (\beta_i - \alpha_i)x^i_d - b^i_d + y_d) - \sqrt{(\beta_{\text{end}})^2 + (\varepsilon_{\text{end}} + \varepsilon - (x^{\text{end}}_d)^T \sum_{i=\text{front}}^{\text{end}} (\beta_i - \alpha_i)x^i_d - b^i_d + y_d)^2} \\
\varepsilon_{\text{front}} + (C_e - \alpha_{\text{front}} - \beta_{\text{front}}) - \sqrt{(\varepsilon_{\text{front}})^2 + (C_e - \alpha_{\text{front}} - \beta_{\text{front}})^2} \\
\varepsilon_{\text{end}} + (C_e - \alpha_{\text{end}} - \beta_{\text{end}}) - \sqrt{(\varepsilon_{\text{end}})^2 + (C_e - \alpha_{\text{end}} - \beta_{\text{end}})^2} \\
\sum_{j=\text{front}}^{\text{end}} \alpha_j - \sum_{j=\text{front}}^{\text{end}} \beta_j
\end{pmatrix}
\]
The lower-level problem with fixed \((C_e, \varepsilon_e)\) produces the unique optimal solution \(w_s\) but not the unique \(b_s\). This is because strong convexity is imposed on \(w_s\) but not on \(b_s\) (See [21] for exceptions).

The semismooth method is neither the only nor the guaranteed best way to solve the lower-level problem with a fixed \((C_e, \varepsilon_e)\). The successive overrelaxation method [87] and the interior method [42] applied in the SVM might be substitutions, but the comparison is not within the scope of this work.

### 3.2 Piece of the complementarity and data point allocation (grouping) in space

Consider the \(LCP_{SVR}^f\) in (7). We define the binary variables \(z_j, z'_j\) and \(\eta_j\) for each \(j = \text{front}^f, \ldots, \text{end}^f\) as

\[
\begin{align*}
  z_j &= \begin{cases} 
    1, & \text{if } e_{s_j} + \varepsilon_e - (x_{d_j}^f)^T w_s^f - b_s^f + y_{d_j} = 0, \\ 
    0, & \text{if } \alpha_j = 0. 
  \end{cases} \\
  z'_j &= \begin{cases} 
    1, & \text{if } e_{s_j} + \varepsilon_e + (x_{d_j}^f)^T w_s^f + b_s^f - y_{d_j} = 0, \\ 
    0, & \text{if } \beta_j = 0, 
  \end{cases} \\
  \eta_j &= \begin{cases} 
    1, & \text{if } e_{s_j} = 0, \\ 
    0, & \text{if } C_e - \alpha_j - \beta_j = 0. 
  \end{cases}
\end{align*}
\]

\(\vdots\)

\(\begin{align*}
  (27)
\end{align*}\)

---

\(^2\)Based on the numerical experiments provided in work [43], the semismooth method outperforms the interior method [42] specifically in solving the large-scale SVM classification problems.
Figure 5: Within the SVR context, a single data point can be labeled by its allocation in space.

Provided large numbers $\theta_1^j, \theta_2^j, \theta_3^j, \theta_4^j, \theta_5^j$, and $\theta_6^j$, an equivalent formulation of (7) can be written as:

$$\forall f = 1, \ldots, F :$$

$$w_s^f - \sum_{j=\text{front}^f}^{\text{end}^f} (\beta_j - \alpha_j) x_d^j = 0,$$

$$\sum_{j=\text{front}^f}^{\text{end}^f} \alpha_j - \sum_{j=\text{front}^f}^{\text{end}^f} \beta_j = 0,$$

$$\forall j = \text{front}^f \ldots \text{end}^f :$$

$$0 \leq \alpha_j \leq \theta_{1j} \cdot z_j,$$

$$0 \leq e_s^j + \varepsilon_e - (x_d^j)^T w_s^f - b_s^f + y_d^j \leq \theta_{2j} \cdot (1 - z_j),$$

$$0 \leq \beta_j \leq \theta_{3j} \cdot z'_j,$$

$$0 \leq e_s^j + \varepsilon_e + (x_d^j)^T w_s^f + b_s^f - y_d^j \leq \theta_{4j} \cdot (1 - z'_j),$$

$$0 \leq C_e - \alpha_j - \beta_j \leq \theta_{5j} \cdot \eta_j,$$

$$0 \leq e_s^j \leq \theta_{6j} \cdot (1 - \eta_j).$$

The values of $[z_j, z'_j, \eta_j] \forall j = \text{front}^f \ldots, \text{end}^f$ have important meaning geo-
metrically. As Figure 5 shows, the allocation of a single data point \((x^j_d, y^j_d)\) in the space of an SVR system has 5 possible locations: below the lower-hyperplane, above the upper-hyperplane, on the lower-hyperplane, on the upper-hyperplane, and inside the tube. The index sets \(A^f_1, A^f_2, A^f_3, A^f_4,\) and \(A^f_5\) can be defined by the binary solutions:

\[
\begin{align*}
A^f_1 & := \{ j \in \{ \text{front}^f, \ldots, \text{end}^f \} \mid (z_j, z'_j, \eta_j) = (1, 0, 0) \}, \\
A^f_2 & := \{ j \in \{ \text{front}^f, \ldots, \text{end}^f \} \mid (z_j, z'_j, \eta_j) = (0, 1, 0) \}, \\
A^f_3 & := \{ j \in \{ \text{front}^f, \ldots, \text{end}^f \} \mid (z_j, z'_j, \eta_j) = (1, 0, 1) \}, \\
A^f_4 & := \{ j \in \{ \text{front}^f, \ldots, \text{end}^f \} \mid (z_j, z'_j, \eta_j) = (0, 1, 1) \}, \text{ and} \\
A^f_5 & := \{ j \in \{ \text{front}^f, \ldots, \text{end}^f \} \mid (z_j, z'_j, \eta_j) = (0, 0, 1) \},
\end{align*}
\]

such that \(A^f_1 \cup A^f_2 \cup A^f_3 \cup A^f_4 \cup A^f_5 = \{ \text{front}^f, \ldots, \text{end}^f \}.\) Since a point can’t be allocated on both hyperplanes or on both sides, there are two natural cuts derived:

\[
\begin{align*}
z_j + z'_j & \leq 1, \quad \forall j = \text{front}^f, \ldots, \text{end}^f, f = 1, \ldots, F \\
z_j + z'_j + \eta_j & \geq 1, \quad \forall j = \text{front}^f, \ldots, \text{end}^f, f = 1, \ldots, F.
\end{align*}
\]

We define grouping and piece in the following.

**Definition 5.** A grouping \(G\) corresponding to a \((C_e, \varepsilon_e)\)-pair is a vector of integers whose \(j^{th}\) entry captures the spacial allocation of the \(j^{th}\) observation of training data. Let \((w^f_s, b^f_s) \forall f\) be the hyperplanes produced by applying Algorithm 3.1 to the lower-level problems fixed at \((C_e, \varepsilon_e).\) If \((x^j_d, y^j_d)\) is below the lower-hyperplane\(^3\), \(G_j = 1.\) If \((x^j_d, y^j_d)\) is above the upper-hyperplane\(^4\), \(G_j = 2.\) If \((x^j_d, y^j_d)\) is on the lower-hyperplane, \(G_j = 3.\) If \((x^j_d, y^j_d)\) is on the upper-hyperplane, \(G_j = 4.\) If \((x^j_d, y^j_d)\) is inside the \(\varepsilon_e\)-tube, \(G_j = 5.\)

**Definition 6.** A piece of the \(\text{LCP}_{SVR}^f\) is a set of linear equality and inequality constraints that result from fixing the binary variables \([z_j, z'_j, \eta_j]\) at one of the following values:

\([1, 0, 0], [0, 1, 0], [1, 0, 1], [0, 1, 1], \text{ or } [0, 0, 1]\)

for each \(j,\) in model (28).

\(^3\)lower-hyperplane: \(y_{d_j} = (x^j_d)^T w^f_s + b^f_s - \varepsilon_e\)

\(^4\)upper-hyperplane: \(y_{d_j} = (x^j_d)^T w^f_s + b^f_s + \varepsilon_e\)
A grouping vector has dimension $n$. There are at most $5^n$ possible grouping vectors for $n$ training data points, regardless of the choices of $C_e$ and $\varepsilon_e$. Given a grouping, the following is a corresponding piece:

$$\mathbf{w}_s^f + \sum_{j=\text{front}}^{\text{end}} \alpha_j \mathbf{x}_d^j - \sum_{j=\text{front}}^{\text{end}} \beta_j \mathbf{x}_d^j = 0,$$

$$\sum_{j=\text{front}}^{\text{end}} \alpha_j - \sum_{j=\text{front}}^{\text{end}} \beta_j = 0,$$

(31)

$$\begin{align*}
(x_d^j)^T \mathbf{w}_s^f + b_s^f - y_{dj} - \varepsilon_e & \geq 0, \\
\alpha_j &= C_e, \beta_j = 0, \forall j \in A_1^f,
\end{align*}$$

(32)

$$\begin{align*}
y_{dj} - (x_d^j)^T \mathbf{w}_s^f - b_s^f - \varepsilon_e & \geq 0, \\
\alpha_j = 0, \beta_j = C_e, \forall j \in A_2^f,
\end{align*}$$

(33)

$$\begin{align*}
C_e & \geq \alpha_j \geq 0, \beta_j = 0, \\
y_{dj} = (x_d^j)^T \mathbf{w}_s^f + b_s^f - \varepsilon_e, \forall j \in A_3^f,
\end{align*}$$

(34)

$$\begin{align*}
\alpha_j = 0, C_e & \geq \beta_j \geq 0, \\
y_{dj} = (x_d^j)^T \mathbf{w}_s^f + b_s^f + \varepsilon_e, \forall j \in A_4^f,
\end{align*}$$

(35)

$$\begin{align*}
\varepsilon_e & - (x_d^j)^T \mathbf{w}_s^f - b_s^f + y_{dj} \geq 0, \\
\varepsilon_e & - y_{dj} + (x_d^j)^T \mathbf{w}_s^f + b_s^f \geq 0, \\
\alpha_{j'} = 0, \beta_{j'} = 0, \forall j \in A_5^f.
\end{align*}$$

(36)

Thus, by recording the vector of the grouping, the piece can be monitored and recovered on the fly. It is noteworthy that given a piece, more than one data point allocation can be feasible under the constraints. Consider an arbitrary index $j'$ and a piece defined by

$$\begin{align*}
\varepsilon_e & - (x_d^j)^T \mathbf{w}_s - b_s + y_{dj'} \geq 0, \\
\varepsilon_e & - y_{dj'} + (x_d^j)^T \mathbf{w}_s + b_s \geq 0, \\
\alpha_{j'} = 0, \beta_{j'} = 0.
\end{align*}$$

(37)

We can see that $\{y_{dj'} = (x_d^j)^T \mathbf{w}_s + b_s - \varepsilon_e, \alpha_{j'} = 0, \beta_{j'} = 0\}$ is a feasible solution.
to (37). However, provided that \( \{ y_{dj'} = (x_{dj}')^T w_s + b_s - \varepsilon - \alpha_{j'} = 0, \beta_{j'} = 0 \} \), \( j' \) is eligible to be contained in \( A_5, A_4 \) and \( A_3 \). This phenomenon occurs when there is degeneracy, i.e., the two sides of the complementarity are both zero.

In the implementation, we follow the algorithm below to transform the solution of the lower-level problem of the LCP with a fixed \((C_e, \varepsilon_e)\) into a grouping vector for storage.

**Algorithm 3.2 Transform solutions \((\alpha, \beta, e_s, w_f, b_f)\) to a grouping vector.**

Given solutions \( \alpha, \beta, e_s, w_f \) and \( b_f \). Declare \( GroupingV \) as a vector with a length of \( n \). Let \( A_{f 1}, A_{f 2}, A_{f 3}, A_{f 4} \) and \( A_{f 5} = \emptyset \).

for \( f = 1, \ldots, F \)

for \( j = \text{front}^f, \ldots, \text{end}^f \)

if \( \alpha_j > e_s_j + \varepsilon - (x_{dj}')^T w_s + b_s - y_{dj} \)

if \( e_s_j > C_e - \alpha_j - \beta_j \)

\( GroupingV_j = 1, \) and \( A_{f 1} \leftarrow A_{f 1} \cup \{j\} \).

otherwise

\( GroupingV_j = 3, \) and \( A_{f 3} \leftarrow A_{f 3} \cup \{j\} \).

end

otherwise

if \( e_s_j + \varepsilon + (x_{dj}')^T w_s + b_f - y_{dj} > \beta_j \)

\( GroupingV_j = 5, \) and \( A_{f 5} \leftarrow A_{f 5} \cup \{j\} \).

otherwise

if \( C_e - \alpha_j - \beta_j > e_s_j \)

\( GroupingV_j = 4, \) and \( A_{f 4} \leftarrow A_{f 4} \cup \{j\} \).

otherwise

\( GroupingV_j = 2, \) and \( A_{f 2} \leftarrow A_{f 2} \cup \{j\} \).

end

end

end for
We define the invariance region in the context of this work and show that it is convex.

**Definition 7.** An invariance region \( \mathcal{IR} \) is a region on the parameter space, i.e., the \((C_e, \varepsilon_e)\)-plane, such that the grouping vector induced by every \((C_e, \varepsilon_e) \in \mathcal{IR}\) is the same.

**Theorem 8.** Consider the following process 1)-3): 1) Solving for a solution to LCP\(_{SVR}^f\) (7) with \((C_e, \varepsilon_e)\) fixed at \((\bar{C}_e, \bar{\varepsilon}_e)\) by Algorithm 3.1. 2) Transforming the solution to a grouping vector \(G\) by Algorithm 3.2. 3) Using the grouping \(G\) to form index sets \(A_i\) \(\forall i = 1, \ldots, 5\) and form a piece \(P\) as (31)-(36). Let invariance region \(\mathcal{IR}\) be a set of \((C_e, \varepsilon_e)\)-pairs such that the grouping vectors equal to \(G\). Then \(\mathcal{IR}\) is a convex set.

**Proof.** Without loss of generality, let \(F = 1\) and ignore the superscripts \(f\) in the notation of variables. Let \((\bar{C}_{e(1)}, \bar{\varepsilon}_{e(1)}) \in \mathcal{IR}\) and \((\bar{C}_{e(2)}, \bar{\varepsilon}_{e(2)}) \in \mathcal{IR}\). Let the solution to LCP\(_{SVR}^f\) with \((C_e, \varepsilon_e)\) fixed at \((\bar{C}_{e(1)}, \bar{\varepsilon}_{e(1)})\) and \((\bar{C}_{e(2)}, \bar{\varepsilon}_{e(2)})\) be \(\{\alpha_{j(1)}, \beta_{j(1)}, e_{j(1)}, w_{s(1)}, b_{s(1)}\}\) and \(\{\alpha_{j(2)}, \beta_{j(2)}, e_{j(2)}, w_{s(2)}, b_{s(2)}\}\) respectively. Assume that they give the same grouping, i.e., \(A_{i(1)} = A_{i(2)}, \forall i = 1, \ldots, 5\), and that the correspondent pieces \(P\) are given as (31)-(36). For any arbitrary \(\lambda \in (0, 1)\), consider \((C_{e(3)}, \varepsilon_{e(3)}) = (\lambda \bar{C}_{e(1)} + (1 - \lambda) \bar{C}_{e(2)}, \lambda \bar{\varepsilon}_{e(1)} + (1 - \lambda) \bar{\varepsilon}_{e(2)})\). Since \((\bar{C}_{e(1)}, \bar{\varepsilon}_{e(1)})\) and \((\bar{C}_{e(2)}, \bar{\varepsilon}_{e(2)})\) produce the same groupings, we claim that the solution \(\{\alpha_{j(3)}, \beta_{j(3)}, e_{j(3)}, w_{s(3)}, b_{s(3)}\}\) equals to \(\{\lambda \alpha_{j(1)} + (1 - \lambda) \alpha_{j(2)}, \lambda \beta_{j(1)} + (1 - \lambda) \beta_{j(2)}, \lambda e_{j(1)} + (1 - \lambda) e_{j(2)}, \lambda w_{s(1)} + (1 - \lambda) w_{s(2)}, \lambda b_{s(1)} + (1 - \lambda) b_{s(2)}\}\) because the following system
can be satisfied by it.

\[
\mathbf{w}_s(3) - \sum_{j=1}^{n} (\beta_j(3) - \alpha_j(3)) \mathbf{x}_d = 0, \\
\sum_{j=1}^{n} \alpha_j(3) - \sum_{j=1}^{n} \beta_j(3) = 0, \\
\forall j = 1, \ldots, n :
\begin{cases}
0 \leq e_{sj}(3) + [\lambda \varepsilon_{e(1)} + (1 - \lambda) \varepsilon_{e(2)}] - (\mathbf{x}_d^j)^T \mathbf{w}_s(3) - b_s(3) + y_{d_i} & \perp \alpha_j(3) \geq 0, \\
0 \leq e_{sj}(3) + [\lambda \varepsilon_{e(1)} + (1 - \lambda) \varepsilon_{e(2)}] + (\mathbf{x}_d^j)^T \mathbf{w}_s(3) + b_s(3) - y_{d_i} & \perp \beta_j(3) \geq 0, \\
0 \leq [\lambda C_{e(1)} + (1 - \lambda) C_{e(2)}] - \alpha_j(3) - \beta_j(3) & \perp e_{sj}(3) \geq 0.
\end{cases}
\]

The grouping for \( \{\alpha_j(3), \beta_j(3), e_{j}(3), \mathbf{w}_s(3), b_s(3)\} \) is again the same. So \( \{C_{e(3)}, \varepsilon_{e(3)}, \alpha_j(3), \beta_j(3), e_{j}(3), \mathbf{w}_s(3), b_s(3)\} \) is feasible to \( \mathcal{P} \), and \( \mathcal{IR} \) is convex.

The following property directly results from Theorem 8, which is one of the sufficient conditions we use in the algorithm to claim that all the invariancy regions inside a rectangular area has been found.

**Proposition 9.** Given a rectangle\(^5\) on the \((C_{e}, \varepsilon_{e})\)-plane, if its four corner points produce the same vector of grouping, the whole rectangle all produce the same vector of the grouping.

**Proof.** The invariancy regions \( \mathcal{IR} \) are convex. \( \square \)

### 3.3 Restricted LP, Reduced Restricted LP, and Restricted QCP

In this section we introduce three types of restricted programs: restricted linear program, reduced restricted linear program, and restricted quadratic constrained program. They are called restricted in the sense of fixing at the invariancy region of \((C_{e}, \varepsilon_{e})\) (in the cases of Restricted LP and Reduced Restricted LP) or at a single \((C_{e}, \varepsilon_{e})\)-pair (in the case of Restricted QCP).

A restricted linear program \( \mathbb{R}LP \) of the LPCC (9) is obtained by replacing the

---

\(^5\)A rectangle is defined by four bounds: upper and lower bounds of \( C_{e} \) and \( \varepsilon_{e} \).
LCP\textsuperscript{f}_{SVR} with one of its pieces. Since the input of RLP are index sets \(A^f_i\) for all \(i = 1, \ldots, 5\) and \(f = 1, \ldots, F\), there are \(\prod_{f=1}^F 5^{n_f}\) many RLP defined as follows:

\[
\text{RLP}(A^f_i \mid i = 1, \ldots, 5, f = 1, \ldots, F) : \\
\min_{c, \varepsilon, \varepsilon_e, w^f_s, b^f_s, p_i, \alpha_j, \beta_j} \sum_{f=1}^F \sum_{i=\text{front}_f^l}^{\text{end}_f^l} p_i \\
\text{subject to } 0 \leq c \leq C_e \leq \overline{c}, \\
0 \leq \varepsilon \leq \varepsilon_e \leq \overline{\varepsilon}, \\
\alpha_j \in A^f_1, \beta_j \in A^f_2, \\
\alpha_j = 0, \beta_j = C_e, \\
\varepsilon_e - (x^d_j)^T w^f_s + b^f_s + y_{d_j} \geq 0, \\
\alpha_j = 0, \beta_j = 0.
\]

If we look closely at the model (38), \(\alpha_j \in A^f_1\) and \(\beta_j \in A^f_2\) can be replaced by a single
variable $C_e$. The variables $\alpha_j$, $\forall j \in A^f_2, A^f_4, A^f_5$, and $\beta_j$, $\forall j \in A^f_1, A^f_3, A^f_5$, can be eliminated. Thus, we obtain a reduced restricted linear program $\text{RRLP}$ as follows:

$$ \text{RRLP}(A^f_i \mid i = 1, \ldots, 5, f = 1, \ldots, F) : $$

$$ \min_{C_e, \varepsilon_e, w^f_s, b^f_s, p_i} \sum_{f=1}^{F} \sum_{i=\text{front}^f_s}^{\text{end}^f_s} p_i $$

subject to $0 \leq C \leq C_e \leq C$, $0 \leq \varepsilon \leq \varepsilon_e \leq \varepsilon$,

and $\forall f = 1, \ldots, F$:

$$ \begin{cases} (x^d_i)^T w^f_s + b^f_s - y_d \leq p_i, \forall i = \text{front}^f_s, \ldots, \text{end}^f_s, \\ - (x^d_i)^T w^f_s - b^f_s + y_d \leq p_i, \forall i = \text{front}^f_s, \ldots, \text{end}^f_s, \\ w^f_s + C_e \sum_{j \in A^f_1} x^d_j - C_e \sum_{j \in A^f_2} x^d_j + \sum_{j \in A^f_3} \alpha_j x^d_j - \sum_{j \in A^f_4} \beta_j x^d_j = 0, \\ |A^f_1| C_e - |A^f_2| C_e + \sum_{j \in A^f_3} \alpha_j - \sum_{j \in A^f_4} \beta_j = 0, \\ (x^d_i)^T w^f_s + b^f_s - y_{d_j} - \varepsilon_e \geq 0, \forall j \in A^f_1, \\ y_{d_j} - (x^d_i)^T w^f_s - b^f_s - \varepsilon_e \geq 0, \forall j \in A^f_2, \\ C_e \geq \alpha_j \geq 0, \beta_j = 0, \forall j \in A^f_3, \\ y_{d_j} = (x^d_i)^T w^f_s + b^f_s - \varepsilon_e, \forall j \in A^f_3, \\ \alpha_j = 0, C_e \geq \beta_j \geq 0, \forall j \in A^f_4, \\ y_{d_j} = (x^d_i)^T w^f_s + b^f_s, \forall j \in A^f_4, \\ \varepsilon_e - (x^d_i)^T w^f_s - b^f_s + y_{d_j} \geq 0, \forall j \in A^f_5, \\ \varepsilon_e - y_{d_j} + (x^d_i)^T w^f_s + b^f_s \geq 0, \forall j \in A^f_5, \end{cases} $$

(39)

where $|A^f_i|$ denotes the cardinality of the set $A^f_i$.

Except for the two linear restricted programs, a fixed $(C_e, \varepsilon_e)$ pair allows us to formulate a restricted convex quadratically constrained program $\text{RQCP}$ as follows:
$\text{RQCP}(C^f_{\text{fix}}, \varepsilon^f_{\text{fix}})$:

$$\begin{aligned}
\min_{w^f_s, b^f_s, p_i, \alpha_j, \beta_j} \quad & \sum_{f=1}^{F} \sum_{i=\text{front}^f_s}^{\text{end}^f_s} p_i \\
\text{subject to} \quad & \forall f = 1, \ldots, F:
\begin{cases}
(x^1_d)^T w^f_s + b^f_s - y_{d_i} \leq p_i, \forall i = \text{front}^f_s, \ldots, \text{end}^f_s, \\
-(x^1_d)^T w^f_s - b^f_s + y_{d_i} \leq p_i, \forall i = \text{front}^f_s, \ldots, \text{end}^f_s, \\
w^f_s + \sum_{j=\text{front}^f_s}^{\text{end}^f_s} \alpha_j x^j_d - \sum_{j=\text{front}^f_s}^{\text{end}^f_s} \beta_j x^j_d = 0, \\
\sum_{j=\text{front}^f_s}^{\text{end}^f_s} \alpha_j - \sum_{j=\text{front}^f_s}^{\text{end}^f_s} \beta_j = 0, \\
\left( \sum_{j=\text{front}^f_s}^{\text{end}^f_s} \varepsilon^f_{d_j} \right) C^f_{\text{fix}} + \left( \sum_{j=\text{front}^f_s}^{\text{end}^f_s} (\alpha_j + \beta_j) \right) \varepsilon^f_{\text{fix}} \\
+ \left( \sum_{j=\text{front}^f_s}^{\text{end}^f_s} (\beta_j - \alpha_j) (x^j_d)^T \right) \left( \sum_{i=\text{front}^f_s}^{\text{end}^f_s} (\beta_i - \alpha_i) (x^i_d) \right) \\
+ \sum_{j=\text{front}^f_s}^{\text{end}^f_s} (\alpha_j - \beta_j) y_j = 0,
\end{cases}
\end{aligned}$$

(40)

where the convex quadratic constraints are the aggregation of complementarities. We postpone the details on obtaining these convex quadratic constraints to Section 4.

The objective values obtained from solving every RLP (or RRLP or RQCP) are the upper bounds of the problem (9). The problem $\text{RQCP}(C^f_{\text{fix}}, \varepsilon^f_{\text{fix}})$ is theoretically more restricted than solving $\text{LCP}^f_{\text{SV R}}$ followed by RLP($A^f_i \mid i = 1, \ldots, 5, f = 1, \ldots, F$) or RRLP($A^f_i \mid i = 1, \ldots, 5, f = 1, \ldots, F$), yet the vector of the grouping obtained from either method is identical. Thus, the problem RQCP can be a substitute for $\text{LCP}^f_{\text{SV R}}$ in order to get the grouping and the piece. Based on our numerical experiments, solving an RQCP is more time-consuming than solving an $\text{LCP}^f_{\text{SV R}}$ plus an RLP. We only use RQCP when $\text{LCP}^f_{\text{SV R}}$ or RLP fails to be solved.
3.4 Invariancy interval along a chosen line

To identify the invariancy interval on a line is not as complicated as the work (See the methods in [52, 13]) of identifying the invariancy region of a point \((C_e, \varepsilon_e)\). The line passing through a point \((C_e, \varepsilon_e) = (\hat{C}, \hat{\varepsilon})\) is either of the form

\[
\varepsilon_e = L_a C_e + L_b,
\]

where \(L_a\) and \(L_b\) are the slope and intercept such that the line passes through \((\hat{C}, \hat{\varepsilon})\), or it’s a vertical line as

\[
C_e = \hat{C}.
\]

The invariancy interval \([\hat{C}_1, \hat{\varepsilon}_1), (\hat{C}_2, \hat{\varepsilon}_2)]\) can be obtained by solving the following four linear optimization problems:

\[
C_{\text{max}} \setminus C_{\text{min}} = \max \setminus \min C_e
\]

subject to \(\varepsilon_e = L_a C_e + L_b,\) (or \(C_e = \hat{C}\))

and constraints in (38),

and

\[
\varepsilon_{\text{max}} \setminus \varepsilon_{\text{min}} = \max \setminus \min \varepsilon_e
\]

subject to \(\varepsilon_e = L_a C_e + L_b,\) (or \(C_e = \hat{C}\))

and constraints in (38).

The solution to (43) and (44) is a line segment that belongs in one of the following four cases:

(i) On \(C_e = \hat{C}\) (a vertical line): \(\hat{C}_1 = \hat{C}, \hat{\varepsilon}_1 = \varepsilon_{\text{max}}, \hat{C}_2 = \hat{C},\) and \(\hat{\varepsilon}_2 = \varepsilon_{\text{min}}.\)

(ii) On \(\varepsilon_e = L_a C_e + L_b\) where \(L_a = 0, L_b = \hat{\varepsilon}\) (a horizontal line): \(\hat{C}_1 = C_{\text{max}}, \hat{\varepsilon}_1 = \hat{\varepsilon}, \hat{C}_2 = C_{\text{min}},\) and \(\hat{\varepsilon}_2 = \hat{\varepsilon}.\)

(iii) When \(L_a\) is positive: \(\hat{C}_1 = C_{\text{max}}, \hat{\varepsilon}_1 = \varepsilon_{\text{max}}, \hat{C}_2 = C_{\text{min}},\) and \(\hat{\varepsilon}_2 = \varepsilon_{\text{min}}.\)

(iv) When \(L_a\) is negative: \(\hat{C}_1 = C_{\text{max}}, \hat{\varepsilon}_1 = \varepsilon_{\text{min}}, \hat{C}_2 = C_{\text{min}},\) and \(\hat{\varepsilon}_2 = \varepsilon_{\text{max}}.\)

We propose a procedure to find all the groupings along the boundaries of a given rectangle \(\{\hat{C}, \underline{C}, \hat{\varepsilon}, \underline{\varepsilon}\}\). When searching along a boundary line, either \(\varepsilon_e\) or \(C_e\)
is fixed at the corresponding value. The procedure starts with solving for the grouping at a vertex of the boundary and finding the invariancy interval $[C_{\text{min}}, C_{\text{max}}]$ or $[\varepsilon_{\text{min}}, \varepsilon_{\text{max}}]$. After obtaining the endpoints of the interval, to compute a new grouping vector, we select a point that is outside and deviates a very small amount from the endpoint of the current interval. The deviation needs to be small enough to make sure that no groupings are missed and that the invariancy interval containing the point is adjacent to the original interval. During the process, the numbers of groupings on each side of the boundary line are recorded by $\text{countTop}$, $\text{countLeft}$, $\text{countBottom}$, and $\text{countRight}$. At the end of the procedure, a list of grouping vectors $\text{GroupingVFound}$ and the least objective value $\text{LeastUpperBound}$ are obtained.

Algorithm 3.3 Identifying groupings on boundaries of a rectangle.

**Step 0. Input.**

Set the parameter $\text{deviation}$ (0.0001 for example).

Initialize the $\text{countTop}$, $\text{countLeft}$, $\text{countBottom}$, and $\text{countRight} = 0$.

Initialize $\text{LeastUpperBound}$ at any valid upper bound.

Initialize the set $\text{GroupingVFound}$.

Exogenously find the grouping vectors at the four corners $(\check{C}, \check{\varepsilon}), (\bar{C}, \bar{\varepsilon}), (\check{C}, \bar{\varepsilon})$, and $(\bar{C}, \check{\varepsilon})$: $\text{GroupingV}^{\text{uu}}$, $\text{GroupingV}^{\text{ll}}$, $\text{GroupingV}^{\text{ul}}$, and $\text{GroupingV}^{\text{lu}}$ respectively.

**Step 1. Search on the horizontal line $\varepsilon_e = \check{\varepsilon}$.**

Initialize $\text{GroupingV}^{\text{top}} = \text{GroupingV}^{\text{uu}}$.

1a: Solve (43) subject to $\varepsilon_e = \check{\varepsilon}$ and the piece corresponding to $\text{GroupingV}^{\text{top}}$ to obtain the invariancy interval $[C_{\text{min}}, C_{\text{max}}]$.

1b: If $C_{\text{min}}$ is greater than $\bar{C}$, let $\text{newStarting} = (C_{\text{min}} - \text{deviation}, \check{\varepsilon})$.

1c: Solve $\text{LCP}_{\text{SVR}}$ at $\text{newStarting}$. $\text{countTop} + 1$.

Let $\text{GroupingV}^{\text{top}}$ be the obtained grouping vector. If it is not in the set $\text{GroupingVFound}$, add it to the set and solve the corresponding $\text{RLP}$.

1d: If the objective value is smaller than $\text{LeastUpperBound}$, update $\text{LeastUpperBound}$.

1e: Repeat 1a-1d until $C_{\text{min}} = \bar{C}$.

**Step 2. Search on the vertical line $C_e = \check{C}$.**
Initialize the $\text{GroupingV}^{left} = \text{GroupingV}^{lu}$.

2a: Solve (44) subject to $C_e = \bar{C}$ and the piece corresponding to $\text{GroupingV}^{left}$.

to obtain the invariancy interval $[\varepsilon_{\min}, \varepsilon_{\max}]$.

2b: If $\varepsilon_{\min}$ is greater than $\varepsilon$, let $\text{newStarting} = (\bar{C}, \varepsilon_{\min} - \text{deviation})$.

2c: Solve $\mathbb{LCP}_S^{fr}$ at $\text{newStarting}$. $\text{countLeft} + 1$.

Let $\text{GroupingV}^{left}$ be the obtained grouping vector. If it is not in the set $\text{GroupingV Found}$, add it to the set and solve the corresponding $\mathbb{RLP}$.

2d: If the objective value is smaller than $\text{LeastUpperBound}$, update $\text{LeastUpperBound}$.

2e: Repeat 2a-2d until $\varepsilon_{\min} = \varepsilon$.

**Step 3. Search on the horizontal line** $\varepsilon_e = \varepsilon$.

Initialize $\text{GroupingV}^{bottom} = \text{GroupingV}^{ul}$.

3a: Solve (43) subject to $\varepsilon_e = \varepsilon$ and the piece corresponding to $\text{GroupingV}^{bottom}$.

to obtain the invariancy interval $[C_{\min}, C_{\max}]$.

3b: If $C_{\min}$ is greater than $\bar{C}$, let $\text{newStarting} = (C_{\min} - \text{deviation}, \varepsilon)$.

3c: Solve $\mathbb{LCP}_S^{fr}$ at $\text{newStarting}$. $\text{countBottom} + 1$.

Let $\text{GroupingV}^{bottom}$ be the obtained grouping vector. If it is not in $\text{GroupingV Found}$, add it to the set and solve the corresponding $\mathbb{RLP}$.

3d: If the objective value is smaller than $\text{LeastUpperBound}$, update $\text{LeastUpperBound}$.

3e: Repeat 3a-3d until $C_{\min} = \bar{C}$.

**Step 4. Search on the vertical line** $C_e = \bar{C}$.

Initialize the $\text{GroupingV}^{right} = \text{GroupingV}^{uu}$.

4a: Solve (44) subject to $C_e = \bar{C}$ and the piece corresponding to $\text{GroupingV}^{right}$.

to obtain the invariancy interval $[\varepsilon_{\min}, \varepsilon_{\max}]$.

4b: If $\varepsilon_{\min}$ is greater than $\varepsilon$, let $\text{newStarting} = (\bar{C}, \varepsilon_{\min} - \text{deviation})$.

4c: Solve $\mathbb{LCP}_S^{fr}$ at $\text{newStarting}$. $\text{countRight} + 1$.

Let $\text{GroupingV}^{right}$ be the obtained grouping vector. If it is not in the set $\text{GroupingV Found}$, add it to the set and solve the corresponding $\mathbb{RLP}$.

4d: If the objective value is smaller than $\text{LeastUpperBound}$, update $\text{LeastUpperBound}$.

4e: Repeat 4a-4d until $\varepsilon_{\min} = \varepsilon$. 

We have noticed from the experiments that the invariancy intervals obtained at Step 1a, 2a, 3a, and 4a are sometimes problematic because of complementarity degeneracy or arithmetic error. Figure 6 shows two examples of the problematic intervals on a horizontal boundary. In these examples, the second intervals ($I^B$) are not properly adjacent to the first intervals ($I^A$). Figure 7 illustrates an invariancy interval $I^B = [C^B_{\text{min}}, C^B_{\text{max}}]$ which appears to be located appropriately, given the location of its adjacent interval $I^A = [C^A_{\text{min}}, C^A_{\text{max}}]$. An interval $I^B$ is said to be appropriately located adjacent to the previous interval $I^A$ if:

$$C^B_{\text{max}} \leq C^A_{\text{min}} \text{ and } C^B_{\text{min}} < C^B_{\text{max}}.$$ (45)

We can see that the locations of the intervals in Figure 6 are not appropriate since

Overlapping

Repeating

Figure 6: Problematic invariancy intervals $I^B = [C^B_{\text{min}}, C^B_{\text{max}}]$ subsequent to the interval $I^A$ on a line with fixed $\varepsilon_e$

Figure 7: The appropriate invariancy interval $I^B = [C^B_{\text{min}}, C^B_{\text{max}}]$ subsequent and adjacent to the interval $I^A$ on a line with fixed $\varepsilon_e$
$C_{\text{max}}^B > C_{\text{min}}^B > C_{\text{min}}^A$ in the case of overlapping, and $C_{\text{max}}^B = C_{\text{min}}^B$ in the case of repeating. The former case is usually due to complementarity degeneracy, while the latter case can be a natural result of a single-valued invariancy interval. The former case can cause an endless loop in Algorithm 3.4 and the later case would be a problem in counting the exact numbers of the invariancy regions, which is significant to the 2nd stage of the algorithm (that will be seen in Section 3.6).

We thus propose a small add-in to screen out or modify the problematic intervals violating the conditions of (45).

The following add-in procedure should be adopted in Algorithm 3.4 after Step 1a and before 1b:

**Add-In: Enforcing the Forward Searching (between Step 1a and 1b).**

0. **Input:** The invariancy interval obtained from Step 1a of Algorithm 3.4: $[C_{\text{min}}, C_{\text{max}}]$. The previously\(^\text{6}\) obtained adjacent invariancy interval $[C_{\text{adj}}^\text{min}, C_{\text{adj}}^\text{max}]$. Parameter *perturbation* (say 0.001). Parameter *counter* = 1. Other parameters and containers used in Algorithm 3.4.

1. **Check:** If $C_{\text{max}} \leq C_{\text{adj}}^\text{min}$ and $C_{\text{min}} < C_{\text{max}}$, this is an appropriate invariancy interval. Stop.

2. **Perturb:** Let $\text{newStarting} = C_{\text{adj}}^\text{min} - \text{deviation} - \text{counter} \times \text{perturbation}$.

Solve $\mathbb{LCP}_{SVR}^f$ at $\text{newStarting}$ and obtain the grouping vector $GroupingV^\text{top}$. Do the same thing as in Step 1c and 1d in Algorithm 3.4.

3. **Solve for a new interval:** Solve (43) subject to $\varepsilon_e = \bar{\varepsilon}$ and the piece out of $GroupingV^\text{top}$. Obtain the invariancy interval $[C_{\text{min}}, C_{\text{max}}]$. *counter* + 1. Go to 1.

The procedures used between the steps 2a and 2b, 3a and 3b, and 4a and 4b are similar.

Figure 8 provides a numerical example of using Algorithm 3.4 with the add-in to identify the groupings along the four boundaries: boundary on the top ($\varepsilon_e = \bar{\varepsilon}$), left-hand-side boundary ($C_e = \bar{C}$), bottom boundary ($\varepsilon_e = \underline{\varepsilon}$), and right-hand-side boundary ($C_e = \bar{C}$). In Figure 8, all appropriate and problematic intervals are sequentially shown. The parameter *deviation* is set at 0.0001 and *perturb* is set at

\(^{6}\)If the interval $[C_{\text{min}}, C_{\text{max}}]$ is the first interval found on that boundary, there is no previous interval and no need to run the add-in.
0.001. For this specific instance with 2 folds, each fold has 35 training and 35 testing data points with 5 features, given $C = 10$, $C_e = 1$, $\varepsilon = 1$, and $\varepsilon_e = 0.1$. Among the intervals obtained, intervals #12, #20, #25, #26, #27, #28, #30, #32, #33, #35, #36, #38, #39, and #41 are problematic intervals at which the add-in procedure is applied to enforce forward searching. These problematic intervals are not counted in the number of groupings at each side of boundary, so at the end of the algorithm, we obtained $\text{countTop}$, $\text{countLeft}$, $\text{countBottom}$, and $\text{countRight}$ at 2, 13, 2, and 11 respectively in the rectangular area $[1, 10] \times [0.1, 1]$.

<table>
<thead>
<tr>
<th>along the top boundary</th>
<th>leastUpperBound</th>
<th>along the bottom boundary</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C_e$</td>
<td>$\varepsilon_e$</td>
<td>$C_{min}$</td>
</tr>
<tr>
<td>#1</td>
<td>10</td>
<td>0.5</td>
</tr>
<tr>
<td>2</td>
<td>1.01179</td>
<td>0.5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>along the left-hand-side boundary</th>
<th>leastUpperBound</th>
<th>along the right-hand-side boundary</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C_e$</td>
<td>$\varepsilon_e$</td>
<td>$C_{min}$</td>
</tr>
<tr>
<td>#3</td>
<td>1</td>
<td>0.5</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>0.498064</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>0.495807</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>0.462789</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>0.442832</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>0.425836</td>
</tr>
<tr>
<td>9</td>
<td>1</td>
<td>0.313019</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>0.310835</td>
</tr>
<tr>
<td>11</td>
<td>1</td>
<td>0.223918</td>
</tr>
<tr>
<td>12</td>
<td>1</td>
<td>0.212199</td>
</tr>
<tr>
<td>13</td>
<td>1</td>
<td>0.212209</td>
</tr>
<tr>
<td>14</td>
<td>1</td>
<td>0.156503</td>
</tr>
<tr>
<td>15</td>
<td>1</td>
<td>0.135792</td>
</tr>
<tr>
<td>16</td>
<td>1</td>
<td>0.129056</td>
</tr>
<tr>
<td>#17</td>
<td>10</td>
<td>0.5</td>
</tr>
<tr>
<td>18</td>
<td>10</td>
<td>0.495807</td>
</tr>
<tr>
<td>19</td>
<td>10</td>
<td>0.495817</td>
</tr>
<tr>
<td>20</td>
<td>10</td>
<td>0.494817</td>
</tr>
<tr>
<td>21</td>
<td>10</td>
<td>0.462789</td>
</tr>
<tr>
<td>22</td>
<td>10</td>
<td>0.442832</td>
</tr>
<tr>
<td>23</td>
<td>10</td>
<td>0.425836</td>
</tr>
<tr>
<td>24</td>
<td>10</td>
<td>0.313019</td>
</tr>
<tr>
<td>25</td>
<td>10</td>
<td>0.310835</td>
</tr>
<tr>
<td>26</td>
<td>10</td>
<td>0.223918</td>
</tr>
<tr>
<td>27</td>
<td>10</td>
<td>0.212199</td>
</tr>
<tr>
<td>28</td>
<td>10</td>
<td>0.212209</td>
</tr>
<tr>
<td>29</td>
<td>10</td>
<td>0.156503</td>
</tr>
<tr>
<td>30</td>
<td>10</td>
<td>0.135792</td>
</tr>
<tr>
<td>31</td>
<td>10</td>
<td>0.129056</td>
</tr>
<tr>
<td>32</td>
<td>10</td>
<td>0.223918</td>
</tr>
<tr>
<td>33</td>
<td>10</td>
<td>0.212199</td>
</tr>
<tr>
<td>34</td>
<td>10</td>
<td>0.212209</td>
</tr>
<tr>
<td>35</td>
<td>10</td>
<td>0.156503</td>
</tr>
<tr>
<td>36</td>
<td>10</td>
<td>0.135792</td>
</tr>
<tr>
<td>37</td>
<td>10</td>
<td>0.129056</td>
</tr>
<tr>
<td>38</td>
<td>10</td>
<td>0.146469</td>
</tr>
<tr>
<td>39</td>
<td>10</td>
<td>0.145479</td>
</tr>
<tr>
<td>40</td>
<td>10</td>
<td>0.144479</td>
</tr>
<tr>
<td>41</td>
<td>10</td>
<td>0.132445</td>
</tr>
<tr>
<td>42</td>
<td>10</td>
<td>0.131455</td>
</tr>
</tbody>
</table>

Figure 8: An example (35, 35, 5, 2) of searching along the boundaries of a rectangle and finding the invariancy intervals using the Algorithm 3.4 with the add-in.

### 3.5 The 1st stage: identify rectangles of the same grouping

The 1st stage of the $(C_e, \varepsilon_e)$-rectangle search algorithm explores possible grouping vectors and the corresponding objective values of restricted linear programs for
each rectangle $[C, \bar{C}] \times [\varepsilon, \bar{\varepsilon}]$ in the queue. The search of groupings and invariance intervals starts at the four vertices and along the boundaries of the rectangles. If we can’t conclude that all groupings in this rectangle are realized, we choose an interior point of the rectangle and a direction along which the invariance interval is computed, then partition at one of the endpoints of the invariance interval. The process proceeds to sequentially search the rectangular areas decomposed from the initial box.

By Proposition 9, a rectangular area is guaranteed to have only one grouping when the same grouping vector is obtained at its four vertices. A rectangular area with this property requires no more partitioning and is eliminated from a list of $AreaToBeSearch$ (the Step 2d). The total area of the eliminated rectangles is recorded at $AreaRealizedInThe1stStage$. At the end of the 1st stage, a least upper bound ($LeastUpperBound$) for the cross-validated SVR model is obtained.

**Algorithm 3.4 The 1st Stage**

**Step 0. Declaration.**

Initialize the parameters $m$, $n$, $F$, and $K$. $\bar{C}$, $C$, $\bar{\varepsilon}$, and $\varepsilon$.

Initialize the tuple $Area := \{C, \bar{C}, \bar{\varepsilon}, \varepsilon\}$.

Initialize the list $AreasToBeSearched = \{Area\}$.

Initialize the set $GroupingVFound = \emptyset$.

Initialize $LeastUpperBound = 0$.

Initialize $AreaRealizedInThe1stStage = 0$.

Initialize the parameter insensitive (say 0.00001).

**Step 1. Get the first entry in the list $AreasToBeSearched$.**

If $AreasToBeSearch$ is empty, Terminate.

Otherwise,

let $Area^{current}$ be the first entry in the list $AreasToBeSearched$.

Let $\bar{C} = Area^{current}(1)$, $C = Area^{current}(2)$, $\bar{\varepsilon} = Area^{current}(3)$, and $\varepsilon = Area^{current}(4)$.

**Step 2. Find groupings corresponding to the four vertices of $Area^{current}$.**

3a: For $f=1...F$, solve $\text{LCP}^F_{SVR}$ with $(C, \varepsilon_e)$ fixed at $(\bar{C}, \bar{\varepsilon})$, $(C, \bar{\varepsilon})$, $(\bar{C}, \varepsilon)$, and $(C, \varepsilon)$ using Algorithm 3.1.
3b: Use Algorithm 3.2 to transform the solution to grouping vectors \( \text{GroupingV}_{uu} \), \( \text{GroupingV}_{ll} \), \( \text{GroupingV}_{ul} \), and \( \text{GroupingV}_{lu} \) respectively.

3c: Check whether \( \text{GroupingV}_{uu} \), \( \text{GroupingV}_{ll} \), \( \text{GroupingV}_{ul} \), and \( \text{GroupingV}_{lu} \) are in the set \( \text{GroupingV}_{\text{Found}} \). If not, add them to the set.

3d: Do only when \( \text{GroupingV}_{uu} = \text{GroupingV}_{ll} = \text{GroupingV}_{ul} = \text{GroupingV}_{lu} \):

Solve the corresponding RLP and let the objective value be \( VUB \). Let
\[ \text{LeastUpperBound} \leftarrow VUB \text{ if } VUB < \text{LeastUpperBound}. \]

\[ \text{AreaRealizedInThe1stStage} = \text{AreaRealizedInThe1stStage} + (\overline{C} - \underline{C}) \times (\overline{\mathbf{e}} - \underline{\mathbf{e}}). \]

Go to Step 1 after eliminating \( \text{Area}_{\text{current}} \) from \( \text{AreasToBeSearched} \).

---

**Step 3.** Find groupings along the four boundaries of \( \text{Area}_{\text{current}} \).

If \( \bar{C} - \underline{C} < \text{insensitive} \): Apply Algorithm 3.4 but skip steps 1, 3, and 4. Go to Step 1 after eliminating \( \text{Area}_{\text{current}} \) from the list of \( \text{AreasToBeSearched} \).

Or if \( \bar{e} - \underline{e} < \text{insensitive} \): Apply Algorithm 3.4 but skip steps 2, 3, and 4. Go to Step 1 after eliminating \( \text{Area}_{\text{current}} \) from \( \text{AreasToBeSearched} \).

Otherwise, apply Algorithm 3.4.

**Step 4.** Check \( \text{countTop} \), \( \text{countLeft} \), \( \text{countBottom} \), and \( \text{countRight} \).

If \( \text{countTop} \leq 2 \) and \( \text{countLeft} \leq 2 \) and \( \text{countBottom} \leq 2 \) and \( \text{countRight} \leq 2 \), throw the \( \text{Area}_{\text{current}} \) to the 2nd stage (Algorithm 3.6) and go to Step 1 after eliminating \( \text{Area}_{\text{current}} \) from the list \( \text{AreasToBeSearched} \).

Otherwise,

go to Step 5.

**Step 5.** Solve for the grouping vector of an arbitrary interior point.

Select an arbitrary interior point \((C^{in}, \varepsilon^{in})\) (say \( C^{in} = \frac{\bar{C} + \underline{C}}{2} \) and \( \varepsilon^{in} = \frac{\bar{\varepsilon} + \underline{\varepsilon}}{2} \)).

Solve \( \text{LCP}^{f}_{\text{SVR}} \) at \((C^{in}, \varepsilon^{in})\) and apply Algorithm 3.2 to get the grouping vector \( \text{GroupingV}^{in} \). Add \( \text{GroupingV}^{in} \) to the set \( \text{GroupingV}_{\text{Found}} \) if it is not one of the elements in the set. Solve the corresponding RLP and let the objective value be \( UB^{in} \) and the optimal \((C_o, \varepsilon_o)\) be \((C_o, \varepsilon_o) \). Let
\[ \text{LeastUpperBound} \leftarrow UB^{in} \text{ if } UB^{in} < \text{LeastUpperBound}. \]

If \((C^{in}, \varepsilon^{in}) = (C_o, \varepsilon_o)\), let \((C_o, \varepsilon_o) = (\bar{C}, \bar{\varepsilon}). \)

**Step 6.** Identifying the point to partition at.
(i) \((C^o, \varepsilon^o)\) is a vertex of \(\text{Area}_{\text{current}}\): 
Find the line function \(\varepsilon_e = L_a C_e + L_b\) (or \(C_e = \hat{C}\)) that passes through \((C^\text{in}, \varepsilon^\text{in})\) and \((C^o, \varepsilon^o)\). Solve (43) and/or (44) to find the invariancy interval. If at least one end of the interval is not located at the vertices, let any of it be \((C^\text{cut}, \varepsilon^\text{cut})\). If both ends of the interval are located at the vertices, throw the \(\text{Area}_{\text{current}}\) to the 2nd stage.

(ii) \((C^o, \varepsilon^o)\) is not a vertex of \(\text{Area}_{\text{current}}\): 
Find the line function \(\varepsilon_e = L_a C_e + L_b\) (or \(C_e = \hat{C}\)) that passes through \((C^\text{in}, \varepsilon^\text{in})\) and \((C^o, \varepsilon^o)\). Solve (43) and/or (44) to find the invariancy interval. If at least one end of the interval is not located at the vertices, let any of it be \((C^\text{cut}, \varepsilon^\text{cut})\).

Step 7. Partition the area into 4 or 2 at \((C^\text{cut}, \varepsilon^\text{cut})\). In Step 6, we found a point \((C^\text{cut}, \varepsilon^\text{cut})\) which is either in the interior or at boundaries, but not at the vertices.

(i) \((C^\text{cut}, \varepsilon^\text{cut})\) is an interior point: 
Search for all groupings along the vertical line \(C_e = C^\text{cut}\) and the horizontal line \(\varepsilon_e = \varepsilon^\text{cut}\). If any groupings are not in the set \(\text{GroupingVFound}\), add them(it) to the set and solve the corresponding \(\text{RLP}\). If the objective value is less than \(\text{LeastUpperBound}\), replace it. Eliminate \(\text{Area}_{\text{current}}\) and add the following 4 areas to the end of the list \(\text{AreasToBeSearched}\). 
\[
\{ \hat{C}, C^\text{cut} + \text{insensitive}, \bar{\varepsilon}, \varepsilon^\text{cut} + \text{insensitive} \}.
\]
\[
\{ \hat{C}, C^\text{cut} + \text{insensitive}, \varepsilon^\text{cut} - \text{insensitive}, \bar{\varepsilon} \}.
\]
\[
\{ C^\text{cut} - \text{insensitive}, \bar{C}, \bar{\varepsilon}, \varepsilon^\text{cut} + \text{insensitive} \}.
\]
\[
\{ C^\text{cut} - \text{insensitive}, \bar{C}, \varepsilon^\text{cut} - \text{insensitive}, \bar{\varepsilon} \}.
\]

(ii) \((C^\text{cut}, \varepsilon^\text{cut})\) is located on \(C_e = \hat{C}\) or \(C_e = \bar{C}\): 
Search for all groupings along the horizontal line \(\varepsilon_e = \varepsilon^\text{cut}\). If any groupings are not in the set \(\text{GroupingVFound}\), add them(it) to the set and solve the corresponding \(\text{RLP}\). If the objective value is less than \(\text{LeastUpperBound}\), replace it.
Eliminate $\text{Area}_{\text{current}}$ and add the following 2 areas to the front of the list $\text{AreasToBeSearched}$.

\begin{itemize}
    \item $\{ \bar{C}, C, \bar{\varepsilon}, \varepsilon^{\text{cut}} + \text{insensitive} \}$.
    \item $\{ C, \bar{C}, \varepsilon^{\text{cut}} - \text{insensitive}, \bar{\varepsilon} \}$.
\end{itemize}

(iii) $(C^{\text{cut}}, \varepsilon^{\text{cut}})$ is located at $\varepsilon_e = \bar{\varepsilon}$ or $\varepsilon_e = \bar{\varepsilon}$:

Search for all groupings along the vertical line $C_e = C^{\text{cut}}$. If any groupings are not in the set $\text{GroupingVFound}$, add them to the set and solve the corresponding $\text{RLP}$. If the objective value is less $\text{LeastUpperBound}$, replace it.

Eliminate $\text{Area}_{\text{current}}$ and add the following 2 areas to the front of the list $\text{AreasToBeSearched}$.

\begin{itemize}
    \item $\{ C, C^{\text{cut}} + \text{insensitive}, \bar{\varepsilon}, \varepsilon \}$.
    \item $\{ C^{\text{cut}} - \text{insensitive}, C, \bar{\varepsilon}, \varepsilon \}$.
\end{itemize}

Go to Step 1 after partitioning. \hfill $\Box$

It is noteworthy that in Step 7, we crop out small margins around the new rectangular subareas resulting from the partitioning. This step in the rectangle search algorithm is imposed to eliminate the trouble of overlapping among the adjacent rectangular areas. By margin-cropping, no single point of $(C_e, \varepsilon_e)$ is repeatedly contained in any two subareas. The cropped-out margin size, $\text{insensitive}$, has to be small enough that no information about the groupings is lost.

We don’t explicitly find the edges of invariancy regions. Instead, we monitor the allocation of the invariancy intervals along the boundaries of each rectangular area. Maintaining the search area as a rectangle is especially convenient in partitioning and eliminating, but the drawback is the possibility of revisiting the same invariancy region many times. One can compare this method with the partitioning technique in [13] and the graph identifying technique in [52].
3.6 The 2nd stage: identify the non-vertical-and-non-horizontal boundary

In the 1st stage of the algorithm (Section 3.5), we only examine, remove, and partition into areas which are rectangular. We view the areas on the base of a rectangle mainly for the convenience of partitioning and the ease of searching along the boundaries as described in Section 3.4. However, the convexity of the invariancy region only implies that a boundary of the region is not a curve but not necessarily a vertical or a horizontal line. Since the termination criterion in the 1st stage is to eliminate rectangular areas of the same grouping, there must be some “leftovers” in the list of areas to be searched (AreasToBeSearched). The 2nd stage of the algorithm is proposed to resolve this issue.

When the number of grouping vectors is not greater than 2 on any side of the boundary of one rectangle, the 2nd stage of the algorithm is activated for this rectangle. The condition which initializes the 2nd stage is stated in Step 5 of the 1st stage (Algorithm 3.5). In essence, the 2nd stage aims to confirm that there is a single straight line bisecting the input rectangular area into two invariancy regions in terms of the grouping vector, thus concluding the realization of the area. There are a total of six possible cases as shown in Figure 9. In the figure, the node denotes where a task of finding the grouping vector and solving the restricted linear program is done. The arrow denotes a task of solving for the invariancy interval given the grouping vector. In Case 1, the bisecting line passes through the top and left-hand-side boundaries; in Case 2, the bisecting line passes through the top and bottom boundaries; in Case 3, the bisecting line passes through the top and right-hand-side boundaries; in Case 4, the bisecting line passes through the left-hand-side and right-hand-side boundaries; in Case 5, the bisecting line passes through the left-hand-side and bottom boundaries; in Case 6, the bisecting line passes through the bottom and right-hand-side boundaries.

To confirm whether a rectangular area fits one of the six cases or not, we first need to check the number of groupings at each side of the boundary, denoted as countTop, countLeft, countBottom, and countRight in Algorithm 3.5. We then compare the vectors of the groupings obtained. The following algorithm describes the details of confirming the six cases and the way to handle exceptions. The
input area for this algorithm has no greater than two groupings at each side of the boundary.

Algorithm 3.5 The 2nd Stage

Step 0: Given an Area := \{\bar{C}, \underline{C}, \bar{\varepsilon}, \underline{\varepsilon}\}. Set parameter insensitive the same as in the 1st stage.

Step 1: Follow Algorithm 3.4 to get countTop, countLeft, countBottom, and countRight. During the process of Algorithm 3.4, additionally maintain four lists of grouping vectors in the order of obtaining:

$$GroupingVList_{\text{Top}}, GroupingVList_{\text{Left}}, GroupingVList_{\text{Bottom}}, \text{ and } GroupingVList_{\text{Right}},$$

whose entries are the grouping vectors identified on the top boundary, left-hand-side boundary, bottom boundary, and right-hand-side boundary respectively. Also maintain four lists of invariancy intervals in the order of obtaining:

$$IList_{\text{Top}}, IList_{\text{Left}}, IList_{\text{Bottom}}, \text{ and } IList_{\text{Right}},$$

whose entries are the \((C_{\min}, C_{\max})\) pairs or \((\varepsilon_{\min}, \varepsilon_{\max})\) pairs corresponding to each grouping vector. We denote the first entry of the lists by “(1)” adjacent to the name of lists, and the second entry of lists, if it exists, by “(2)” adjacent to the name of lists.

Step 2: Find one of the following cases that fits the reality.

(i) countTop = countLeft = 2 and countBottom = countRight = 1:

If \(GroupingVList_{\text{Top}}(1) = GroupingVList_{\text{Left}}(2)\)

and \(GroupingVList_{\text{Top}}(2) = GroupingVList_{\text{Left}}(1)\),

the Area satisfies Case 1. Terminate.

Otherwise,

go to Step 3, Exception 2.

(ii) countTop = countBottom = 2 and countLeft = countRight = 1:

If \(GroupingVList_{\text{Top}}(1) = GroupingVList_{\text{Bottom}}(1)\)
and \( \text{GroupingVList}^{\text{Top}}(2) = \text{GroupingVList}^{\text{Bottom}}(2) \),
the Area satisfies Case 2. Terminate.

Otherwise,

go to Step 3, Exception 2.

(iii) \( \text{countTop} = \text{countRight} = 2 \) and \( \text{countLeft} = \text{countBottom} = 1 \):

If \( \text{GroupingVList}^{\text{Top}}(1) = \text{GroupingVList}^{\text{Right}}(1) \)
and \( \text{GroupingVList}^{\text{Top}}(2) = \text{GroupingVList}^{\text{Right}}(2) \),
the Area satisfies Case 3. Terminate.

Otherwise,

go to Step 3, Exception 2.

(iv) \( \text{countTop} = \text{countBottom} = 1 \) and \( \text{countLeft} = \text{countRight} = 2 \):

If \( \text{GroupingVList}^{\text{Left}}(1) = \text{GroupingVList}^{\text{Right}}(1) \)
and \( \text{GroupingVList}^{\text{Left}}(2) = \text{GroupingVList}^{\text{Right}}(2) \),
the Area satisfies Case 4. Terminate.

Otherwise,

go to Step 3, Exception 2.

(v) \( \text{countTop} = \text{countRight} = 1 \) and \( \text{countLeft} = \text{countBottom} = 2 \):

If \( \text{GroupingVList}^{\text{Left}}(1) = \text{GroupingVList}^{\text{Bottom}}(1) \)
and \( \text{GroupingVList}^{\text{Left}}(2) = \text{GroupingVList}^{\text{Bottom}}(2) \),
the Area satisfies Case 5. Terminate.

Otherwise,

go to Step 3, Exception 2.

(vi) \( \text{countTop} = \text{countLeft} = 1 \) and \( \text{countBottom} = \text{countRight} = 2 \):

If \( \text{GroupingVList}^{\text{Bottom}}(1) = \text{GroupingVList}^{\text{Right}}(2) \)
and \( \text{GroupingVList}^{\text{Bottom}}(2) = \text{GroupingVList}^{\text{Right}}(1) \),
the Area satisfies Case 6. Terminate.

Otherwise

go to Step 3, Exception 2.

(vii) None of the above items fit: Go to Step 3, Exception 1.

Step 3.

Exception 1: more than two groupings.
Partition at the end of the invariancy interval and throw the margin-cropped areas back to the 1st stage.

(i) If \(\text{countTop} \geq 2\), partition the Area vertically at the lower end \((C_{\text{min}})\) of the interval \(IList^{Top}(i)\) and the upper end \((C_{\text{max}})\) of the interval \(IList^{Top}(i+1)\), \(\forall i = 1...\text{countTop} - 1\). A total number of \(\text{countTop}\) areas are thrown back to the 1st stage. By “throwing back to the 1st stage,” we mean to add the following two areas in the list \(\text{AreaToBeSearched}\) and do Steps 1-7 of the 1st stage. For example, when \(\text{countTop} = 2\), the resulting two cropped areas are

\[
\{ \bar{C} - \text{insensitive}, C_{\text{min}} + \text{insensitive}, \bar{\varepsilon} - \text{insensitive}, \varepsilon + \text{insensitive} \}
\]

\[
\{ C_{\text{max}} - \text{insensitive}, \bar{C} + \text{insensitive}, \bar{\varepsilon} - \text{insensitive}, \varepsilon - \text{insensitive} \}
\]

(ii) If the above item doesn’t hold, and if \(\text{countLeft} \geq 2\), partition the Area horizontally at the lower end \((\varepsilon_{\text{min}})\) of the interval \(IList^{Left}(i)\) and the upper end \((\varepsilon_{\text{max}})\) of the interval \(IList^{Left}(i+1)\), \(\forall i = 1...\text{countLeft} - 1\). A total number of \(\text{countLeft}\) areas are thrown back to the 1st stage. For example, when \(\text{countLeft} = 2\), the resulting two cropped areas are

\[
\{ \bar{C} - \text{insensitive}, C_{\text{min}} + \text{insensitive}, \bar{\varepsilon} - \text{insensitive}, \varepsilon_{\text{min}} + \text{insensitive} \}
\]

\[
\{ \bar{C} - \text{insensitive}, C_{\text{min}} + \text{insensitive}, \bar{\varepsilon} - \text{insensitive}, \varepsilon_{\text{min}} + \text{insensitive} \}
\]

(iii) If the above items don’t hold, and if \(\text{countBottom} \geq 2\), partition the Area vertically at the lower end \((C_{\text{min}})\) of the interval \(IList^{Bottom}(1)\) and the upper end \((C_{\text{max}})\) of the interval \(IList^{Bottom}(2)\), \(\forall i = 1...\text{countBottom} - 1\). A total number of \(\text{countBottom}\) areas are thrown back to the 1st stage. For example, when \(\text{countBottom} = 2\), the resulting two cropped areas are

\[
\{ \bar{C} - \text{insensitive}, C_{\text{min}} + \text{insensitive}, \bar{\varepsilon} - \text{insensitive}, \varepsilon + \text{insensitive} \}
\]

\[
\{ C_{\text{max}} - \text{insensitive}, \bar{C} + \text{insensitive}, \bar{\varepsilon} - \text{insensitive}, \varepsilon + \text{insensitive} \}
\]

(iv) If the above items don’t hold, and if \(\text{countRight} \geq 2\), partition the Area horizontally at the lower end \((\varepsilon_{\text{min}})\) of the interval \(IList^{Right}(i)\) and the upper end \((\varepsilon_{\text{max}})\) of the interval \(IList^{Right}(i+1)\), \(\forall i = 1...\text{countRight} - 1\). Throw the resulting two areas back to the 1st stage. A total number of \(\text{countRight}\) areas are thrown back to the 1st stage. For example, when \(\text{countRight} = 2\), the resulting two cropped areas are

\[
\{ \bar{C} - \text{insensitive}, C_{\text{min}} + \text{insensitive}, \bar{\varepsilon} - \text{insensitive}, \varepsilon_{\text{min}} + \text{insensitive} \}
\]
\{ \bar{C} - \text{insensitive}, \underline{C} + \text{insensitive}, \varepsilon_{\max} - \text{insensitive}, \varepsilon + \text{insensitive} \}.

**Exception 2: unknown situation.**

Partition at a random interior point and throw all areas back to the 1st stage. Let the randomly picked interior point be \((C^r, \varepsilon^r)\). Partition the Area into the following four areas and throw them back to the 1st stage.

\{ \bar{C} - \text{insensitive}, C^r + \text{insensitive}, \bar{\varepsilon} - \text{insensitive}, \varepsilon^r + \text{insensitive} \}.

\{ \bar{C} - \text{insensitive}, C^r + \text{insensitive}, \varepsilon^r - \text{insensitive}, \bar{\varepsilon} + \text{insensitive} \}.

\{ C^r - \text{insensitive}, \underline{C} + \text{insensitive}, \bar{\varepsilon} - \text{insensitive}, \varepsilon^r + \text{insensitive} \}.

\{ C^r - \text{insensitive}, \underline{C} + \text{insensitive}, \varepsilon^r - \text{insensitive}, \bar{\varepsilon} + \text{insensitive} \}.

An Example of Exception 1 in Step 3 is shown in Figure 10. Such an area activates the 2nd stage at Step 4 of the 1st stage, but will be thrown to Exception 1 at item 7 in Step 2 of the 2nd stage. We can see there are in fact more than two groupings, thus more than two non-vertical-and-non-horizontal boundaries inside the area. Exception 2 captures the unknown cases resulting from either the degeneracy of the complementarity or arithmetic errors. For the areas thrown to Exception 1, a conclusion about the realization of groupings can be made after further partitioning; but the areas thrown to Exception 2 in the worst case can only be partitioned and cropped until they are too thin to have any unrevealed groupings.

An overview of running the whole algorithm on the \((C_e, \varepsilon_e)\)-plane is shown in Figure 11. \((C_e, \varepsilon_e)\) is denoted by \((C, \varepsilon)\) in the figure for simplicity. To make the illustration simple, we omit every subscript, superscript, typeface, and symbol in the statements of the figure. The node on the plane denotes a task of solving for the grouping vector and the restricted linear program; the number marked next to a node denotes the objective value of the restricted linear program if this value has not been obtained in the early iterations; the arrow denotes a task of solving for the invariancy interval given the grouping vector at the starting point of the arrow.

Some details in Algorithm 3.5 and Algorithm 3.6 are not stated in this overview.
Figure 9: The six cases in which the rectangular area is bisected by a straight line into two invariancy regions A1 and A2 on the $(C_e, \varepsilon_e)$-plane.
CHAPTER 3.

Figure 10: Example for the Exception 1 in the Step 3 of the 2nd stage. There are three invariancy regions, A1, A2 and A3.

Figure 11: \((C_e, \varepsilon_e)\)-rectangle search algorithm. The 1st and the 2nd stages are both illustrated.
CHAPTER 3.

Iteration 5: Start working on the upper rectangle. The search of any new rectangle is initialized by checking the number of invariancy intervals along its boundaries. If it’s less than 2 at any side, confirm whether there are only 2 groupings in the rectangle.

Iteration 6: Yes, we found there are only two groupings. We have examined every invariancy region in the upper rectangle. Remove it from the queue.

Iteration 7: For the lower rectangle, do as iterations 1-3. The optimal \((\hat{C}, \hat{c})\) is at one of the corners. If it happens, connect the interior point (chosen in iteration 3) and the corner point. Find the invariancy interval on this line.

Iteration 8: Cut the rectangle at one of the end points of the invariancy interval.

Iteration 9: The top-right rectangle has 2 groupings and is eliminated. The 4 corners of the bottom-right rectangle have the same grouping, so does the top-left rectangle. They are both eliminated.

Iteration 10: Repeat the iterations 1-9 on all the remaining rectangles in the queue until all regions are eliminated. When the algorithm stops, the smallest objective values in our record is the global optimum.

Figure 11: (continued)
4 Integer Program with the Big-Numbers Tightening Procedure

In this section, we propose a procedure for finding and tightening the valid big numbers $\theta_1 j$, $\theta_2 j$, $\theta_3 j$, $\theta_4 j$, $\theta_5 j$, and $\theta_6 j$ employed in model (28). When the big numbers $\theta_i j, i = 1, \ldots, 6$ are tightened, the direct effect is that the feasible region defined by (28) shrinks. In the meantime, a valid lower bound of the testing data residuals is also lifted. We use this technique to form a binary-integer program that can be solved using any IP commercial solvers.

Among the big numbers $\theta_i j, \forall i = 1, \ldots, 6$, the $\theta_1 j$, $\theta_3 j$ and $\theta_5 j$ are not related with $e_s j$ and are up-bounded by $C$. The following optimization programs, with choices of the objective functions $\alpha_j$, $\beta_j$, and $C e^{-\alpha_j - \beta_j}$, solve for the valid $\theta_1 j$, $\theta_3 j$, and $\theta_5 j$ respectively:

\[
\begin{align*}
\max_{C, e, \alpha_1, \alpha_2, \ldots, \alpha_F, \beta_1, \beta_2, \ldots, \beta_F} \quad & \alpha_j \\beta_j \\ C e - \alpha_j - \beta_j \\
\text{subject to} \quad & 0 \leq C \leq C e \leq \overline{C}, \\
& 0 \leq e \leq \epsilon e \leq \overline{\epsilon} e, \\
& (x_i^d)^T w^1_s + b^1_s - y_d \leq p_i, \quad \forall i = n + 1, \ldots, n + m_1, \\
& -(x_i^d)^T w^1_s - b^1_s + y_d \leq p_i, \quad \forall i = n + 1, \ldots, n + m_1, \\
& (x_i^d)^T w^2_s + b^2_s - y_d \leq p_i, \quad \forall i = n + m_1 + 1, \ldots, n + m_1 + m_2, \\
& -(x_i^d)^T w^2_s - b^2_s + y_d \leq p_i, \quad \forall i = n + m_1 + 1, \ldots, n + m_1 + m_2, \\
& \vdots \\
& (x_i^d)^T w^F_s + b^F_s - y_d \leq p_i, \quad \forall i = \text{front}_{ts}^F, \ldots, \text{end}_{ts}^F, \\
& -(x_i^d)^T w^F_s - b^F_s + y_d \leq p_i, \quad \forall i = \text{front}_{ts}^F, \ldots, \text{end}_{ts}^F, \\
& \text{objLB} \leq \sum_{f=1}^{\text{end}_{ts}^F} \sum_{i=\text{front}_{ts}^f}^{\text{end}_{ts}^f} p_i \leq \text{objUB}, \\
& \sum_{j=\text{front}_{ts}^f}^{\text{end}_{ts}^f} \alpha_j - \sum_{j=\text{front}_{ts}^f}^{\text{end}_{ts}^f} \beta_j = 0, \quad \forall f = 1, \ldots, F,
\end{align*}
\]
and

\[
\begin{align*}
&\left( \sum_{j=\text{front}^f}^{\text{end}^f} e_s j \right) C_e + \left( \sum_{j=\text{front}^f}^{\text{end}^f} (\alpha_j + \beta_j) \right) \varepsilon_e \\
&+ \left( \sum_{j=\text{front}^f}^{\text{end}^f} (\beta_j - \alpha_j) (x_d^j) \right)^T \left( \sum_{i=\text{front}^f}^{\text{end}^f} (\beta_i - \alpha_i) (x_d^i) \right) \\
&+ \sum_{j=\text{front}^f}^{\text{end}^f} (-\alpha_j + \beta_j) b_s + \sum_{j=\text{front}^f}^{\text{end}^f} (\alpha_j - \beta_j) y_j = 0,
\end{align*}
\]

(47)

where the last \( F \) equalities are the corresponding \( F \)-fold sums of zeros obtained from aggregating the products of the two sides to the \( \bot \) sign. The advantage of aggregating the complementarities is on the resulting tractable quadratic terms \( \left( \sum_{j=\text{front}^f}^{\text{end}^f} (\beta_j - \alpha_j) (x_d^j) \right)^T \left( \sum_{i=\text{front}^f}^{\text{end}^f} (\beta_i - \alpha_i) (x_d^i) \right) \). The \( \sum_{j=\text{front}^f}^{\text{end}^f} (-\alpha_j + \beta_j) b_s \) terms in (47) can be canceled out because \( \sum_{j=\text{front}^f}^{\text{end}^f} (\alpha_j - \beta_j) = 0, \forall f = 1, \ldots, F \).

The remaining two nonconvex terms \( \left( \sum_{j=\text{front}^f}^{\text{end}^f} e_s j \right) C_e \) and \( \left( \sum_{j=\text{front}^f}^{\text{end}^f} (\alpha_j + \beta_j) \right) \varepsilon_e \), which pose a computational challenge for existing solvers, can be approximated by the Tyler-expansion type of linear constraints. This quadratically constrained relaxation of (46) is as follows:

\[
\begin{align*}
\mathbf{x}^h + \begin{bmatrix} C_e \\ \varepsilon_e \end{bmatrix} &= 0, \\
\mathbf{z}^f + \begin{bmatrix} \sum_{j=\text{front}^f}^{\text{end}^f} e_s j \\ \sum_{j=\text{front}^f}^{\text{end}^f} (\alpha_j + \beta_j) \end{bmatrix} &= 0, \forall f = 1, \ldots, F,
\end{align*}
\]

(48)

\[
\begin{align*}
0 &\leq \mathbf{x}^h \circ \mathbf{z}^f + \mathbf{z}^f \circ \mathbf{x}^h - \mathbf{x}^h \circ \mathbf{z}^f + \mathbf{v}^f, \forall f = 1, \ldots, F, \\
0 &\leq \mathbf{x}^h \circ \mathbf{z}^f + \mathbf{z}^f \circ \mathbf{x}^h - \mathbf{x}^h \circ \mathbf{z}^f + \mathbf{v}^f, \forall f = 1, \ldots, F,
\end{align*}
\]

(48)

\[
\begin{align*}
0 &\leq \mathbf{x}^h \circ \mathbf{z}^f + \mathbf{z}^f \circ \mathbf{x}^h - \mathbf{x}^h \circ \mathbf{z}^f + \mathbf{v}^f \leq 0, \forall f = 1, \ldots, F, \\
\mathbf{x}^h \circ \mathbf{z}^f + \mathbf{z}^f \circ \mathbf{x}^h - \mathbf{x}^h \circ \mathbf{z}^f + \mathbf{v}^f \leq 0, \forall f = 1, \ldots, F,
\end{align*}
\]

(48)

\[
\begin{align*}
v^f(1) + v^f(2) + \left( \sum_{j=\text{front}^f}^{\text{end}^f} (\beta_j - \alpha_j) (x_d^j) \right)^T \left( \sum_{i=\text{front}^f}^{\text{end}^f} (\beta_i - \alpha_i) (x_d^i) \right) \\
+ \sum_{j=\text{front}^f}^{\text{end}^f} (\alpha_j - \beta_j) y_j \leq 0, \forall f = 1, \ldots, F,
\end{align*}
\]

(48)
where \( \mathbf{x}_h \in \mathbb{R}^2 \), \( \mathbf{z}_h^f \in \mathbb{R}^2 \) \( \forall f \) and \( \mathbf{v}^f \in \mathbb{R}^2 \) \( \forall f \). \( \mathbf{v}^f(1) \) and \( \mathbf{v}^f(2) \) are the first and second entries of \( \mathbf{v}^f \) respectively.

\[
\begin{align*}
\mathbf{x}_h &= [-C; -\varepsilon], \\
\mathbf{z}_h &= [-C; -\varepsilon], \\
\mathbf{z}_h^f &= [0; 0], \text{ and} \\
\mathbf{z}_h^f &= \begin{bmatrix}
-\left( \text{upper bound of } \sum_{j=\text{front}^f}^{\text{end}^f} c_{sj}^f \right) \\
-\left( \text{upper bound of } \sum_{j=\text{front}^f}^{\text{end}^f} (\alpha_j^f + \beta_j^f) \right)
\end{bmatrix}.
\end{align*}
\]

The second entry of \( \mathbf{z}_h^f \) is a direct result of solving the model (46) subject to (48). A refinement of \( \mathbf{z}_h^f \) thus can be made if the objective values of (46) are improved. This is the key property in our big-numbers tightening procedure.

On the other hand, the valid values of \( \theta_{2j}, \theta_{4j}, \text{and } \theta_{6j} \), are all related to \( e_{sj} \).

By definition,

\[
\begin{align*}
e_{sj} &\geq (\mathbf{x}_d^j)^T \mathbf{w}_s^f + b_s^f - y_{dj} - \varepsilon_e, \\
e_{sj} &\geq -(\mathbf{x}_d^j)^T \mathbf{w}_s^f - b_s^f + y_{dj} - \varepsilon_e, \\
e_{sj} &\geq 0.
\end{align*}
\]

This implies that \( \theta_{6j} \) is the larger objective value of the following two optimization problems:

\[
\begin{align*}
\max_{\text{variables in (46), } \mathbf{x}_h, \mathbf{z}_h^f, \mathbf{v}^f} (\mathbf{x}_d^j)^T \mathbf{w}_s^f + b_s^f - y_{dj} - \varepsilon_e \\
\text{subject to constraints in (46) and (48),}
\end{align*}
\]

and

\[
\begin{align*}
\max_{\text{variables in (46), } \mathbf{x}_h, \mathbf{z}_h^f, \mathbf{v}^f} -(\mathbf{x}_d^j)^T \mathbf{w}_s^f - b_s^f + y_{dj} - \varepsilon_e \\
\text{subject to constraints in (46) and (48).}
\end{align*}
\]

Let the larger objective values among (50) and (51) be \( e_{sj} \), then a valid \( \theta_{2j} \) can be
obtained by solving

\[
\max_{x_h, z_h, v^f} \quad e_{s_j} + e_{v} - (x_d^j)^T w_{s_f} - b_{s_f}^f + y_{d_j}
\]
subject to constraints in (46) and (48). \hfill (52)

Similarly, a valid \( \theta_{4j} \) can be obtained by solving

\[
\max_{x_h, z_h, v^f} \quad e_{s_j} + e_{v} + (x_d^j)^T w_{s_f} + b_{s_f}^f - y_{d_j}
\]
subject to constraints in (46) and (48). \hfill (53)
The big numbers $\theta_{1j}$, $\theta_{2j}$, $\theta_{3j}$, $\theta_{4j}$, $\theta_{5j}$, and $\theta_{6j}$ well define the integer program:

$$
\min_{C_e, \varepsilon, w_s^1, b_s^1, \ldots, w_s^F, b_s^F, p_i, \alpha_j, \beta_j, \varepsilon_s^j} \sum_{f=1}^{F} \sum_{i=\text{front}_1^f}^{\text{end}_1^f} p_i
$$

subject to

$$
0 \leq C_e \leq \bar{C},
$$

$$
0 \leq \varepsilon \leq \varepsilon_e \leq \bar{\varepsilon},
$$

$$(x_d^i)^T w_s^1 + b_s^1 - y_{d_i} \leq p_i, \quad \forall i = n + 1, \ldots, n + m_1,$$

$${- (x_d^i)^T w_s^1} - b_s^1 + y_{d_i} \leq p_i, \quad \forall i = n + 1, \ldots, n + m_1,$$

$$(x_d^i)^T w_s^2 + b_s^2 - y_{d_i} \leq p_i, \quad \forall i = n + m_1 + 1, \ldots, n + m_1 + m_2,$$

$${- (x_d^i)^T w_s^2} - b_s^2 + y_{d_i} \leq p_i, \quad \forall i = n + m_1 + 1, \ldots, n + m_1 + m_2,$$

$$
\vdots
$$

$$(x_d^i)^T w_s^F + b_s^F - y_{d_i} \leq p_i, \quad \forall i = \text{front}_F^f, \ldots, \text{end}_F^f,$$

$${- (x_d^i)^T w_s^F} - b_s^F + y_{d_i} \leq p_i, \quad \forall i = \text{front}_F^f, \ldots, \text{end}_F^f,$$

$$
\forall f = 1, \ldots, F:
$$

$$
\begin{aligned}
& w_s^f - \sum_{j=\text{front}_1^f}^{\text{end}_1^f} (\beta_j - \alpha_j)x_d^j = 0, \\
& \sum_{j=\text{front}_1^f}^{\text{end}_1^f} \alpha_j - \sum_{j=\text{front}_1^f}^{\text{end}_1^f} \beta_j = 0,
\end{aligned}
$$

$$
\forall j = \text{front}_1^f, \ldots, \text{end}_1^f:
$$

$$
\begin{aligned}
0 \leq \alpha_j & \leq \theta_{1j} \cdot z_j, \\
0 \leq \varepsilon_s^j + \varepsilon - (x_d^j)^T w_s^f - b_s^f + y_{d_j} & \leq \theta_{2j} \cdot (1 - z_j), \\
0 \leq \beta_j & \leq \theta_{3j} \cdot z'_j, \\
0 \leq \varepsilon_s^j + \varepsilon + (x_d^j)^T w_s^f + b_s^f - y_{d_j} & \leq \theta_{4j} \cdot (1 - z'_j), \\
0 \leq C_e - \alpha_j - \beta_j & \leq \theta_{5j} \cdot \eta_j, \\
0 \leq e_s^j & \leq \theta_{6j} \cdot (1 - \eta_j).
\end{aligned}
$$

The big-values tightening procedure is as follows:
Algorithm 4.1 Obtaining the modified integer program.

Step 1: Initialization.
Set $\text{obj}LB$ and $\text{obj}UB$ at the exogenous valid lower and upper bound of
$\sum_{f=1}^{F} \sum_{i=\text{front}f}^{\text{end}f} p_i$ respectively.

Step 2: Obtaining initial big numbers by solving the linear program.

2a: Solve (46) for all choices of the objective functions without constraints (48).
Let the optimal objective values of $\alpha_j$, $\beta_j$, and $C_c - \alpha_j - \beta_j$ be $\theta_{1j}$, $\theta_{3j}$, and $\theta_{5j}$ respectively.

2b: Solve (50) and (51) without constraints (48). Let the optimal objective values be $e_{s_j}^{(1)}$ and $e_{s_j}^{(2)}$ respectively. Then $\overline{e_{s_j}} = \max(e_{s_j}^{(1)}, e_{s_j}^{(2)})$.

$\theta_{6j} = \overline{e_{s_j}}$.

2c: Solve (52) and (53) without constraints (48). Let the optimal objective values be $\theta_{2j}$ and $\theta_{4j}$ respectively.

Step 3: Solving for the improved lower bound.
Solve for $\max \sum_{f=1}^{F} \sum_{i=\text{front}f}^{\text{end}f} p_i$ subject to the constraints in (46) and (48), where $zh^f = [-\sum_{j=\text{front}}^{\text{end}f} \theta_{6j} - \sum_{j=\text{front}}^{\text{end}f} (\theta_{1j} + \theta_{3j})]$. Let the objective be $\text{obj}LB^c$.

If $\text{obj}LB^c > \text{obj}LB$, let $\text{obj}LB \leftarrow \text{obj}LB^c$ and go to Step 3.
Otherwise, go to Step 4.

Step 4: Solving for the tightened big numbers.

4a: Solve (46) for all choices of the objective functions with constraints (48).
Let the optimal objective values of $\alpha_j$, $\beta_j$, and $C_c - \alpha_j - \beta_j$ be $\tilde{\theta}_{1j}$, $\tilde{\theta}_{3j}$, and $\tilde{\theta}_{5j}$ respectively.

4b: Solve (50) and (51) with constraints (48). Let the optimal objective values be $e_{s_j}^{(1)}$ and $e_{s_j}^{(2)}$ respectively. Then $\overline{e_{s_j}} = \max(e_{s_j}^{(1)}, e_{s_j}^{(2)})$, $\tilde{\theta}_{6j} = \overline{e_{s_j}}$.

4c: Solve (52) and (53) with constraints (48). Let the optimal objective values be $\tilde{\theta}_{2j}$ and $\tilde{\theta}_{4j}$ respectively.

If any of the last entries of $\tilde{\theta}_{ij}$, $\forall i = 1, \ldots, 6$, $\tilde{\theta}_{ij=n}$, are smaller than $\theta_{ij=n}$, let $\theta_{ij} \leftarrow \tilde{\theta}_{ij}$, $\forall i = 1, \ldots, 6$. Update $zh^f$ and repeat Step 4.
Otherwise, go to Step 5.

Step 5: Solve the integer program (54).
For instances that are solvable with arbitrary valid big numbers, the usage of tightened big numbers significantly improves the running time. The tightened big numbers also allow more instances to be solved. However, the tightening level which can be reached via this procedure is limited due to the effect of aggregating the complementarity constraints. The approximation to the aggregated complementarities becomes looser as the number of complementarity constraints increase. When the size of training data is above some threshold, the tightened values $\theta^*_1, \theta^*_2, \theta^*_3, \theta^*_4, \theta^*_5, \text{and } \theta^*_6$ resulting from Algorithm 4 are not small enough, and the IP (54) cannot be solved by the solver. In this situation, we lack good big-numbers and a good lower bound on the objective function of the outer problem.

Now suppose a global solution set is known: $(\alpha^*_j, \beta^*_j, e^*_s, \varepsilon^*_e, C^*_e, w^*_s f, b^*_s f)$. We say the integer program defined by the big numbers $\theta^*_1, \theta^*_2, \theta^*_3, \theta^*_4, \theta^*_5, \text{and } \theta^*_6$ in the following

$$
\begin{align*}
\theta^*_1 &= \theta^*_2 = \max(\alpha^*_j, e^*_s + \varepsilon^*_e - (x^d_j)T w^*_sf - b^*_sf + y_{dj}), \\
\theta^*_3 &= \theta^*_4 = \max(\beta^*_j, e^*_s + \varepsilon^*_e + (x^d_j)T w^*_sf + b^*_sf - y_{dj}), \\
\theta^*_5 &= \theta^*_6 = \max(e^*_sj, C^*_e - \alpha^*_j - \beta^*_j),
\end{align*}$$

contains at least one of the global optimal solutions. Although portions of the feasible regions are cut off, the values of (55) provide a benchmark for the tightness of the big numbers $\theta$.

Table 7 gives examples of the averaged tightened values $\theta_1, \theta_2, \theta_3, \theta_4, \theta_5,$ and $\theta_6$ that are obtained from Algorithm 4. The instances in the examples have 30 features and $C_e \in [1, 10]$, $\varepsilon_e \in [0.1, 0.5]$. Compared with the benchmark values of $\theta^*_1, \theta^*_2, \text{and } \theta^*_5$, these big-numbers are still very large even after tightening, especially when the numbers of training data points exceed 30.

5 Numerical Experiments

In this section we provide the numerical experiments for solving the cross-validated SVM regression parameters selection. The following themes are covered:

- Data sources: synthetic data where $(x^d, y_{d})$ are random values in $[0, 10]$ and real-world data where $x^d$ are the indicators and $y_{d}$ represents the diseases.
Table 7: Examples of the averaged tightened values $\theta_1$, $\theta_2$, $\theta_3$, $\theta_4$, $\theta_5$, and $\theta_6$. Each data point has 30 features.

<table>
<thead>
<tr>
<th># training data</th>
<th>$\theta_1$</th>
<th>$\theta_2$</th>
<th>$\theta_3$</th>
<th>$\theta_4$</th>
<th>$\theta_5$</th>
<th>$\theta_6$</th>
<th>$\theta_1^* = \theta_3^*$</th>
<th>$\theta_5^* = \theta_6^*$</th>
<th>$\theta_1^* = \theta_6^*$</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>0.126235</td>
<td>1.081324</td>
<td>0.126234</td>
<td>1.081114</td>
<td>9.989524</td>
<td>0.04928</td>
<td>0.086908</td>
<td>0.126908</td>
<td>0.986183</td>
</tr>
<tr>
<td>10</td>
<td>0.161111</td>
<td>1.113009</td>
<td>0.161113</td>
<td>1.112452</td>
<td>9.996173</td>
<td>0.063365</td>
<td>0.275497</td>
<td>0.275497</td>
<td>0.991417</td>
</tr>
<tr>
<td>15</td>
<td>0.239673</td>
<td>1.176334</td>
<td>0.239672</td>
<td>1.177155</td>
<td>9.996574</td>
<td>0.094693</td>
<td>0.468006</td>
<td>0.539704</td>
<td>0.99229</td>
</tr>
<tr>
<td>20</td>
<td>0.515332</td>
<td>1.383488</td>
<td>0.515283</td>
<td>1.386732</td>
<td>9.995492</td>
<td>0.207011</td>
<td>0.341058</td>
<td>0.536199</td>
<td>0.986745</td>
</tr>
<tr>
<td>25</td>
<td>1.333268</td>
<td>2.023181</td>
<td>1.333353</td>
<td>2.018388</td>
<td>9.992558</td>
<td>0.540339</td>
<td>0.491325</td>
<td>0.974053</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>5.066198</td>
<td>9.066363</td>
<td>5.063483</td>
<td>9.061301</td>
<td>9.957015</td>
<td>4.378865</td>
<td>0.561134</td>
<td>0.625488</td>
<td>0.813378</td>
</tr>
<tr>
<td>35</td>
<td>8.862497</td>
<td>88.7688</td>
<td>8.795676</td>
<td>88.94464</td>
<td>9.998</td>
<td>45.24857</td>
<td>global solution not known</td>
<td></td>
<td></td>
</tr>
<tr>
<td>40</td>
<td>9.910701</td>
<td>127.2467</td>
<td>9.886906</td>
<td>127.2841</td>
<td>9.998719</td>
<td>64.8866</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Number of folds for the training data: 1 fold, 2 folds, 3 folds, or 5 folds.
- Number of folds for the testing data: the same as the number of folds for the training data or 1 single fold.
- Number of features: 5 features or 30 features.
- Algorithms for the global optimal solution: the $(C_e, \varepsilon_e)$-rectangle search algorithm or the improved inter program solved by CPLEX.
- Local optimal solution or a valid upper bound: an intermediate solution of the $(C_e, \varepsilon_e)$-rectangle search algorithm or a solution obtained from KNITRO, an sequential quadratic programming (SQP) nonlinear programming (NLP) solver that can be used for mathematic programs with equilibrium constraints (MPEC).

The experiments are run on a machine with Intel i7-2600k CPU, 16 GB memory, and OS windows 7, except those that use KNITRO on NEOS. The $(C_e, \varepsilon_e)$-rectangle search algorithm described in Section 3 and the improved inter program described in Section 4 are implemented in C++.

In the implementation of the $(C_e, \varepsilon_e)$-rectangle search algorithm, we have an “expediting mode,” which expedites the convergence imposed on Algorithm 3.4:

\[
\text{In processing Algorithm 3.4, when any value of the countTop, countLeft, countBottom and countRight exceeds 2, terminate the process.}
\]  

\[(56)\]
The function of this trick is to avoid spending too much time on searching along the boundaries to find all the groupings for every rectangles. Under this trick, the complete Algorithm 3.4 is done only when the rectangles have no more than 2 groupings at each side of boundaries, that is, when the rectangles are eligible to enter the 2nd stage (Algorithm 3.6).

5.1 Synthetic data

In Tables 8, 9 and 10, the results of the parameter selection problem for synthetic data with one, two and three folds of training data are shown respectively. Here the number of folds for the testing data is the same as that for the training data. The first column is a self-explanatory name for each instance. For example, instance 5_5_30_1 means that there are 5 data points in each fold of the testing data, 5 data points in each fold of the training data, 30 features for each data point, and a total of 1 fold of training data. The second column is the number of complementarity constraints. The 3rd to 6th columns record the solution obtained at convergence (if the algorithm converges) or at termination (if the algorithm is interrupted due to an error or after a long time waiting). The possible entries in the 3rd column are Yes, fail^1, No^2, or No^3. “Yes” denotes the case where the global optimum is obtained, and the $C_e$ and $\varepsilon_e$ values in the 5th and 6th columns are the optimal solution. “fail^1” denotes the case where errors occur either in the 1st or 2nd stage of the algorithm so the running process is forced to stop. “No^2” denotes the case where the instance cannot be solved within a limit of time. In this case, the $C_e$ and $\varepsilon_e$ recorded in the 5th and 6th columns are solutions for which the least valid upper bound is obtained. The time limit imposed on these instances is 8000 seconds for each stage. Note that we impose the time limit on some instances only when we know the instance is unlikely to be solved in a decent running time, or we have confronted a long waiting time on a simpler or equal size of instance. “No^3” is similar to “No^2” except that the running process was interrupted manually before converging. The 7th column records the time in seconds to obtain the global optimum. If the algorithm doesn’t converge, we mark it by “n/a”. The 8th column records the number of groupings found during the search. The 9th column records the number of different objective values ever obtained from solving RLP or RQCP. If the instance is solved to global
optimum, the values in the 8th and 9th column are the ultimate realizations of possible groupings and objective values respectively; otherwise, they only show the intermediate understandings. The number of objective values of RLP or RQCP is at most the number of possible groupings. The values in the 9th column can be less than those in the 8th column because it is possible for two different groupings to lead to the same objective. The 9th column records the leastUpperBound obtained at the completion of the 1st stage of the algorithm before proceeding to the 2nd stage. This value can be a tight valid upper bound, but we can only confirm it at the completion of both stages. The 10th column records the number of rectangular areas being processed in the 1st stage. The 11th column is the time spent in the 1st stage. The 12th column is the proportion of areas which were realized during the 1st stage. The value in the 12th column is the ratio of AreaRealizedInThe1stStage divided by the initial box-constrained area of \((C_e, \varepsilon_e)\). The 13th column records the number of rectangular areas being processed in the 2nd stage of the algorithm.

For the instances with 5 features shown in Table 8, the \((C_e, \varepsilon_e)\)-rectangle search algorithm can solve the 1-fold-5-features problems with up to 75 training and 75 testing data points to global optimum. The unsuccessful runs for instances with more than 75 data points are due to the failure in solving the restricted linear program RLP and the restricted quadratic constrained program RQCP\(^7\) at the vertices of some rectangular areas. This is indeed a problematic issue for Algorithm 3.5 and 3.6 when there is any unsolved RLP (and RQCP) during the process of searching and eliminating. The constraints set (38) of RLP is used in models (43) and (44) obtaining the invariancy intervals. Without successfully solving RLP and RQCP, an invariancy interval cannot be identified, followed by an inefficient partitioning at a non-boundary point.

Shown in Table 9 and 10, the \((C_e, \varepsilon_e)\)-rectangle search algorithm solves the 2-fold- and 3-fold-5-features problems to global optimum without troubles. From the 7th and 8th columns of the tables, we can see that the time spent to get convergence is about linear to the number of groupings being discovered. The number of feasible groupings that an instance can carry is determined by the two factors: number of training data points and number of features.

For instances with 30 features, as shown in Tables 8, 9 and 10, the search

\(^7\)Recall that we solve a RQCP when RLP or LCP fail to be solved.
algorithm can only solve up to 30 data points for each fold with 1, 2, or 3 folds of training data, except a single instance. The intermediate number of groupings for these unsolved instances is already very large compared with those instances of the same size of training data but fewer features.

We notice that the relationship between number of features and the number of data points affect the difficulty of solving an instance by the \((C, \varepsilon)-rectangle\) search algorithm. If the number of training data points is less than the number of features, the problem is easier. This observation can be related to the geometric analysis where we aim to identify a \(k-1\)-dimensional hyperplane in a \(k\)-dimensional space to fit \(n\) training data points to a least square linear regression. Consider the fact that \(k\) properly located data points determine a hyperplane in \(k\)-dimension. If \(n < k\), there are more than one hyperplane containing the \(n\) data points on it. If \(n = k\), given non-collinearity and other conditions on the relationship of points, there is only one hyperplane containing the \(n\) data points on it. If \(n > k\), a hyperplane is determined under the rule of least square, yet there will be at least \(n - k\) points outside the hyperplane. Therefore, when \(n < k\), there is more freedom in determining a hyperplane with no residuals.

To analyze the difficulty level of an instance for the \((C, \varepsilon)-rectangle\) search algorithm, we propose to separate the instances into four quadrants- \(I, II, III,\) and

![Figure 12: The four quadrants of difficulty that are separated by the number of training data points each fold and the features of a data point of an instance.](image-url)
to categorize the instances into four levels of difficulty: the easiest, easy, hard, the hardest, as shown in Figure 12. The separating horizontal and vertical axes represent the number of training data points and number of features respectively, and the origin denotes the instance with equal number of the training data points and features. Within each quadrant, the running time of the instances is locally proportional to the number of training data points and folds. From Figure 13, for instances belonging to the same difficulty level, the running time is about linear to number of training data points and number of folds.

The global optimal results shown in Tables 8, 9, and 10 contain the successful runs for the easy, the easiest, and hard instances, but the hardest instances remain unsolved using the \((C, \varepsilon, \varepsilon)-rectangle\) search algorithm.

### 5.2 Real-world data

Besides the synthetic data, we run parameter selection for the real-world data of chemoinformatics, which has been used in the work of [120, 72, 33]. The original usage of these data sets is for building the Quantitative Structure Activity Relationship (QSAR) models, but we borrow the data only to test the algorithm without discussing the meaning of the real-world application. A profile of these data sets is shown in Table 11.

The setting in [120, 72] is to divide the training data into 5 folds and keep the testing data as 1 fold. For example, the 100 training data points for data set “aquasol” is divided into 5 folds, that is, each fold of training data contains 20 data points and the single fold of testing data contains all 97 testing data points. We follow the same method to create the folds of data. In addition, compared to the parameter selection studies in [120, 72] on the same data set, our experiments have some modifications. In [72], a lower and upper bound is imposed on the normal vector \(w_d\) of the hyperplane in SVM regression problems, thus making them slightly different from problem (3). The approaches employed in [72] don’t aim at confirming the global optimality. In [120], a branch-and-cut-based algorithm was proposed for the global optimality, yet all experiments with real-world data were terminated in 7200 seconds at a local optimal solution. In both [120] and [72] the outer-level objective function takes the average of the regression error, while we take
Figure 13: Running time of the instances belonging in four quadrants separately.
the sum. This small change doesn’t affect the optimal values of $C_e$ and $\varepsilon_e$ when the number of training data points in each fold is equal, which is our case, but it affects the outer objective values being presented to readers. The analysis in these two previous works focuses on performance, which is viewed by the Mean Average Deviation (MAD) and Mean Squared Error (MSE) of the trained parameters on a hold-out set of data. We don’t use the hold-out set because the performance of the parameters on a hold-set is really not guaranteed. (See the debates about the meaning of a “best parameter” in Section 2.2.) Instead, we focus on the global optimality itself. Under these modifications, we are the first to obtain a certificate of global optimality for the problem sets “cancer,” “BBB” and “CCK,” while the global optimality for the set “aquasol” remains unrealized. The results of solving real-world instances using the $(C_e, \varepsilon_e)$-rectangle search are shown in Table 12.

From Table 12, the runs on data sets “cancer,” “BBB” and “CCK” are successful both in global optimality and in convergent time. For the data set “aquasol,” we can see that the algorithm cannot converge after 1,052,390 seconds, and the number of groupings identified for “aquasol_1” is already really large. We then acknowledge that the set “aquasol” is a challenge to our rectangle search algorithm. Runs on the remaining instances “aquasol_2 - aquasol_10” are forced to stop at around 8000 seconds. We reasonably believe that the number of groupings identified at 8000 seconds is much smaller than the number of all possible groupings.

Numerical results show that the $(C_e, \varepsilon_e)$-rectangle search algorithm is less sensitive to the increase of the number of folds than to the number of features. For the data sets “cancer,” “BBB” and “CCK,” dividing the training data set into 5 folds actually makes the number of training data points each fold drop far below the number of features. The difficulty of these instances is categorized in the $II$ quadrant- easy. Although we need to solve more $\mathcal{LCP}_{SVR}$, it seems to be beneficial for the $(C_e, \varepsilon_e)$-rectangle search to divide the data set into many folds such that the problem of each fold is less difficult.

### 5.3 Performance analysis between methods

In this subsection, we compare the global and local optimal solutions with the convergent efficiency provided by various approaches. Among these algorithms, the
(\(C_e, \varepsilon_e\))-rectangle search algorithm and the improved integer program with tightened big numbers \(\theta\) solve the instances to the global optimum at convergence. We implement Algorithm 4 in C++ and use CPLEX 12.2 to solve the program \((54)\). The initial \(\text{obj}LB\) in the Step 1 can be set at 0, or at the least sum of residuals of fitting the testing data to the absolute regression hyperplane. The initial \(\text{obj}UB\) is set at the least upper bound obtained in the 1st stage of the \((C_e, \varepsilon_e)\)-rectangle search algorithm. Besides the two global optimal algorithms, the instances were run on NEOS machine in order to use KNITRO, the mathematic program with equilibrium constraints solver. In this way, we obtained a quick local solution to compare with others.

The results obtained by the three methods on the synthetic instances with 1-fold, 2-fold, and 3-fold training and testing data sets are shown in Tables 13, 14, and 15 respectively. In the tables, the results of the \((C_e, \varepsilon_e)\)-rectangle search algorithm are excerpted from Table 8, 9, and 10. Next to the objective values obtained by KNITRO in the tables, some are marked by a double asterisk (**.) This means that KNITRO returned a solution but it is claimed an infeasible point.

From Tables 13, 14, and 15, the global optimal objective values obtained from the \((C_e, \varepsilon_e)\)-rectangle search algorithm and the improved integer program basically match with each other except for small discrepancies in instances 20_20_30_1, 25_25_30_1 and 15_15_30_2. We think these discrepancies are the result of different precisions on the right-hand-side feasibility. The local optimal solutions provided by KNITRO are quite good considering the running time, yet we can also see that the global optimal solution to the application of cross-validated SVR is always better than the global optimal solution provided by KNITRO.

A series of figures that compare the running time of the \((C_e, \varepsilon_e)\)-rectangle search algorithm and the improved integer program is shown in Figure 14. In general, the \((C_e, \varepsilon_e)\)-rectangle search algorithm can solve many more instances to the global optimum than the improved integer program. The instance 35_35_30_1 is the only instance that was unsolved by the \((C_e, \varepsilon_e)\)-rectangle search algorithm but solved as the improved integer program. However, for instances with fewer training data points, the convergent speed of an improved integer program outperforms that of the \((C_e, \varepsilon_e)\)-rectangle search algorithm, including the 1-fold-5-feature instances.
Figure 14: Comparisons of convergent capability and convergent speed between the \((C_e, \varepsilon_e)\)-rectangle search algorithm and the improved integer program on the same instances.

with up to 55 data points, 1-fold-30-feature instances with up to 30 data points, 2-fold-5-feature instances with up to 25 data points, 2-fold-30-feature instances with up to 15 data points, 3-fold-5-feature instances with up to 15 data points, and 3-fold-30-feature instances with up to 10 data points. As the number of training data points increases, the required processing time for the improved integer program rises suddenly at a critical point, after which the larger instances cannot be solved. This sudden rise confirms the aggregation effect mentioned in Section 4.

Similarly, we employed the three approaches on real-world chemoinformatics data sets. The solution provided by KNITRO on the real-world data is shown in Table 16. Our conclusion about the comparison between KNITRO and the \((C_e, \varepsilon_e)\)-
rectangle search algorithm remains the same: the running time of KNITRO is definitely desirable, while the solution quality of the \((C_e, \varepsilon_e)\)-rectangle search algorithm is absolutely better. Results from the improved integer programming on the real-world data are limited because we only solved the set “cancer” and a few instances in “CCK.” We didn’t actually wait long enough to get the optimal solutions of the remaining instances in “CCK” because it takes 25 days to get a convergence on “CCK_3.” Table 17 shows that the processing time needed for the modified integer program for this set “cancer” is sometimes more than but is sometimes less than the time needed for the \((C_e, \varepsilon_e)\)-rectangle search algorithm. Knowing that the set “CCK” and “BBB” can be solved by the \((C_e, \varepsilon_e)\)-rectangle search algorithm, we conclude that the improved integer program is less effective on instances with a large number of folds.

5.4 Observation on solutions

In all the numerical experiments provided in this work, we suspect that an optimal \((C_e, \varepsilon_e)\) must exist on the boundaries of the initial rectangular area \([\bar{C}, \bar{C}] \times [\bar{\varepsilon}, \bar{\varepsilon}]\). For example, for those instances solved to global optimum shown in Figure 8, either the optimal \(C_e \in \{1, 10\}\) or the optimal \(\varepsilon_e \in \{0.1, 0.5\}\). Even though the optimal \((C_e, \varepsilon_e)\) for the instance 70_70_5_1 was recorded at an interior point, we have noted that a point on the boundaries is also optimal.

In application, though we have not yet able to either prove that the global optimal parameters are always on the boundaries of \([\bar{C}, \bar{C}] \times [\bar{\varepsilon}, \bar{\varepsilon}]\) or find a counter example, it appears to be a good and efficient strategy to follow Algorithm 3.4 and search on the boundaries in order to get a good upper bound. It is highly probable that this upper bound is equal to the true global optimum based on our experiments. Moreover, out of a total of 140 instances (80 generated synthetic instances and 60 real-world instances), the \((C_e, \varepsilon_e)\)-rectangle search algorithm solved 97 of them to global optimum. Among the 97 instances, 96 of them obtain a least valid upper bound at the completion of the 1st stage of the algorithm equivalent to the global optimum found later (except BBB_2 in Table 12). We may be able to apply the 1st stage of the algorithm alone to obtain a set of trustworthy parameters and save running time for the convergence at the 2nd stage.
6 Conclusion

In this chapter, we present our study about selecting the optimal parameters for support vector machine regression (SVR) within the framework of multi-fold cross validation. The conclusion is made in wide aspects:

In modeling, the cross-validated SVR is formulated as a bi-level optimization program, and can be equivalently reformulated as a linear program with complementarity constraints and as an integer program with valid big numbers. For the former reformulation, the feasible set to the complementarity constraints with fixed design parameters forms a linear complementarity problem which can be solved by existing methods such as the semismooth Newton method. For the later reformulation, we discussed the form of valid big numbers and suggested a technique (motivated by [120]) to tighten the big numbers. In this tightening procedure, the McCormick bound (or essentially the Taylor expansion) plays a role in relating the valid big numbers and the valid lower objective bound, and thus enables the two to be refined alternatively. These two reformulations outweigh one another in different situations.

In solution property, we defined the data point allocation in a space separated by two parallel hyperplanes of the SVM and denoted it as the “grouping” vector. On the parameter-plane, we discussed the concept of the “invariancy region” and “invariancy interval” where the grouping remains unchanged, and showed that these regions are convex. Corresponding to the grouping, replacing the complementarities by a “piece” allows us to form a restricted linear program which gives an upper objective bound for the original mathematical program with complementarity constraints. All these properties inspire us to solve for the optimal parameters and global optimum using an algorithm searching on the parameter-plane.

In algorithm development, the rectangle search algorithm consists of two stages. In the 1st stage, the feasible regions are decomposed by a bunch of rectangular areas either being partitioned or eliminated. In the 2nd stage, we aim to identify the single straight line that bisects the rectangular area into two invariancy regions. This straight line is a confirmation of the realization. We provide factual details about the algorithm with which we implemented to obtain the correct solution at convergence for many numerical instances. So far we only found two conditions.
sufficient to conclude that the groupings in a rectangular area are realized: 1) The four vertices have the same grouping. 2) The rectangle contains two groupings (and the condition is as in Step 2 of Algorithm 3.6). A potential direction for improving this rectangle search algorithm is thus in discovering other sufficient conditions.

In numerical experiments, we ran 140 instances including 80 synthetic instances and 60 real-world instances. The synthetic instances were generated by ourselves without natural structure between the indicators and the dependent variables, while the real-world instances are statistical data of chemoinformatics that has also been studied in [120, 72, 33]. A total of 54 of the synthetic instances and 43 of the real-world instances were solved to global optimum, while good valid upper bounds were provided for the remaining instances. We are the first to solve the real-world data sets, though not all of them, to global optimum.

As a practical extension, we propose to categorize the difficulty level of the instances under our parameter-plane search algorithm by sorting instances into a 4-quadrant diagram with four difficulty levels. The categorizing principals is the number of training data in each fold and the number of features. The effect of increasing the number of folds remains within each quadrant because the effect of this factor is of a smaller scale compared with other factors. This categorization helps us to understand the performance of the rectangle search algorithm and to be aware of its limitations. Beyond this, we also gained knowledge about the location of the optimal parameters. The optimal parameter-pair can be located at boundaries or in the interior.

The long processing time for obtaining the global optimum really challenges our patience. The inefficiency of obtaining the global optimal parameters seems to hinder our efforts to embed the optimization into the standard process of model selection. The fact that a certificate of global optimality for the cross-validated SVR doesn’t guarantee the best performance of the resulting SVR model is understood. Rather, we have found parameters that are best selected under the specific training process, cross-validation. The global optimum acts as a benchmark for every local optimum obtained from various methods under the same training process, so this study of the global optimum for parameter selection is meaningful.
### Table 8: Result of 1-fold training and 1-fold testing data solved by the \((C_e, \varepsilon_e)\)-rectangle search algorithm. \(C_e \in [1, 10]\) and \(\varepsilon_e \in [0.1, 0.5]\)
Table 9: Result of 2-fold training and 2-fold testing data solved by the \((C_e, \varepsilon_e)\)-rectangle search algorithm. \(C_e \in [1, 10]\) and \(\varepsilon_e \in [0.1, 0.5]\)
### Table 10: Result of 3-fold training and 3-fold testing data solved by the \((C_e, \varepsilon_e)\)-rectangle search algorithm. \(C_e \in [1, 10]\) and \(\varepsilon_e \in [0.1, 0.5]\)

<table>
<thead>
<tr>
<th>Name</th>
<th># sets</th>
<th># obs.</th>
<th># train. data</th>
<th># test. Data</th>
<th># features</th>
</tr>
</thead>
<tbody>
<tr>
<td>aquasol</td>
<td>10</td>
<td>197</td>
<td>100</td>
<td>97</td>
<td>25</td>
</tr>
<tr>
<td>blood/brain barrier (BBB)</td>
<td>20</td>
<td>62</td>
<td>60</td>
<td>2</td>
<td>25</td>
</tr>
<tr>
<td>cancer</td>
<td>20</td>
<td>46</td>
<td>40</td>
<td>6</td>
<td>25</td>
</tr>
<tr>
<td>cholecystokinin (CCK)</td>
<td>10</td>
<td>66</td>
<td>60</td>
<td>6</td>
<td>25</td>
</tr>
</tbody>
</table>

### Table 11: Profile of the chemoinformatics data. (Table 3.1 in [72])

<table>
<thead>
<tr>
<th>Name of data set</th>
<th># sets</th>
<th># obs.</th>
<th># train. data</th>
<th># test. Data</th>
<th># features</th>
</tr>
</thead>
<tbody>
<tr>
<td>aquasol</td>
<td>10</td>
<td>197</td>
<td>100</td>
<td>97</td>
<td>25</td>
</tr>
<tr>
<td>blood/brain barrier (BBB)</td>
<td>20</td>
<td>62</td>
<td>60</td>
<td>2</td>
<td>25</td>
</tr>
<tr>
<td>cancer</td>
<td>20</td>
<td>46</td>
<td>40</td>
<td>6</td>
<td>25</td>
</tr>
<tr>
<td>cholecystokinin (CCK)</td>
<td>10</td>
<td>66</td>
<td>60</td>
<td>6</td>
<td>25</td>
</tr>
</tbody>
</table>
### Table 12: Result of the real-world, 5-fold training, and 1-fold testing data solved

| Name               | Data           | # comp. | Opt.? | Cc   | εc   | Time to convergence (sec) | # grouping found | # obj. val. upper bound | # rec. area | 1st stage info | 2nd stage info | 2nd stage info | area of the 2nd stage (%) | Solution                | Instance Info | Solving Statistics | The 1st stage info | The 2nd stage info | 2nd stage info | 2nd stage info |
|--------------------|----------------|---------|-------|------|------|---------------------------|------------------|--------------------------|-------------|----------------|----------------|----------------|----------------|----------------|--------------------------------|-------------------|-------------------|------------------|------------------|----------------|------------------|
| aquasol_1          | 97_20_25_5     | 300     | n/a   | 0.1  | 0.187382 | n/a                       | 5260             | 5220                      | 470.703     | 1579           | n/a            | 520.2950       | 57.94%          | n/a            | n/a                                      |
| aquasol_7          | 97_20_25_5     | 300     | n/a   | 0.1  | 0.134748 | n/a                       | 648              | 641                       | 434.990     | 424            | 8064.17        | 70.10%          | n/a            | n/a            | n/a                                      |
| aquasol_9          | 97_20_25_5     | 300     | n/a   | 0.1  | 0.473021 | n/a                       | 379              | 371                       | 420.964     | 304            | 9021.09        | 71.83%          | n/a            | n/a            | n/a                                      |
| aquasol_10         | 97_20_25_5     | 300     | n/a   | 0.1  | 0.01   | n/a                       | 492              | 487                       | 432.577     | 338            | 8089.09        | 59.03%          | n/a            | n/a            | n/a                                      |
| cancer_1           | 6_8_25_5       | 120     | Yes   | 0.1  | 0.208403 | n/a                       | 566              | 550                       | 415.818     | 307            | 8038.09        | 43.78%          | n/a            | n/a            | n/a                                      |
| cancer_5           | 6_8_25_5       | 120     | Yes   | 0.1  | 0.255794 | n/a                       | 635              | 612                       | 470.129     | 302            | 8088.49        | 25.58%          | n/a            | n/a            | n/a                                      |
| BBB_2              | 2_12_25_5      | 120     | Yes   | 0.1  | 0.171155 | n/a                       | 558              | 550                       | 417.380     | 341            | 8018.43        | 58.70%          | n/a            | n/a            | n/a                                      |
| BBB_6              | 2_12_25_5      | 120     | Yes   | 0.1  | 0.01   | n/a                       | 676              | 658                       | 104.679     | 324            | 8000.11        | 60.99%          | n/a            | n/a            | n/a                                      |
| BBB_9              | 2_12_25_5      | 120     | Yes   | 0.1  | 0.437519 | n/a                       | 607              | 599                       | 411.289     | 358            | 8065.61        | 57.21%          | n/a            | n/a            | n/a                                      |
| cancer_1           | 6_8_25_5       | 120     | Yes   | 0.1  | 0.351328 | n/a                       | 621              | 613                       | 463.362     | 478            | 8111.71        | 55.40%          | n/a            | n/a            | n/a                                      |

**CHAPTER 3.**

"C. Ck∈ [0,1,10] and εc∈[0,01,1]"
### Table 13: Comparisons between three methods. Results of solving the cross-
vali-dated SVR with 1-fold training and 1-fold testing data. $C_e \in [1, 10]$ and $
\varepsilon_e \in [0.1, 0.5]$
<table>
<thead>
<tr>
<th>Name</th>
<th>global obj.</th>
<th>local obj.</th>
<th>$C_e$</th>
<th>$\varepsilon_e$</th>
<th>time (sec)</th>
<th>local obj.</th>
<th>$C_e$</th>
<th>$\varepsilon_e$</th>
<th>time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>5_5_5_2</td>
<td>31.7797</td>
<td>31.7797</td>
<td>10</td>
<td>0.5</td>
<td>1.49422</td>
<td>32.4176</td>
<td>1</td>
<td>0.1</td>
<td>0.066989</td>
</tr>
<tr>
<td>10_10_5_2</td>
<td>70.7773</td>
<td>70.7773</td>
<td>1</td>
<td>0.1</td>
<td>17.428</td>
<td>73.2668**</td>
<td>1.0032</td>
<td>0.5</td>
<td>0.050991</td>
</tr>
<tr>
<td>15_15_5_2</td>
<td>84.0852</td>
<td>84.0852</td>
<td>9.97703</td>
<td>0.1</td>
<td>58.3685</td>
<td>93.297</td>
<td>1</td>
<td>0.48358</td>
<td>0.143977</td>
</tr>
<tr>
<td>20_20_5_2</td>
<td>111.454</td>
<td>111.454</td>
<td>1</td>
<td>0.382452</td>
<td>95.4661</td>
<td>111.7276</td>
<td>1.09922</td>
<td>0.44892</td>
<td>0.25996</td>
</tr>
<tr>
<td>25_25_5_2</td>
<td>136.731</td>
<td>136.731</td>
<td>4.42991</td>
<td>0.1</td>
<td>941.125</td>
<td>137.7786</td>
<td>1</td>
<td>0.5</td>
<td>0.373942</td>
</tr>
<tr>
<td>30_30_5_2</td>
<td>155.316</td>
<td>153.316</td>
<td>1</td>
<td>0.269019</td>
<td>160899</td>
<td>156.8355</td>
<td>1</td>
<td>0.381687</td>
<td>0.821875</td>
</tr>
<tr>
<td>35_35_5_2</td>
<td>183.513</td>
<td>183.7393**</td>
<td></td>
<td></td>
<td></td>
<td>183.7393</td>
<td>1</td>
<td>0.5</td>
<td>0.898863</td>
</tr>
<tr>
<td>40_40_5_2</td>
<td>203.311</td>
<td>205.2279**</td>
<td>4.4885</td>
<td>0.5</td>
<td>1.2888</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>45_45_5_2</td>
<td>227.785</td>
<td>228.5208</td>
<td>1</td>
<td>0.5</td>
<td>2.184</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>50_50_5_2</td>
<td>255.149</td>
<td>255.1912</td>
<td>1.58265</td>
<td>0.449468</td>
<td>1.9807</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 14: Comparisons between three methods. Results of solving the cross-validated SVR with 2-fold training and 2-fold testing data. $C_e \in [1, 10]$ and $\varepsilon_e \in [0.1, 0.5]$.
Table 15: Comparisons between three methods. Results of solving the cross-validated SVR with 3-fold training and 3-fold testing data. $C_e \in [1, 10]$ and $\varepsilon_e \in [0.1, 0.5]$.  

<table>
<thead>
<tr>
<th>Name</th>
<th>C</th>
<th>C_e</th>
<th>$\varepsilon_e$</th>
<th>time (sec)</th>
<th>local obj.</th>
<th>global obj.</th>
<th>time (sec)</th>
<th>local obj.</th>
<th>global obj.</th>
<th>time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>5_5_5_3</td>
<td>49.2105</td>
<td>49.2105</td>
<td>49.2105</td>
<td>1</td>
<td>0.1</td>
<td>16.4627</td>
<td>10</td>
<td>0.1</td>
<td>0.053991</td>
<td></td>
</tr>
<tr>
<td>10_10_5_3</td>
<td>99.2702</td>
<td>99.2702</td>
<td>176.936</td>
<td>1</td>
<td>0.1</td>
<td>101.9121**</td>
<td>1</td>
<td>0.5</td>
<td>0.12498</td>
<td></td>
</tr>
<tr>
<td>15_15_5_3</td>
<td>132.017</td>
<td>132.017</td>
<td>4552.27</td>
<td>1</td>
<td>0.1</td>
<td>145.8812</td>
<td>1</td>
<td>0.48358</td>
<td>0.353946</td>
<td></td>
</tr>
<tr>
<td>20_20_5_3</td>
<td>176.307</td>
<td>176.307</td>
<td>0.382452</td>
<td>1</td>
<td>0.1</td>
<td>16630.1</td>
<td>0.1</td>
<td>0.5</td>
<td>0.291954</td>
<td></td>
</tr>
<tr>
<td>25_25_5_3</td>
<td>235.656</td>
<td>235.656</td>
<td>3.37509</td>
<td>1</td>
<td>0.1</td>
<td>3.71889</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>30_30_5_3</td>
<td>259.713</td>
<td>259.713</td>
<td>1.22635</td>
<td>1</td>
<td>0.1</td>
<td>0.978398</td>
<td>0.991848</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>35_35_5_3</td>
<td>298.357</td>
<td>298.357</td>
<td>670.735</td>
<td>1</td>
<td>0.1</td>
<td>1.2908</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>40_40_5_3</td>
<td>334.662</td>
<td>334.662</td>
<td>670.735</td>
<td>1</td>
<td>0.1</td>
<td>1.2908</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>45_45_5_3</td>
<td>364.376</td>
<td>364.376</td>
<td>1.1358</td>
<td>1</td>
<td>0.1</td>
<td>1.1358</td>
<td>0.5</td>
<td>3.35864</td>
<td></td>
<td></td>
</tr>
<tr>
<td>50_50_5_3</td>
<td>407.516</td>
<td>407.516</td>
<td>2.06124</td>
<td>1</td>
<td>0.1</td>
<td>2.06124</td>
<td>1</td>
<td>0.5</td>
<td>3.12357</td>
<td></td>
</tr>
</tbody>
</table>

Table 15: Comparisons between three methods. Results of solving the cross-validated SVR with 3-fold training and 3-fold testing data. $C_e \in [1, 10]$ and $\varepsilon_e \in [0.1, 0.5]$.
Table 16: Results of the cheminformatics cross-validated SVR with 5-fold training and 1-fold testing data solved by KNITRO.
Table 17: Results of the chemoinformatics cross-validated SVR with 5-fold training and 1-fold testing data solved by CPLEX as an improved integer program (with the tightened $\theta$).
Chapter 4: Estimation of Pure Characteristics Demand Models Under Firm’s Competition and Market Optimization

Keywords: pure characteristics demand model, Nash-Bertrand game, parameter estimation, pricing, Mathematical Program with Equilibrium Constraints

1 Introduction

The initiation of the Discrete Choice Demand Model, which decomposes the products sold in markets and views each product as a bundle of product characteristics, can be traced back to McFadden in 1974 [90]. Thereafter, discrete choice became a substantive method for studying consumer behavior. In 1995, Berry, Levinshon, and Pakes proposed a Random Coefficients Logit Demand Model (or BLP model) [15] in which the consumer’s utility function is comprised of taste for the characteristics and an i.i.d. idiosyncratic random variable that is interpreted as taste for the product. The existence of a “taste for the product” variable implies that the dimension of the characteristics space increases whenever new products are introduced into the market. This fact, argued by the authors, imposes a limit on substitution possibilities between products and causes an infinite increase in utility since new
products are constantly being introduced. Later in 2007, Berry and Pakes proposed a Pure Characteristics Demand Model (PCM) [16], which removes the taste for the product variable from the utility function of the BLP model. The PCM implicitly imposes limits in introducing substitutive products in a fixed dimension of the characteristics space and thus revises several counter-intuitive implications in the BLP model.

The methods for estimating PCM in the original work [16] include contraction mapping, element-by-element inverse and homotopy method. These methods are employed to compute the only one unobserved product characteristics. Among them, contraction mapping computes an accurate solution only in specific cases, element-by-element inverse may not converge, and homotopy method can be very slow. We aim to develop a method of estimating that eliminates these restrictions.

The contribution of this work is threefold: first to formulate the PCM estimation problem as a mathematical programming model, specifically, a quadratic program with nonlinear complementarity constraints solvable by existing algorithms and solvers; second to resolve the computational burden of the original estimation method of PCM [16, 108] in equating the true market share with the nonsmooth function of predicted market share; and third to extend the market level data considered in estimating PCM from the observed market share to the competitive environment, a Nash-Bertrand game.

From the point-of-view of a game, this chapter studies the parameter selection problem of a game involving \( F \) firms, \( T \) markets, \( J \) products in each market, \( N \) draws of consumers in each market, and \( K \) characteristics for each product. This game describes the pricing competition of firms and the optimization mechanism of market. The firm aims to maximize it’s profit by determining the product prices, and consumers choose to buy or not to buy a product to maximize their personal utility. For the historical reasons mentioned in the first paragraph, PCM is chosen to describe consumer’s demand. The utility for the whole society covering all markets is assumed to be the aggregation of personal utilities.

The remaining part of this chapter is organized as follows. Section 2 introduces the consumer utility function of PCM. Section 3 develops the competitive environment as a Nash-Bertrand game and defines an equilibrium of \( F + 1 \) players.
Section 4 proposes a variance of the Nash-Bertrand game, which we call "lowest-utility constrained Nash-Bertrand game," to exclude the impractical possibilities of no purchases made in the market. In Section 5, the models that compute estimators of the Generalized Method of Moments (GMM) under two types of Nash-Bertrand game are developed. We also provide a simplified model for estimating PCM disregarding the competitive environment. Section 6 describes the settings and results of numerical experiments, including the procedures for generating game-simulated data, instrumental matrices and weight matrices of GMM. Section 7 discusses the advantages and restrictions of this work.

2 Utility function of the Pure Characteristics Demand Model

In the Pure Characteristics Demand Model (PCM) [16], the utility for consumer $i$ buying product $j$ in market $t$ is defined as

$$u_{ijt} = x_{jt}^T \beta_i - \alpha_i p_{jt} + \xi_{jt},$$

(1)

where

- $x_{jt} \in \mathbb{R}^K$: observed product characteristics,
- $p_{jt}$: price of product $j$ in market $t$,
- $\beta_i \in \mathbb{R}^K$ and $\alpha_i$: consumer specific coefficients, and
- $\xi_{jt}$: the only unobserved characteristic.

Consumer $i$ chooses to buy project $j$ only if it provides the maximum utility. Let $j = 0$ be the "outside option" (e.g., buys nothing) and $u_{i0t} := 0$. If more than one product provides the maximum utility, consumers choose randomly among products. If the maximum utility of products equals the outside option, consumer choose randomly between buying and not buying.

We can define a probability tuple $\{\pi_{ijt}\}_{j=0}^J$ to represent the probability for consumer $i$ to buy product $j$ in market $t$. It is easily verified that this purchasing
mechanism can be written by

$$0 \leq \pi_{ijt} \perp \gamma_{it} - [x_{jt}^T \beta_t - \alpha_i \pi_{jt} + \xi_{jt}] \geq 0,$$

$$0 \leq \gamma_{it} \perp \pi_{0it} = 1 - \sum_{j=1}^{J} \pi_{ijt} \geq 0,$$ (2)

where $\gamma_{it}$ captures consumer $i$’s maximum utility over all products in market $t$ at the price tuple $p$ and

$$\gamma_{it} = \max \left\{ 0, \max_{1 \leq \ell \leq J} \left( x_{\ell t}^T \beta_t - \alpha_i p_{\ell t} + \xi_{\ell t} \right) \right\}. \quad (3)$$

3 Nash-Bertrand game

In this section, we formulate the competitive environment by a game in which $F+1$ players participate. Each of the $F$ firms which produce non-substitutive products aims to maximize profits by determining the prices of their own products given prices of other firms’ products and consumers’ purchasing probability tuple. The remaining one player can be considered as a virtual league of consumers over all the markets, and who maximizes the aggregated utility by determining the purchasing probabilities given prices of products. The objective of maximizing the aggregated utility is also referred to as market optimization. The models introduced in this section is developed in [96]. This Nash-Bertrand game is formulated as a linear complementarity problem, and the equilibrium exists.

3.1 Firm’s pricing problem

The firm $f$’s pricing problem given the prices $p$ of all products available in the market and the purchasing probability $\pi_{ijt}^f$ of the products $j$ produced by firm $f$ is
as follows:

$$\max_{p_f, \pi_f, \gamma} \quad \hat{O}_f(p, \pi_f) \triangleq \frac{1}{N} \sum_{t=1}^{T} \sum_{i=1}^{N} \sum_{j \in J_f} \pi_{ijt} (p_{jt} - mc_{jt})$$

subject to \( \forall t = 1, \ldots, T; \ i = 1, \ldots, N; \) and \( j \in J_f : \)

$$0 \leq \pi_{ijt} \perp \gamma_{it} - \left[ x_{jt}^T \beta_i - \alpha_i p_{jt} + \xi_{jt} \right] \geq 0, \quad (4)$$

$$0 \leq \gamma_{it} \perp 1 - \sum_{j=1}^{J} \pi_{ij't} \geq 0,$$

and \( p_{jt} \geq mc_{jt}, \ \forall j \in J_f; \ \forall t = 1, \ldots, T, \)

where

- \( M_t : \) populations in market \( t, \)
- \( J_f : \) set of products produced by firm \( f, \)
- \( mc_{jt} : \) marginal cost of product \( j \) in market \( t. \)

Model (4) can be written equivalently as:

$$\max_{p_f} \quad O_f(p, \pi) \triangleq \frac{1}{N} \sum_{t=1}^{T} M_t \sum_{i=1}^{N} \sum_{j \in J_f} \pi_{ijt} \left[ \min \left\{ r_{ij0t}, \min_{1 \leq \ell \leq J} r_{ij\ell t}(p_{\ell t}) \right\} - mc_{jt} \right]$$

subject to \( mc_{jt} \leq p_{jt}, \ \forall j \in J_f; \ \forall t = 1, \ldots, T, \quad (5)\)

where the introduced variable \( r_{ij\ell t}(p_{\ell t}) \) we call “pseudo-price” can be interpreted as the price to which product \( j \) adjusted, and at which consumer \( i \)’s utility for buying product \( j \) is the same as buying product \( \ell \) given the price of \( \ell. \) That is,

$$x_{jt}^T \beta_i - \alpha_i r_{ij\ell t}(p_{\ell t}) + \xi_{jt} = x_{jt}^T \beta_i - \alpha_i p_{\ell t} + \xi_{\ell t}$$

and

$$r_{ij0t} \triangleq \frac{x_{jt}^T \beta_i + \xi_{jt}}{\alpha_i}.$$
3.2 Market optimization problem

Assume that the welfare of the whole market (society) is the sum of personal utility. Then the market optimization problem is formulated as:

\[
\begin{align*}
\max_{\pi \geq 0} & \quad O_s(p, \pi) \triangleq \sum_{t=1}^{T} \sum_{i=1}^{N} \left\{ \sum_{j=1}^{J} \pi_{ijt} \left[ x_{jt}^{T} \beta_i - \alpha_i p_{jt} + \xi_{jt} \right] \right\} \\
\text{subject to} & \quad \sum_{j=1}^{J} \pi_{ijt} \leq 1, \quad \forall i = 1, \cdots, N; \forall t = 1, \cdots, T.
\end{align*}
\]

(6)

In this problem, consumers decide the purchasing probability rationally, i.e., following the rule of (2), given the prices of all products available in the market.

3.3 Equilibrium of the \( F + 1 \) players

Now we define the equilibrium of the Nash-Bertrand game within the framework of competitive pricing and overall market optimization as follows.

**Definition 10.** An equilibrium in the Nash-Bertrand game with \( F + 1 \) players (\( F \) firms and 1 virtual league of consumers) is a product-prices and consumer-purchasing probabilities pair

\[
p^* := \left( (p_{jt}^*)^T \right)_{j=1}^{J}, \quad \text{and} \quad \pi^* := \left\{ ((\pi_{ijt}^*)^T)_{i=1}^{N} \right\}_{j=1}^{J}
\]

(7)

such that for each \( f = 1, \cdots, F \)

\[
p_f^* \in \arg \max_{p_f \geq 0} O_f(p_f, p^*_{-f}, \pi^*)
\]

subject to \( mc_{jt} \leq p_{jt}, \forall j \in J_f; t = 1, \cdots, T, \)

and

\[
\pi^* \in \arg \max_{\pi \geq 0} O_s(p^*, \pi)
\]

subject to \( \sum_{j=1}^{J} \pi_{ijt} \leq 1, \forall i = 1, \cdots, N; t = 1, \cdots, T. \)
By letting $v_{ijt} \triangleq r_{ij0t} - \hat{r}_{ijt}$ and $\hat{p}_{jt} \triangleq p_{jt} - mc_{jt}$, the program (5) is rewritten as:

$$\max_{\hat{p}_j, v} \sum_{t=1}^T \frac{M_t}{N} \sum_{i=1}^N \sum_{j \in J_f} \pi_{ijt} \left[ -v_{ijt} + r_{ij0t} - mc_{jt} \right]$$

subject to $\forall t = 1, \cdots, T; \; i = 1, \cdots, N; \; \ell = 1, \cdots, J_f$ and $j \in J_f$:

$$v_{ijt} \geq 0,$$

$$v_{ijt} + \hat{p}_{jt} \geq \frac{x_i^T \beta_i - \alpha_i mc_{jt} + \xi_{jt}}{\alpha_i},$$

and $\hat{p}_{jt} \geq 0$.

A known optimality condition of (8) is of the form

$$0 \leq v_{ijt} \perp \frac{M_t}{N} \pi_{ijt} - \sum_{t=1}^T \lambda_{ijtt} \geq 0,$$

$$\forall i = 1, \cdots, N; \; j \in J_f; \; t = 1, \cdots, T; \;$$

$$0 \leq \hat{p}_{jt} \perp -\sum_{i=1}^N \sum_{j' \in J_f} \lambda_{ijj'} \geq 0,$$

$$\forall j \in J_f; \; t = 1, \cdots, T; \;$$

$$0 \leq \lambda_{ijtt} \perp v_{ijt} + \hat{p}_{jt} - \frac{x_i^T \beta_i - \alpha_i mc_{jt} + \xi_{jt}}{\alpha_i} \geq 0,$$

$$\forall i = 1, \cdots, N; \; j \in J_f; \; \ell = 1, \cdots, J_f; \; t = 1, \cdots, T.$$

On the other hand, the optimality condition of (6) is exactly the complementarity condition (2). Substituting $\hat{p}_{jt} \triangleq p_{jt} - mc_{jt}$ into (2), it becomes

$$0 \leq \pi_{ijt} \perp \gamma_{jt} + \alpha_i \hat{p}_{jt} - (x_i^T \beta_i - \alpha_i mc_{jt} + \xi_{jt}) \geq 0,$$

$$0 \leq \gamma_{jt} \perp 1 - \sum_{j=1}^f \pi_{ijt} \geq 0.$$  \hspace{1cm} (10)

Combining the conditions (9) of each $f$ with conditions (10), the equilibrium of the Nash-Betrand game is a solution to the following linear complementarity problem (LCP).
\[0 \leq v_{ijt} \perp \frac{M_t}{N} \pi_{ijt} - \sum_{\ell=1}^{J} \lambda_{ij\ell t} \geq 0, \forall i = 1, \ldots, N; j = 1, \ldots, J; t = 1, \ldots, T,\]

\[0 \leq \hat{p}_{jt} \perp - \sum_{i=1}^{N} \sum_{j' \in J} \lambda_{ij'jt} \geq 0, \forall j = 1, \ldots, J; t = 1, \ldots, T,\]

\[0 \leq \lambda_{ij\ell t} \perp v_{ij\ell t} + \hat{p}_{\ell t} - \frac{x_{j\ell t}^T \beta_i - \alpha_i mc_{j\ell t} + \xi_{\ell t}}{\alpha_i} \geq 0, \forall i = 1, \ldots, N; j = 1, \ldots, J; \ell = 1, \ldots, J; t = 1, \ldots, T,\]

\[0 \leq \pi_{ijt} \perp \gamma_{it} + \alpha_i \hat{p}_{jt} - (x_{j\ell t}^T \beta_i - \alpha_i mc_{j\ell t} + \xi_{\ell t}) \geq 0, \forall i = 1, \ldots, N; j = 1, \ldots, J; \ell = 1, \ldots, J; t = 1, \ldots, T,\]

\[0 \leq \gamma_{it} \perp 1 - \sum_{j=1}^{J} \pi_{ijt} \geq 0, \forall i = 1, \ldots, N; t = 1, \ldots, T.\]  

(11)

If being written as a matrix representation of LCP,

\[0 \leq z \perp Mz + q \geq 0,\]  

(12)

the Nash-Bertrand game is defined by vectors of the following form:

\[z := \begin{bmatrix} \cdots v_{ijt}, \cdots \hat{p}_{jt}, \cdots \lambda_{ij\ell t}, \cdots \pi_{ij\ell t}, \cdots \gamma_{it}, \cdots \end{bmatrix}^T, \quad (13)\]
By Theorem 3.8.6 in [31], LCP$_{NB}$ (11) has an equilibrium because the $M$ matrix in (12) is copositive, i.e., $z^T M z \geq 0 \ \forall \ z \geq 0$. A proof can be derived using the same method provided in [96]. Moreover, from Theorem 3.3.7 in [31], we can verify that LCP$_{NB}$ (11) has multiple solutions because the $M$ matrix in (12) is not a P-matrix. This fact suggests that we may get some undesired equilibrium of (11) from the solver, such as an all-zero $\pi$, meaning that no consumer is willing to buy anything in the market. This is definitely not the general competitive situation.
we want to assume in our analysis. The Lowest-utility constrained Nash-Bertrand game in the next subsection is what we propose to resolve this issue.

## 4 Lowest-utility constrained Nash-Bertrand game

We add the following constraint for each firm \( f = 1, \cdots, F \) into the firm’s pricing model (5) to assure that the lowest market utility, i.e., the aggregated consumer’s utility contributed by \( f \) is above a threshold \( \delta_f \).

\[
\sum_{i=1}^{N} \sum_{t=1}^{T} \sum_{j \in J_f} [x_{jt}^T \beta_i - \alpha_i p_{jt} + \xi_j] \geq \delta_f. \tag{15}
\]

Given that product price should be at least set at the marginal cost, a natural upper bound of \( \delta_f \) is known and

\[
\delta_f \triangleq \sum_{i=1}^{N} \sum_{t=1}^{T} \sum_{j \in J_f} [x_{jt}^T \beta_i - \alpha_i mc_{jt} + \xi_j] \geq \delta_f > 0. \tag{16}
\]

Similar to the procedure in Section 3.3, letting \( v_{ijt} \triangleq r_{ij0t} - \hat{r}_{ijt} \), and the firm’s pricing (5) with additional constraint (15) is reformulated to

\[
\max_{\bar{p}_{jt}, v} \quad \sum_{t=1}^{T} \sum_{i=1}^{N} \sum_{j \in J_f} \pi_{ijt} [-v_{ijt} + \alpha_i r_{ij0t} - mc_{jt}]
\]

subject to \( \forall t = 1, \cdots, T; \ i = 1, \cdots, N; \ \ell = 1, \cdots, J, \ \text{and} \ j \in J_f : \)

\[
\begin{align*}
    v_{ijt} & \geq 0, \\
    v_{ijt} + \tilde{p}_{jt} & \geq \frac{x_{jt}^T \beta_i - \alpha_i mc_{jt} + \xi_{jt}}{\alpha_i}, \\
    -\sum_{i=1}^{N} \sum_{t=1}^{T} \sum_{j \in J_f} \alpha_i \tilde{p}_{jt} & \geq \delta_f - \sum_{i=1}^{N} \sum_{t=1}^{T} \sum_{j \in J_f} [\alpha_i (r_{ij0t} - mc_{jt})], \\
    \text{and} \quad \tilde{p}_{jt} & \geq 0.
\end{align*} \tag{17}
\]

Introducing an additional multiplier \( \mu_f \), an optimality condition to (17) yields the
Combining the conditions (18) of each \( f \) and conditions (10), the equilibrium of the lowest-utility Nash-Betrand game is a solution to the following LCP.
The existence and non-uniqueness results of \( \text{LCP}_{LUNB} \) are similar to those of \( \text{LCP}_{NB} \). With large enough choices of \( \delta_f \), an incentive for consumers to purchase is imposed on the game.

5 PCM Estimation in game

In this section, we derive the models for estimating the utility \( u_{ijt} \) (1). These estimators capture competitive mechanisms of a Nash-Betrand game (or lowest-utility Nash-Bertrand game), observed quantities including the number of products sold in the market and the observed product price, and they reveal the distribution of coefficients for the product characteristics (usually assumed normal-distributed), the reversed-sign price (usually assumed log-normal-distributed) and the marginal cost (usually assumed normal-distributed). Besides that, the generalized method of
moments (GMM) is used for finding the unobserved residuals in PCM utility and marginal cost. We show that the estimation model becomes a tractable quadratic program with complementarity constraints. By adding the sum square of the deviation from incumbent estimators in the GMM style of objective function, we get a restricted set of satisfying estimators that are least perturbed from the incumbent ones.

5.1 Observed quantities

Given the market population \( M_t \), number of draws \( N \), the consumers purchasing probability \( \pi_{ijt} \) and the observed quantity \( q_{jt} \) of product \( j \) sold in market \( t \), the following equation should hold:

\[
q_{jt} = \frac{M_t}{N} \sum_{i=1}^{N} \pi_{ijt}.
\]  

(20)

If given the observed market share \( S_{jt} \) instead of \( q_{jt} \), the following equation should hold:

\[
S_{jt} = \frac{1}{N} \sum_{i=1}^{N} \pi_{ijt}.
\]  

(21)

In the introduction we mentioned the difficulty of equating the estimated market share \( s_{jt} \) to the observed one. It means to find estimators \( \Theta = \{\alpha_i, \beta_{ik}\} \) such that

\[
s_j(x_t, p_t, \xi_t(\Theta); \Theta) = S_{jt},
\]  

(22)

where

\[
s_j(x_t, p_t, \xi_t(\Theta); \Theta) \approx \frac{1}{N} \sum_{i=1}^{N} \left\{ x_{jt}^{T} \beta_i - \alpha_i p_{jt} + \xi_{jt} \geq \max_{1 \leq \ell \leq J} \left\{ x_{\ell t}^{T} \beta_i - \alpha_i p_{\ell t} + \xi_{\ell t}, 0 \right\} \right\}.
\]  

(23)

This is hard because of the existence of the nonsmooth indicator function \( 1 \).\footnote{The approximation of \( s_j(x_t, p_t, \xi_t; \Theta) \) in (23) is formulated by the Monte Carlo simulation taking \( N \) draws of \( \beta_i \) and \( \alpha_i \).} We can see that this difficulty does not exist if the purchasing probability variable \( \pi \) is introduced and is constrained by conditions (2) and (21).

Besides the quantity of sold products, the observed price \( p_{jt}^{obs} \) of product \( j \) in
market \( t \) satisfies
\[
p_{jt}^{obs} = \hat{p}_{jt} + mc_{jt}
\] 
by definition.

### 5.2 Structural parameters

We follow the assumption in [16, 108] for the structures of the “random coefficients” \( \alpha_i \) and \( \beta_{ik} \). The logarithm value of the consumer’s taste for negative price, \( \log \alpha_i \), is log-normally distributed with mean \( \bar{\alpha} \) and variance \( \bar{\alpha}^2 \), and the consumer’s taste for product characteristics, \( \beta_{ik} \), is normally distributed with mean \( \bar{\beta}_i \) and variance \( \sigma_{\beta_k} \). These distributions are written as
\[
\beta_{ik} = \bar{\beta}_k + \sigma_{\beta_k} \eta_{ik}, \quad \text{and} \\
\alpha_i = \exp(\bar{\alpha} w_i), \tag{25}
\]
where \( \eta_{ik} \) and \( w_i \) are Gaussian noise.

Furthermore, we impose a discrete-choice structure that is similar to the utility in (1) on the marginal cost. The marginal cost is then written as
\[
mc_{jt} = y_{jt}^T \phi + \omega_{jt}, \tag{26}
\]
where \( y_{jt} \in \mathbb{R}^K \) are observed product characteristics, \( \phi \in \mathbb{R}^K \) are corresponding coefficients, and \( \omega_{jt} \) is the only unobserved characteristic that may affect the cost.

### 5.3 Generalized Method of Moments for estimating unobserved characteristics

The Generalized Method of Moments (GMM) [56] is chosen to estimate the unobserved characteristics \( \xi_{jt} \) in utility (1) and \( \omega_{jt} \) in marginal cost (26) mainly because \( \xi_{jt} \) in \( u_{ijt} \) (1) may be correlated with other explanatory variables. Specifically, \( x_{jt} \) are the explanatory variables in (1) and are uncorrelated with \( \xi_{jt} \), while \( -p_{jt} \) is often argued correlate with \( \xi_{jt} \) (see [16, 108]). This is because the unobservable term \( \xi_{jt} \) in the model developer’s eyes might be observable to firms and thus affect the product price.
The GMM estimators $\hat{\xi}(\Lambda_{\xi})$ and $\hat{\omega}(\Lambda_{\omega})$ satisfy

$$\hat{\xi}(\Lambda_{\xi}) = \arg \min_{\xi} \xi^T \Lambda_{\xi} \xi,$$  

and

$$\hat{\omega}(\Lambda_{\omega}) = \arg \min_{\omega} \omega^T \Lambda_{\omega} \omega.$$  

(27)  

(28)

$Z_{\xi}$ and $Z_{\omega}$ are "instrumental matrices" in $R^{(J \times T) \times J}$ and $\Lambda_{\xi}$, and $\Lambda_{\omega}$ are symmetric and positive definite "weight matrices" in $R^{H \times H}$. The concept of (27) and (28) is to find estimators $\hat{\xi}$ and $\hat{\omega}$ so that $E[Z_{\xi j t} \xi_{jt}]$ and $E[Z_{\omega j t} \omega_{jt}]$ are as close to 0 as possible. Effective choices for weight matrices $\Lambda_{\xi}$ and $\Lambda_{\omega}$ are those that produce small asymptotic variances of $\hat{\xi}$ and $\hat{\omega}$ respectively, such as $Z_{\xi}Z_{\xi}^T$ and $Z_{\omega}Z_{\omega}^T$ ([56, 121]).

5.4 Estimation model—quadratic programs with complementarity constraints

The parameters to be estimated are summarized by the tuple $\theta = \{\tilde{\beta}, \sigma, \bar{a}, \phi\} \in \Upsilon$ where $\Upsilon$ denotes the side constraints of parameters. We aggregate the two conditions (27) and (28) by equal weight and formulate the GMM estimation under Nash-Betrand game as

\[^{3}E[\cdot] \text{ denotes the expectation of } \cdot.\]
QPNCC_{EsP,NB}(Z_ξ; A_ξ; Z_ω; A_ω; M_t; N; q; p_{obs}; x; y; η; w; α):

\[
\min_{\theta \in \mathcal{T}; mc; \xi; \omega; z} \frac{1}{2} \xi^T Z_\xi A_\xi Z_\xi^T \xi + \frac{1}{2} \omega^T Z_\omega A_\omega Z_\omega^T \omega
\]

subject to

\[
\begin{align*}
\forall t = 1, \ldots, T, j = 1, \ldots, J, \text{ and } f = 1, \ldots, F : \\
\frac{M_t}{N} \sum_{i=1}^N \pi_{ijt} &= q_{jt}; \\
\hat{p}_{jt} &= p_{obs}^{jt} - mc_{jt}
\end{align*}
\]

\[
\begin{align*}
\forall t = 1, \ldots, T; i = 1, \ldots, N; \text{ and } j = 1, \ldots, J : \\
&\text{complementarity constraints in LCP}_{NB}
\end{align*}
\]

\[
\begin{align*}
\bullet &\ 0 \leq mc_{jt} \leq p_{obs}^{jt} \\
\bullet &\ \beta_{ik} = \tilde{\beta}_k + \sigma_{\beta k} \eta_{ik}, \ \forall k = 1, \ldots, K, \\
\bullet &\ \alpha_i = \exp(\bar{\alpha}_i w_i) \\
\text{and} &\ mc_{jt} = y_{jt}^T \phi + \omega_{jt}.
\end{align*}
\]

Model (29) is a quadratic program with nonlinear complementarity constraints (QP-NCC). The nonlinearity of the constraints set comes from \(\alpha_i = \exp(\bar{\alpha}_i w_i)\) and the bilinear term \(\alpha_i \hat{p}_{jt}\) in the Nash-Bertrand game. Assuming \(\bar{\alpha}\) known, the model is reduced to a quadratic program with (linear) complementarity constraints (QPCC).

We let

\[QPC_{EsP,NB}(\bar{\alpha}) = QPNCC_{EsP,NB}(Z_ξ; A_ξ; Z_ω; A_ω; M_t; N; q; p_{obs}; x; y; η; w; \bar{\alpha}),\]

where \(\bar{\alpha}\) becomes one of the attributes of QPNCC_{EsP,NB}.

Similarly, the GMM estimation under lowest-utility constrained Nash-Bertrand game is formulated as
CHAPTER 4.

the following sum square of deviation (SSD) terms of the incumbent parameters are not unique. However, if a set of incumbent parameters is available, we can add

\[ \min_{\theta \in \mathcal{Y}; \mu; \lambda; \xi; \omega; \pi; \gamma} \frac{1}{2} \xi^T Z_\xi \Lambda_\xi' Z_\xi \xi + \frac{1}{2} \omega^T Z_\omega \Lambda_\omega' Z_\omega \omega \]

subject to

- \( \forall t = 1, \ldots, T \) and \( j = 1, \ldots, J \):
  \[ \frac{M_t}{N} \sum_{i=1}^{N} \pi_{ijt} = q_{jt}; \quad \hat{p}_{jt} = p_{jt}^{obs} - mc_{jt} \]

- \( \forall t = 1, \ldots, T; i = 1, \ldots, N; f = 1, \ldots, F; \) and \( j \in J_f \):
  \begin{align*}
  0 \leq v_{ijt} & \perp \frac{M_t}{N} \pi_{ijt} - \sum_{\ell=1}^{J} \lambda_{ij\ell t} \geq 0, \\
  0 \leq \hat{p}_{jt} & \perp \mu_f \sum_{i=1}^{N} \alpha_i - \sum_{i=j}^{N} \sum_{\ell \in J_f} \lambda_{i\ell j t} \geq 0, \\
  0 \leq \lambda_{ij\ell t} & \perp v_{ijt} + \hat{p}_{jt} - \frac{x_{jt}^T \beta_i - \alpha_i mc_{jt} + \xi_{jt}}{\alpha_i} \geq 0, \forall \ell = 1, \ldots, J, \\
  0 \leq \mu_f & \perp \sum_{i=1}^{N} \sum_{t=1}^{T} \sum_{j \in J_f} \left[ x_{jt}^T \beta_i - \alpha_i p_{jt}^{obs} + \xi_{jt} \right] - \delta_f \geq 0, \\
  0 \leq \gamma_{jt} & \perp \gamma_{jt} + \alpha_i \left( \hat{p}_{jt} - \frac{x_{jt}^T \beta_i - \alpha_i mc_{jt} + \xi_{jt}}{\alpha_i} \right) \geq 0, \\
  0 \leq \gamma_{jt} & \perp 1 - \sum_{j' = 1}^{J} \pi_{ijj' t} \geq 0, \\
  \end{align*}

- \( 0 \leq mc_{jt} \leq p_{jt}^{obs} \)

- \( \beta_{ik} = \bar{\beta}_k + \sigma_{\beta k} \eta_{ik}, \quad \forall k = 1, \ldots, K, \)

- \( \alpha_i = \exp(\bar{\omega} w_i) \)

and

- \( mc_{jt} = y_{jt}^T \phi + \omega_{jt}. \) \hspace{1cm} (30)

and we let

\[ \text{QPNC}_{E_{s P}, L N B} (\tilde{\alpha}) \]
\[ = \text{QPNC}_{E_{s P}, L N B} (Z_\xi; \Lambda_\xi; \Lambda_\omega; \mathcal{M}_t; N; q; p^{obs}; x; y; \eta; w; \tilde{\alpha}; \delta_f). \] \hspace{1cm} (31)

The estimated parameters obtained from \( \text{QPNC}_{E_{s P}, N B} (\tilde{\alpha}) \) and \( \text{QPNC}_{E_{s P}, L N B} (\tilde{\alpha}) \) are not unique. However, if a set of incumbent parameters is available, we can add the following sum square of deviation (SSD) terms of the incumbent parameters
\( \tilde{\beta}^{\text{inc}}, \sigma_\beta^{\text{inc}} \) and \( \phi^{\text{inc}} \) into the objective functions in (29) and (30) to greatly restrict the estimators being identified.

\[
\text{SSD of } \text{QPCC}_{E_{sP,NB}} = \sum_{k=1}^{K} \left[ (\tilde{\beta}_k - \tilde{\beta}^{\text{inc}}_k)^2 + (\sigma_{\beta k} - \sigma^{\text{inc}}_{\beta k})^2 + (\phi_k - \phi^{\text{inc}}_k)^2 \right] \tag{32}
\]

and

\[
\text{SSD of } \text{QPCC}_{E_{sP,LU_{NB}}} = \text{SSE of } \text{QPCC}_{E_{sP,NB}}. \tag{33}
\]

### 5.5 Simplified estimation model

A simplified GMM estimation model, which only considers market level data at the observed quantity of sold products, is written as

\[
\text{QPCC}_{SE_{sP}}(Z_t \xi; \Lambda \xi; M_t; N; q; x; \eta; w):
\]

\[
\min_{\Theta, \xi, \pi, \gamma} \quad \xi^T Z_t \Lambda \xi Z_t^T \xi
\]

subject to

\[
\frac{M_t}{N} \sum_{i=1}^{N} \pi_{ijt} = q_{jt}, \quad \forall \ j = 1, \cdots, J; t = 1, \cdots, T
\]

\[
0 \leq \pi_{ijt} \perp \gamma_{it} - (x_{jt}^T \beta_i - \alpha_i p_{jt} + \xi_{jt}) \geq 0, \quad \forall \ i = 1, \cdots, N; j = 1, \cdots, J; t = 1, \cdots, T,
\]

\[
0 \leq \gamma_{it} \perp 1 - \sum_{j=1}^{J} \pi_{ijt} \geq 0, \quad \forall \ i = 1, \cdots, N; t = 1, \cdots, T,
\]

\[
\beta_{ik} = \bar{\beta}_k + \sigma_{\beta k} \eta_{ik}, \quad \forall \ i = 1, \cdots, N; k = 1, \cdots, K,
\]

and

\[
\alpha_i = \exp(\bar{\alpha} w_i), \quad \forall \ i = 1, \cdots, N. \tag{34}
\]

If the following SSD term:

\[
\text{SSD of } \text{QPCC}_{SE_{sP}} = (\bar{\alpha} - \alpha^{\text{inc}})^2 + \sum_{k=1}^{K} \left[ (\bar{\beta}_k - \tilde{\beta}^{\text{inc}}_k)^2 + (\sigma_{\beta k} - \sigma^{\text{inc}}_{\beta k})^2 \right] \tag{35}
\]

is added into the objective function of (34), the set of \( \Theta \) that solves \( \text{QPCC}_{SE_{sP}} \) is greatly restricted. Model (34) is a quadratic program with linear complementarity constraints, and no a priori knowledge about \( \bar{\alpha} \) is needed.
6 Numerical study

The numerical study in this section aims at validating the parameter estimation model of the pure characteristics demand, which is formulated as a quadratic program with complementarity constraints. We present a procedure for generating the game-simulated market parameters and the instrumental and weight matrices of the generalized method of moments. Following this method, we generate several groups of instances that cover different combinations of size of consumer pool, product characteristics, and number of products. In these experiments, we rely on the solver PATH to solve every linear complementarity program and SNOPT to solve every program with linear complementarity constraints. Notice that SNOPT finds only the stationary point of a complementarity constrained program, but in some runs, we are actually able to verify global optimality at the obtained stationary point. The parameter $\tilde{\alpha}$ is first assumed known and fixed. This makes the complementarity constraints linear and solvable. A sensitivity analysis on choice of $\tilde{\alpha}$ is then performed for some instances.

6.1 List of parameters, sets and variables

The following list specifies the parameters, sets and variables used in $\text{QPCC}_{EsP,NB}(\tilde{\alpha})$.

Parameters:

- $F$: number of firms,
- $T$: number of markets,
- $J$: number of products in each market,
- $N$: number of consumers in each market,
- $K$: product characteristics, and
- $H$: number of instrumental variables in GMM.

- $Z_\xi \in \mathbb{R}^{(T \times J) \times H}$: instrumental matrix for $\xi$ in the GMM type of objective function,
- $A_\xi \in \mathbb{R}^{H \times H}$: weight matrix for $\xi$ in the GMM type of objective function,
- $Z_\omega \in \mathbb{R}^{(T \times J) \times H}$: instrumental matrix for $\omega$ in the GMM type of objective function,
- $A_\omega \in \mathbb{R}^{H \times H}$: weight matrix for $\omega$ in the GMM type of objective function,
- $M_t$, $M \in \mathbb{R}^T$: population in market $t$,
- $p_{jt}^{obs}, p_{obs}^{obs} \in \mathbb{R}^{(J \times T) \times 1}$: observed price,
$x_{jtk}$, $x \in \mathbb{R}^{(J \times T) \times K}$ : product characteristics, a parameter in consumer utility,
$\alpha$, $\alpha \in \mathbb{R}^N$ : the reduction of utility associated with per-unit increases of price,
$\alpha$, $\alpha \in \mathbb{R}^N$: a parameter in $\alpha$'s distribution, assumed known,
$w_i$, $w \in \mathbb{R}^N$ : parameters in $\alpha$’s distribution,
$y_{jtk}$, $y \in \mathbb{R}^{(J \times T) \times K}$ : product characteristics, a parameter in marginal cost,
$q_{jt}$, $q \in \mathbb{R}^{(J \times T) \times 1}$ : observed quantity of products sold,
$\eta_{ik}$, $\eta \in \mathbb{R}^{N \times K}$ : parameters in $\beta_{ik}$’s distribution,
$\bar{\beta}$, $\underline{\beta}$ : lower and upper bound for $\beta$ respectively,
$\bar{\sigma}_\beta$, $\underline{\sigma}_\beta$ : lower and upper bound for $\sigma_\beta$ respectively, and
$\bar{\phi}$, $\underline{\phi}$ : lower and upper bound for $\phi$ respectively.

Set:
$J_f$ : products produced by firm $f$, $f=1, \ldots, F$.

Variables:
$\xi_{jt}$, $\xi \in \mathbb{R}^{(J \times T) \times 1}$ : unobserved characteristics,
$\pi_{ijt}$, $\pi \in \mathbb{R}^{N \times J \times T}$ : probability of buying product $j$ in market $t$ for consumer $i$,
$mc_{jt}$, $mc \in \mathbb{R}^{(J \times T) \times 1}$ : marginal cost,
$\hat{p}_{jt}$, $\hat{p} \in \mathbb{R}^{(J \times T) \times 1}$ : $= p_{jt} - mc_{jt}$, where $p_{jt}$ is the product price,
$\nu_{ijt}$, $\nu \in \mathbb{R}^{N \times J \times T}$ : multiplier,
$\lambda_{ijt}$, $\lambda \in \mathbb{R}^{N \times J \times J \times T}$ : multiplier,
$\beta_{ik}$, $\beta \in \mathbb{R}^{N \times K}$ : parameters in utility function,
$\bar{\beta}_k$, $\underline{\beta}$ $\in \mathbb{R}^K$ : parameters in $\beta$’s distribution,
$\sigma_{\beta k}$, $\sigma_\beta \in \mathbb{R}^K$ : parameters in $\beta$’s distribution,
$\phi_k$, $\phi \in \mathbb{R}^K$ : parameters in $mc$’s distribution,
$\omega_{jt}$, $\omega \in \mathbb{R}^{(J \times T) \times 1}$ : parameters in $mc$’s distribution, and
$\gamma_{jt}$, $\gamma \in \mathbb{R}^{N \times T}$ : multiplier.

6.2 Selection of the instrumental matrices and weight matrices

Consider that $\xi_{jt}$ can be correlated with $-p_{jt}$ in a consumer’s utility function $u_{ijt} = x_{jt}^T \beta - \alpha p_{jt} + \xi_{jt}$. For estimating $u_{ijt}$ with GMM, an instrumental matrix $Z_\xi$ is required to satisfy the orthogonality condition ($E[\xi^T Z_\xi] = 0$), rank condition
Example 11. Generate instrumental matrix for $K = 4$ and $H$ up to 33.

$Z_{\xi,1} = x_1$

$Z_{\xi,2} = x_2$

$Z_{\xi,3} = x_3$

$Z_{\xi,4} = x_4$

$Z_{\xi,5} = \left(\frac{x_1 + x_2 + x_3 + x_4}{4}\right) \circ \left(\frac{x_1 + x_2 + x_3 + x_4}{4}\right)$

$Z_{\xi,6} = 1$

$Z_{\xi,7} = \text{median}(x_1, x_2, x_3, x_4)$

$Z_{\xi,8} = x_1 \circ x_2$

$Z_{\xi,9} = x_2 \circ x_3$

$Z_{\xi,10} = x_3 \circ x_4$

$Z_{\xi,11} = x_4 \circ x_1$

$Z_{\xi,12} = x_1 \circ x_2 \circ x_3$

$Z_{\xi,13} = x_2 \circ x_3 \circ x_4$

$Z_{\xi,14} = x_1 \circ x_2 \circ x_4$

$Z_{\xi,15} = x_1 \circ x_2 \circ x_3 \circ x_4$

$Z_{\xi,16} = \left(\frac{Z_{\xi,1} + Z_{\xi,2} + \ldots + Z_{\xi,15}}{15}\right) \circ \left(\frac{Z_{\xi,1} + Z_{\xi,2} + \ldots + Z_{\xi,15}}{15}\right)$

$Z_{\xi,17} = \left(\frac{Z_{\xi,1} + Z_{\xi,2} + \ldots + Z_{\xi,15}}{15}\right) \circ \left(\frac{Z_{\xi,1} + Z_{\xi,2} + \ldots + Z_{\xi,15}}{15}\right) \circ \left(\frac{Z_{\xi,1} + Z_{\xi,2} + \ldots + Z_{\xi,15}}{15}\right)$

$Z_{\xi,18} = \text{median}(Z_{\xi,1}, Z_{\xi,2}, \ldots, Z_{\xi,17})$

$Z_{\xi,19} = \left(\text{median}(Z_{\xi,1}, Z_{\xi,2}, \ldots, Z_{\xi,17})\right) \circ \left(\text{median}(Z_{\xi,1}, Z_{\xi,2}, \ldots, Z_{\xi,17})\right)$

$Z_{\xi,20} = x_1 \circ x_1$

$Z_{\xi,21} = x_2 \circ x_2$

$Z_{\xi,22} = x_3 \circ x_3$

$Z_{\xi,23} = x_4 \circ x_4$

$Z_{\xi,24} = x_1 \circ x_1 \circ x_2 \circ x_2$

$Z_{\xi,25} = x_2 \circ x_2 \circ x_3 \circ x_3$

$^{4K+1}: K$ dimension of $x_{jt}$ and 1 dimension of $-p_{jt}$
\[ Z_{\xi,26} = x_3 \circ x_3 \circ x_4 \circ x_4 \]
\[ Z_{\xi,27} = x_1 \circ x_2 \circ x_3 \circ x_4 \]
\[ Z_{\xi,28} = x_1 \circ x_1 \circ x_4 \circ x_4 \]
\[ Z_{\xi,29} = x_2 \circ x_2 \circ x_4 \circ x_4 \]
\[ Z_{\xi,30} = \left( \frac{Z_{\xi,1} + Z_{\xi,2} + \ldots + Z_{\xi,29}}{29} \right) \circ \left( \frac{Z_{\xi,1} + Z_{\xi,2} + \ldots + Z_{\xi,29}}{29} \right) \]
\[ Z_{\xi,31} = \left( \frac{Z_{\xi,1} + Z_{\xi,2} + \ldots + Z_{\xi,29}}{29} \right) \circ \left( \frac{Z_{\xi,1} + Z_{\xi,2} + \ldots + Z_{\xi,29}}{29} \right) \]
\[ Z_{\xi,32} = \text{median}(Z_{\xi,1}, Z_{\xi,2}, \ldots, Z_{\xi,29}) \]
\[ Z_{\xi,33} = \left( \text{median}(Z_{\xi,1}, Z_{\xi,2}, \ldots, Z_{\xi,29}) \right) \circ \left( \text{median}(Z_{\xi,1}, Z_{\xi,2}, \ldots, Z_{\xi,29}) \right) \]

where \( Z_{\xi,i} \) denotes the \( i^{th} \) column of \( Z_{\xi} \), \( x_i \) denotes the \( i^{th} \) column of \( x \), 1 is a vector whose entries are all one, \( v_1 \circ v_2 \) is the Hadamard (entry-wise) product of vector \( v_1 \) and \( v_2 \), and function \( \text{median}(v_1, v_2, \ldots, v_n) \) produces a vector whose \( j^{th} \) entry is the median in the set comprising the \( j^{th} \) entries of \( v_1, v_2, \ldots, v_n \).

Other than the instances generated by Example 11, when \( K \neq 4 \), including \( K = 3, 8, \) or 11, we use the procedure as in Example 12 to generate the instances. These instances have \( H \) up to \( K + 2 \). When \( H < K + 2 \), the instrumental matrix contains the first \( H \) columns.

**Example 12.** Generate instrumental matrix for \( K \neq 4 \) and \( H \) up to \( K + 2 \). for \( k = 1 \ldots K \)
\[ Z_{\xi,k} = x_k \]
end
\[ Z_{\xi,(K+1)} = \left( \frac{x_1 + x_2 + \ldots + x_K}{K} \right) \circ \left( \frac{x_1 + x_2 + \ldots + x_K}{K} \right) \]
\[ Z_{\xi,(K+2)} = 1 \]

In estimating \( m_{cjt} = y_j^T \phi + \omega_{jt} \), the explanatory variable \( y \) is not correlated with the residual \( \omega \). Similar to \( Z_{\xi} \), an instrumental matrix \( Z_{\omega} \) needed to satisfy \( E[\omega^T Z_{\omega}] = 0 \), \( E[Z_{\omega}^T y] \in \mathbb{R}^{H \times K} \) has rank \( K \), and \( H \geq K \). The \( Z_{\omega} \) is generated as in Example 11 and Example 12 but every \( x \) is replaced with \( y \).

The weight matrices \( A_{\xi} \) in the experiments are generated by the following procedure:

**Weight Matrix Generation:**

**Step 1:** Compute \( B = Z_{\xi} Z_{\xi}^T \) and find the smallest eigenvalue \( \epsilon_\ell \) of \( B \).
Step 2: If \( e_\ell \leq 0 \), \( A = B + (|e_\ell| + 10^{-4})I_H \). (\( I_H \in \mathbb{R}^{H \times H} \) is an identity matrix.)

Step 3: Compute \( \Lambda_\xi = A^{-1} \).

If \( H \leq J \times T \), the smallest eigenvalue \( e_\ell \) obtained in Step 1 is nonnegative. When \( e_\ell \) is 0, a small amount \( 10^{-4} \) is added to the diagonal entries to make \( A \) positive definite in Step 2 without harming the original structure too much. If \( H > J \times T \), matrix \( B \) can’t have full rank, and the smallest eigenvalue \( e_\ell \) is negative. The absolute value of the smallest eigenvalue and a very small value (here \( 10^{-4} \)) is then added to the diagonal entries in Step 2. This step guarantees that \( A \) and \( A^{-1} \) are positive definite. The generation of \( \Lambda_\omega \) is basically the same, but \( B = Z_\omega Z_\omega^T \) in Step 1.

### 6.3 Game-simulated data generation

The coefficients inputting to the \( \text{QPCC}_{E_kP,N_B} \) are generated by the following procedure.

**Game-simulated Data Generation:**

**Step 1:** Determine \( N, T, J, K, F, \) and \( J_f \). Select \( \beta \leq \beta \leq \bar{\beta}, \sigma_\beta \leq \sigma_\beta \leq \sigma_\beta, \phi \leq \phi \leq \bar{\phi}, \) and \( \alpha \).

**Step 2:** Generate \( \eta \sim \text{Normal}(0, 1) \), then compute \( \beta_{ik} = \beta_k + \sigma_\beta \eta_{ik} \).

**Step 3:** Generate \( w \sim \text{Normal}(0, 1) \), then compute \( \alpha_i = \exp(\alpha w_i) \).

**Step 4:** Generate \( x \sim \text{Uniform}(0, 1) \) and \( y \sim \text{Uniform}(0, 1) \).

**Step 5a:** Generate \( Z_\xi \in \mathbb{R}^{(T \times J) \times H} \).

5b: Generate \( \xi_{jt} \in \text{null}(Z_\xi) \).

**Step 6a:** Generate \( Z_\omega \in \mathbb{R}^{(T \times J) \times H} \).

6b: Generate \( \omega_{jt} \in \text{null}(Z_\omega) \).

6c: Calculate \( mc_{jt} = y_{jt}^T \phi + \omega_{jt} \). If \( mc_{jt} < 0 \), adjust it to 0.

**Step 7:** Solve the LCP (11) (or an alternative LPCC with the objective \( \sum_{ijt} \pi_{ijt} \)).

Get feasible solution \( \hat{p}_{jt} \), and \( \pi_{ijt} \).
Step 8: Obtain \( q_{jt} = \frac{M_t}{N} \sum_{i=1}^{N} \pi_{ijt} \) and \( p_{jt}^{obs} = \hat{p}_{jt} + mc_{jt} \).

The idea is simple. We arbitrarily select \( \beta, \sigma_\beta, \phi \) and \( \alpha \), randomly generate Gaussian noise \( \eta, \omega, \xi \), and assume that the product characteristics \( x \) and \( y \) are uniformly distributed. The values \( q_{jt} \) and \( \pi_{ijt} \) are an equilibrium to the simulated Nash-Bertrand price game. The QPCCEsP with parameters generated by this procedure is guaranteed feasible and captures the competition nature. In Step 1 of the procedure, \( \phi \) is a trick to possibly avoid \( mc \) falling below zero. And since the \( y \) generated here is nonnegative, \( \phi \geq 0 \). In the implementation we let \( \phi = 2 \).

We generate \( \xi_{jt} \) and \( \omega_{jt} \) in the null space of \( Z_\xi \) and \( Z_\omega \) in Step 5b and 6b respectively. This step guarantees the existence of a feasible \( \xi_{jt} \) (\( \omega_{jt} \)) such that \( \xi^T Z_\xi = 0 \) (\( \omega^T Z_\omega = 0 \)). The case where the second moments are exactly zero \( (\xi^T Z_\xi = 0 \text{ and } \omega^T Z_\omega = 0) \) is an absolutely perfect situation rarely found on the surface of the earth. Therefore, we perturb \( \xi_{jt} \) and \( \omega_{jt} \) so they are not in the null space of \( Z_\xi \) and \( Z_\omega \) to form some experimental problems. We will see the differences of these two types of instances in the numerical results.

6.4 Numerical experiments and results

In the numerical study, we use solvers to solve the Nash-Bertrand game (11), lowest-utility constrained Nash-Bertrand game (19) (within the process of game-simulated data generation), and the GMM estimation problems (29) and (30). The models were implemented in the format of AMPL. All the experiments were run on a machine with CPU: Intel i7-2600K, memory: 16 GB, and OS: Windows 7.

Data of Nash-Bertrand game

The parameters of the Nash-Bertrand game instances, which are generated with Step 1-6 of the game-simulated data generation procedure, are summarized in Table 18. In Table 18, for those instances with \( F = 2 \), we let \( J_1 = \{1, 3\} \) and \( J_2 = \{2, 4\} \). For P11, \( J_1 = \{1, 2, 3\} \), \( J_2 = \{4, 5\} \), \( J_3 = \{6, 7, 8\} \), \( J_4 = \{9\} \) and \( J_5 = \{10\} \). Among the 11 instances, P1, P3, P4, P6, P9, and P11 have \( \omega \) and \( \xi \) generated inside the null space of \( Z_\omega \) and \( Z_\xi \) respectively. Considering the homoge-
### Pricing Game Instances

<table>
<thead>
<tr>
<th>Name</th>
<th>N</th>
<th>J</th>
<th>K</th>
<th>T</th>
<th>F</th>
<th>H</th>
<th>Fixed Parameters</th>
<th>Markets Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>5</td>
<td>4</td>
<td>11</td>
<td>3</td>
<td>2</td>
<td>12</td>
<td>$\bar{\alpha} = 1; ; \bar{\beta} = [40.26, 39.21, 37.01, 31.9, 38.67, 44.18, 32.32, 31.56, 37.339, 30.67, 33.84]; ; \sigma_\beta = [0, 1.45, 7.18, 6.62, 4.32, 4.46, 5.08, 5.28, 5.73, 3.61, 3.36]; ; \phi = [2.35, 2.17, 2.79, 3.61, 2.02, 2.47, 3.87, 2.45, 3.57, 2.82, 2.24]; ; \xi = 0, ; \omega = 0. $</td>
<td>3 markets: 5000, 5100, 5200</td>
</tr>
<tr>
<td>P2</td>
<td>5</td>
<td>4</td>
<td>11</td>
<td>3</td>
<td>2</td>
<td>12</td>
<td>The same with P1 except $\xi \neq 0$ and $\omega \neq 0$ and they are not in the null space of $Z_\xi$ and $Z_\omega$, respectively.</td>
<td></td>
</tr>
<tr>
<td>P3</td>
<td>50</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td>$\bar{\alpha} = 1; ; \bar{\beta} = [40.26, 39.21, 37.01]; ; \sigma_\beta = [0, 4.34, 7.09]; ; \phi = [2.23, 2.16, 2.74]; ; \xi = 0, ; \omega = 0. $</td>
<td>20 markets: 5000~6900 with an increment of 100</td>
</tr>
<tr>
<td>P4</td>
<td>50</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td>The same with P3 except $\xi \neq 0$ and $\omega \neq 0$.</td>
<td>5000~6900 with an increment of 100</td>
</tr>
<tr>
<td>P5</td>
<td>50</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td>The same with P3 except $\xi \neq 0$ and $\omega \neq 0$ and they are not in the null space of $Z_\xi$ and $Z_\omega$, respectively.</td>
<td>100</td>
</tr>
<tr>
<td>P6</td>
<td>10</td>
<td>4</td>
<td>4</td>
<td>5</td>
<td>2</td>
<td>20</td>
<td>$\bar{\alpha} = 1; ; \bar{\beta} = [40.26, 39.21, 37.01, 31.9, 38.67, 44.18, 32.32, 31.56]; ; \sigma_\beta = [0, 7.09, 1.16, 0.78]; ; \phi = [2.74, 2.07, 2.38, 2.94]; ; \xi = 0, ; \omega = 0. $</td>
<td>5 markets: 5000~5400 with an increment of 100</td>
</tr>
<tr>
<td>P7</td>
<td>10</td>
<td>4</td>
<td>4</td>
<td>5</td>
<td>2</td>
<td>20</td>
<td>The same with P6 except $\xi \neq 0$ and $\omega \neq 0$ and they not in the null space of $Z_\xi$ and $Z_\omega$, respectively.</td>
<td>100</td>
</tr>
<tr>
<td>P8</td>
<td>10</td>
<td>4</td>
<td>4</td>
<td>5</td>
<td>2</td>
<td>20</td>
<td>The same as in P7 except $\xi$ and $\omega$ are different from those in P7.</td>
<td></td>
</tr>
<tr>
<td>P9</td>
<td>10</td>
<td>4</td>
<td>4</td>
<td>5</td>
<td>2</td>
<td>20</td>
<td>$\bar{\alpha} = 1; ; \bar{\beta} = [0.05, -0.16, -0.6, -1.62]; ; \sigma_\beta = [0, 0.71, 0.12, 0.08]; ; \phi = [2.74, 2.07, 2.38, 2.94]; ; \xi = 0; ; \omega = 0. $</td>
<td>5 markets: 200~204 with an increment of 1</td>
</tr>
<tr>
<td>P10</td>
<td>100</td>
<td>4</td>
<td>4</td>
<td>10</td>
<td>2</td>
<td>33</td>
<td>The same with P9 except $\xi \neq 0$ and $\omega \neq 0$ and they not in the null space of $Z_\xi$ and $Z_\omega$, respectively.</td>
<td>10 markets: 200~209 with an increment of 1</td>
</tr>
<tr>
<td>P11</td>
<td>100</td>
<td>10</td>
<td>8</td>
<td>10</td>
<td>5</td>
<td>10</td>
<td>$\bar{\alpha} = 1; ; \bar{\beta} = [40.26, 39.21, 37.01, 31.9, 38.67, 44.18, 32.32, 31.56]; ; \sigma_\beta = [0, 0.34, 1.92, 4.71, 1.45, 7.18, 6.62, 4.32]; ; \phi = [2.89, 3.02, 3.06, 3.15, 2.72, 2.67, 2.35, 2.17]; ; \xi = 0, ; \omega = 0. $</td>
<td>10 markets: 5000~5900 with an increment of 100</td>
</tr>
</tbody>
</table>

Table 18: Values of $\bar{\beta}$, $\bar{\alpha}$, $\sigma_\alpha$, $\phi$, $\xi$, $\omega$, and $M_t$ in the instances of Nash-Bertrand game.
neous system $\omega^T Z_\omega = 0$ and $\xi^T Z_\xi = 0$, $\omega$ and $\xi$ can take nonzero values only when $J \times T > H$ (the case in P4). In other cases, $\omega$ and $\xi$ are simply zero. We also use the perturbed $\omega$ and $\xi$ that are outside the null space of instrumental matrices $Z_\omega$ and $Z_\xi$ respectively in the instances P2, P5, P7, P8, and P10 to test the estimation model. The assumption of $\omega$ affects the game because it is used in the computation of $mc$ in Step 6c of the game-simulated data generation procedure, and $\xi$ is directly used in the definition of the game.

One way to avoid getting an all zero $\pi$ equilibrium in $\text{LCP}_{NB}$, other than employing the alternative $\text{LCP}_{LUNB}$ formulation, is to add an objective function

$$\sum_{ijt} \pi_{ijt}$$

upon (11). This way, an LPCC that maximizes the total probability of purchasing should be solved instead of an LCP. We use SNOPT to solve for a stationary point to this LPCC. In this case, the lack of a certificate of optimality is not harmful because the main purpose is to avoid $\pi = 0$ if there are other solutions. The additional objective function can be interpreted as a mechanism that forces consumers to purchase in the markets. Results of solving for an equilibrium maximizing the buying probability in the Nash-Bertrand game instances are shown in Table 19. From the solutions $\pi_{ijt}$, the consumers’ choices accumulate at a small portion of the products among all that are available in the markets. This phenomenon reflects that this model lacks a mechanism to push the firms to raise their market share away from zero.

The equilibrium solutions $q_{jt}$ and $p_{jt}^{obs}$ to the Nash-Bertrand game are used as the parameters in the GMM estimation problem. In our experiments, the $\tilde{\beta}^{inc}$, $\sigma_\beta^{inc}$, and $\phi^{inc}$ in the SSD term (32) are set at the $\tilde{\beta}$, $\sigma_\beta$, and $\phi$ values that we used to generate the game. Meanwhile, the constraints of known upper and lower bounds for the parameters are imposed:

$$\Gamma = \left\{ \begin{array}{l} \frac{\tilde{\beta}}{\beta} \leq \beta \leq \frac{\tilde{3}}{\beta}, \\ \frac{\sigma_\beta}{\sigma_\beta} \leq \sigma_\beta \leq \frac{\sigma_{\beta}}{\sigma_{\beta}}, \\ \frac{\phi}{\phi} \leq \phi \leq \frac{\phi}{\phi}. \end{array} \right\}$$

\[ (37) \]
### Table 19: Results of solving Nash-Bertrand game with an objective function $\sum_{ijt} \pi_{ijt}$.

The instances of parameter estimation, where $\bar{\alpha}$ is assumed known and fixed at 1, are solved by SNOPT. We first present the typical cases when $J \times T \geq H > K$.

The results are shown in Table 20, where the estimation error is in the format of a mean square error (MSE):

$$\text{MSE of } \bar{\beta} = \frac{1}{K} \sum_{k=1}^{K} \left( \bar{\beta}_k - \bar{\beta}_{\text{inc}} \right)^2,$$

$$\text{MSE of } \sigma_\beta = \frac{1}{K} \sum_{k=1}^{K} \left( \sigma_{\beta_k} - \sigma_{\beta_{\text{inc}}} \right)^2,$$

$$\text{MSE of } \phi = \frac{1}{K} \sum_{k=1}^{K} \left( \phi_k - \phi_{\text{inc}} \right)^2.$$ (38)
### Table 20: Results of the parameters estimation. $\hat{\alpha} = 1$. The objective function minimizes the orthogonality in GMM and the sum of squared deviation of parameters.

<table>
<thead>
<tr>
<th>Name</th>
<th>N</th>
<th>J</th>
<th>K</th>
<th>T</th>
<th>F</th>
<th>H</th>
<th>N-B Game Setting</th>
<th>Estimation Instances</th>
<th>Bounds for Parameters</th>
<th>Selection of H</th>
<th>SNOPT Solving Statistics</th>
<th>Estimation Error (MSE)</th>
<th>$\xi$ and $\omega$ Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>E1</td>
<td>5</td>
<td>4</td>
<td>11</td>
<td>3</td>
<td>2</td>
<td>12</td>
<td>P1 (zero)</td>
<td>JT = H &gt; K</td>
<td>[30,50]</td>
<td>[0,10]</td>
<td>[2,4]</td>
<td>387</td>
<td>593</td>
</tr>
<tr>
<td>E2</td>
<td>5</td>
<td>4</td>
<td>11</td>
<td>3</td>
<td>2</td>
<td>12</td>
<td>P2 (not in null)</td>
<td>JT = H &gt; K</td>
<td>25080</td>
<td>25868</td>
<td>2.73E-12</td>
<td>1222</td>
<td>optimal</td>
</tr>
<tr>
<td>E3</td>
<td>50</td>
<td>4</td>
<td>32</td>
<td>4</td>
<td>2</td>
<td>20</td>
<td>P3 (zero)</td>
<td>JT &gt; H &gt; K</td>
<td>1.82E-12</td>
<td>23116</td>
<td>0.00E+00</td>
<td>0.00E+00</td>
<td>optimal</td>
</tr>
<tr>
<td>E4</td>
<td>50</td>
<td>4</td>
<td>32</td>
<td>4</td>
<td>2</td>
<td>20</td>
<td>P4 (nonzero)</td>
<td>JT &gt; H &gt; K</td>
<td>1.82E-12</td>
<td>23116</td>
<td>0.00E+00</td>
<td>0.00E+00</td>
<td>optimal</td>
</tr>
<tr>
<td>E5</td>
<td>50</td>
<td>4</td>
<td>32</td>
<td>4</td>
<td>2</td>
<td>20</td>
<td>P5 (not in null)</td>
<td>JT &gt; H &gt; K</td>
<td>1.82E-12</td>
<td>23116</td>
<td>0.00E+00</td>
<td>0.00E+00</td>
<td>optimal</td>
</tr>
<tr>
<td>E6</td>
<td>10</td>
<td>4</td>
<td>45</td>
<td>2</td>
<td>20</td>
<td>P6 (zero)</td>
<td>JT = H &gt; K</td>
<td>1.46E-11</td>
<td>2954</td>
<td>0.00E+00</td>
<td>0.00E+00</td>
<td>optimal</td>
<td>0.00E+00</td>
</tr>
<tr>
<td>E7</td>
<td>10</td>
<td>4</td>
<td>45</td>
<td>2</td>
<td>20</td>
<td>P7 (not in null)</td>
<td>JT &gt; H &gt; K</td>
<td>1.82E-12</td>
<td>23116</td>
<td>0.00E+00</td>
<td>0.00E+00</td>
<td>optimal</td>
<td>0.00E+00</td>
</tr>
<tr>
<td>E8</td>
<td>10</td>
<td>4</td>
<td>45</td>
<td>2</td>
<td>20</td>
<td>P8 (not in null)</td>
<td>JT &gt; H &gt; K</td>
<td>1.82E-12</td>
<td>23116</td>
<td>0.00E+00</td>
<td>0.00E+00</td>
<td>optimal</td>
<td>0.00E+00</td>
</tr>
<tr>
<td>E9</td>
<td>10</td>
<td>4</td>
<td>45</td>
<td>2</td>
<td>20</td>
<td>P9 (not in null)</td>
<td>JT &gt; H &gt; K</td>
<td>1.82E-12</td>
<td>23116</td>
<td>0.00E+00</td>
<td>0.00E+00</td>
<td>optimal</td>
<td>0.00E+00</td>
</tr>
<tr>
<td>E10</td>
<td>100</td>
<td>4</td>
<td>102</td>
<td>33</td>
<td>2</td>
<td>32</td>
<td>P10 (not in null)</td>
<td>JT &gt; H &gt; K</td>
<td>25040</td>
<td>26146</td>
<td>3.54E+02</td>
<td>51392</td>
<td>optimal</td>
</tr>
<tr>
<td>E11</td>
<td>100</td>
<td>8</td>
<td>10</td>
<td>5</td>
<td>10</td>
<td>P11 (zero)</td>
<td>JT &gt; H &gt; K</td>
<td>121100</td>
<td>123329</td>
<td>9.09E-12</td>
<td>131139</td>
<td>optimal</td>
<td>0.00E+00</td>
</tr>
</tbody>
</table>
The estimating experiments E1, E3, E4, E6, and E11 with $\omega$ and $\xi$ generated inside the null space of $Z_\omega$ and $Z_\xi$ respectively, as expected, recover the exact value of $\beta^{\text{inc}}, \sigma_\beta^{\text{inc}}$, and $\phi^{\text{inc}}$. The MSE between the default parameters and the estimated parameters is 0 for those instances. Knowing that SNOPT solves the problem locally, we can only verify the global optimality of the instances obtaining a zero objective value. This is because 0 is a valid lower bound of (29), and the valid upper bound provided by the solver coincides with the valid lower bound in this case.

When obtaining an objective value 0, the estimated parameters push $\xi^T Z_\xi \Lambda_\xi Z_\xi^T \xi$ and $\omega^T Z_\omega \Lambda_\omega Z_\omega^T \omega$ to the smallest possible values. Among those problems with $\omega$ and $\xi$ not generated inside the null spaces, E2 and E5 still recover the exact incumbent parameters, and the optimal objective value is 0; E7 is reported infeasible (“infeasibilities minimized”) by the solver; E9 and E10 do not recover the incumbent parameters and have nonzero objective values. From viewing the MSE, we see that the discrepancies are from the estimated $\phi$. It is a reasonable result since the $\omega$ is not generated inside the null space of $Z_\omega$ and the constraint $mc_{jt} = y_{jt}^T \phi + \omega_{jt}$ prevents the possibility of pushing $\omega^T Z_\omega \Lambda_\omega Z_\omega^T \omega$ down to 0. On the other hand, for those problems having nonzero objective values, we do not have the information from the solver whether the solution is global optimum or not. The $\xi$ and $\omega$ solutions are relevant to the dimension of weight matrices. In instances with $J \times T = H$ (E1, E2, E6, E7, and E8), solutions $\xi$ and $\omega$ can only be 0 if the objective value is 0, while in the instances with $J \times T > H$ (E3, E4, and E5), solutions $\xi$ and $\omega$ can be nonzero even if the objective value is 0. The largest estimating problem we have tried is E11. It contains 100 consumers, 10 products in each market, 10 markets and 5 firms competing in each market. The solver successfully solved this QPCC with 121100 linear complementarity constraints and 123320 equalities after 13987 seconds.

The experiments with $H > J \times T > K$ are shown in Table 21. We try to enlarge the size of instrumental matrices, while the remaining information remains the same as in E6, E1, and E9. We say the instances with $H > J \times T > K$ are non-typical because we usually want the sample size $J \times T$ to be large. Meanwhile, if $H > J \times T$, this is the case where the $e_i$ obtained in Step 1 of Weight Matrix Generation procedure is negative. We need a special way to generate positive definite weight matrices.
Table 21: Results of the parameters estimation. $\alpha = 1$. A non-typical size of the instrumental variables: $H > J \times T > K$.

<table>
<thead>
<tr>
<th>Estimation Instances</th>
<th>SNOPT Solving Statistics</th>
<th>MSE</th>
<th>$\xi$ and $\nu$ Sol</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Special Handling</td>
<td>Obj Value</td>
<td>Time (sec)</td>
</tr>
<tr>
<td>E12</td>
<td>crash</td>
<td>8.93E-10</td>
<td>2.42</td>
</tr>
<tr>
<td>E13</td>
<td>same as E6; H &gt; JT &gt; K</td>
<td>8.40E-10</td>
<td>1.73</td>
</tr>
<tr>
<td>E14</td>
<td>same as E1; no SSD</td>
<td>3.64E-12</td>
<td>0.44</td>
</tr>
<tr>
<td>E15</td>
<td>same as E6; no SSD</td>
<td>152.4861</td>
<td>2.01</td>
</tr>
</tbody>
</table>

Table 22: Results of the parameters estimation. $\alpha = 1$. No SSD term in the objective function.

matrices as described in Step 2 of the procedure. In experiments, the solver crashed when solving E12. In E13, E14, and E15, there were basically no differences in the estimated parameters and the objective values from those of the original problems E6 or E1. In E16, the MSE in $\phi$ and the objective values are nonzero.

The runs E17-E20 shown in Table 22 are results of estimation problems which drop the SSD terms from the objective function. All E17-E20 were solved to global optimum since the objective values are 0, but the MSE between the default parameters and the estimated parameters are relatively large, provided that the maximum possible MSE of $\tilde{\beta}$, $\sigma_\beta$ and $\phi$ are 400, 100, and 4 respectively. This phenomenon suggests that there are multiple optimal parameters that satisfy the orthogonality in the framework of the generalized method of moment, and the range of these optimal parameters can be very wide. Therefore, imposing the incumbent parameters and the SSD terms to the objective function is significantly important in obtaining the confined parameters.
Table 23: Parameters in the instances of lowest-utility constrained Nash-Bertrand game and results.

**Data of lowest-utility constrained Nash-Bertrand game**

The instances of the lowest-utility constrained Nash-Bertrand game $LCP_{\text{LUNB}}$ (19) solved by PATH are summarized in Table 23, where the constants $\delta_f$ are chosen at $0.1 \times \delta_f$.

In Table 24, we summarize consumer purchasing behavior in the instances of Table 23. We notice that in these results, two situations are possible. The first situation is that when two products provide equal utility to a consumer, the consumer can have a “mixed strategy” in buying either one of the product, i.e., the probability of buying either one of them is somewhere between 0 and 1 and the sum is 1. The second situation is that if a product provides the utility 0 to a consumer, the consumer can have a mixed strategy for buying or not buying. Compared with the purchasing behavior in Table 19, the purchasing probabilities of a lowest-utility constrained pricing game is more balanced among each product. This improvement is due to both the algorithm behind the PATH solver and the formulation of the least-utility constraint. Without the least-utility constraint, the objective function (36) needs to be added to the LCP to avoid all-zero $\pi$; but PATH cannot be applied to solve an LPCC.

Following the results of the lowest-utility constrained pricing game, the $q_{jt}$ and $p_{jt}^{obs}$ deduced from MP1-MP5 define the estimation problem ME1-ME5. The results of solving ME1-ME5 by SNOPT are shown in Table 25. Among these estimation problems, only ME4 cannot recover the incumbent parameters because $\xi$ and $\omega$ are
CHAPTER 4.

| MP1: | consumer 1 buys product 4 in market 1 and buys nothing in market 2 and 3; consumer 2 buys product 1 in market 1 and 2 and buys product 4 in market 3; consumer 3 buys product 1 in market 1, has 96% of probability buys product 2 and 4% of probability buying product 3 in market 2, and buys product 3 in market 3; consumer 4 buys product 3 in market 1; consumer 5 buys product 3 in market 1 and 2, and buys product 1 in market 3. |
| MP2: | the same as MP1. |
| MP3: | consumer 1 buys product 1 in market 1 and 2, buys product 3 in market 5, and has 14% probability buying product 4 in market 4; consumer 2 buys product 1 in market 1 and 2, buys product 2 in market 5, and buys product 3 in market 3 and 4; consumer 3 buys product 1 in market 1, and buys nothing in other markets; consumer 4 buys product 1 in market 1 and 2, has 85% probability buying product 3 in market 4 and 15% probability buying product 4 in market 3; consumer 5 buys product 1 in market 3, 4, and 5, and buys product 3 in market 1 and 2; consumer 6 buys product 1 in market 1 and 2, and buys product 3 in market 3, 4, and 5; consumer 7 has the same behavior with consumer 5; consumer 8 has the same behavior with consumer 7; consumer 9 buys product 1 in market 1 and 2, buys product 3 in market 3 and 4, has 56% probability buying product 3 in market 5 and 44% probability buying product 4 in market 5; consumer 10 has the same behavior with consumer 8. |
| MP4: | no consumers buy product 2. consumer 1 buys product 1 in market 2, buys product 3 with probability 16% in market 1, with probability 92% in market 2, and with probability 100% in market 4, and has 84% probability buying product 4 in market 1; consumer 2 buys product 1 in market 2, buys product 3 in market 1, 3, 4, and 5; consumer 3 buys product 1 in market 2; consumer 4 has the same behavior with consumer 2; consumer 5 buys product 1 in market 3, 4, and 5, and buys product 3 in market 1 and 2; consumer 6 has the same behavior with consumer 4; consumer 7 has the same behavior with consumer 5; consumer 8 has the same behavior with consumer 7; consumer 9 buys product 1 in market 2, buys product 3 in market 1, 3 and 4, has 56% probability buying product 3 in market 5 and 44% probability buying product 4 in market 5; consumer 10 has the same behavior with consumer 8. |
| MP5: | consumer 1 buys product in market 1 and 2, buys product 3 in market 5, and has 7% probability buying product 4 in market 4; consumer 2 buys product 1 in market 2, has 44% probability buys product 2 in market 5, buys product 3 in market 1, has 85% probability buying product 3 in market 3, has 56% probability buying product 3 in market 5, and has 15% probability buying product 4 in market 3; consumer 3 buys nothing; consumer 4 buys product 1 in market 1 and 2, buys product 3 in market 3 and 5, has 85% probability buying product 3 in market 4 and 15% probability buying product 4 in market 4; consumer 5 buy product 1 in market 3, 4 and 5, and buys product 3 in market 1 and 2; consumer 6 buys product 1 in market 1 and 2, and buys product 3 in market 3, 4, and 5; consumer 7 has the same behavior with consumer 5; consumer 8 has the same behavior with consumer 7; consumer 9 buys product 1 in market 1 and 2, buys product 3 in market 3 and 4, buys product 4 in market 5; consumer 10 has the same behavior with consumer 8. |

Table 24: Consumers’ purchasing behavior in the instances of lowest-utility constrained Nash-Bertrand game.

not generated in the null space of $\mathbf{A}_\xi$ and $\mathbf{A}_\omega$ respectively. In general, the estimating model subject to the lowest-utility constrained Nash-Bertrand game does not change the trend of numerical performance from that in the original estimating model.
Estimation Instances SNOPT Solving Statistics MSE

Table 25: Results of the parameters estimation under lowest-utility constrained pricing game. \( \alpha = 1 \). The objective function minimizes the orthogonality in GMM and the sum of squared deviation of parameters.

Table 26: Results of parameters estimation with \( \alpha \) fixed at values different from the true one.

**One-dimensional grid search of \( \alpha \)**

As of the experiments demonstrated, we fixed the \( \alpha \) at the default value used to generate the game. We are interested in whether the results of estimation will be affected or not if the value of \( \alpha \) is fixed at a different value. The experiments shown in Table 26 were done on a chosen instance, ME4, because there is room for deducing the optimal objective value of ME4. Among ME6-E11, some \( \alpha \) values (ME6 and ME7) push the objective value down to zero, while other \( \alpha \) values (ME8-ME11) actually raise the objective value.
Chapter 4.

Simplified estimation

We also solved the simplified estimation problem (34), where the following bounds of parameters are imposed:

\[
\alpha^\min \leq \hat{\alpha} \leq \bar{\alpha}, \\
\frac{\beta}{\sigma^2} \leq \hat{\beta} \leq \bar{\beta}, \\
\sigma_{\beta} \leq \sigma_{\beta} \leq \bar{\sigma}_\beta. 
\]  

(39)  

In this set of results shown in Table 27, \( \phi \) is not involved, but \( \alpha \) is a parameter to be estimated. The MSEs of \( \beta \) and \( \sigma_{\beta} \) are computed as in (38), and

\[
\text{MSE of } \hat{\alpha} = (\hat{\alpha} - \alpha^{\text{inc}})^2. 
\]  

(40)  

The objective functions in S1-S8 minimize the orthogonality of GMM and the SSD of parameters, \( \xi^T Z \xi A \xi^T Z \xi + (\hat{\alpha}_k - \alpha_k^{\text{inc}})^2 + \sum_{k=1}^K \left[ (\hat{\beta}_k - \beta_k^{\text{inc}})^2 + (\sigma_{\beta_k} - \sigma_{\beta_k}^{\text{inc}})^2 \right]. \)

The estimated parameters recover the incumbent values in S1-S8. The objective functions in S9-S11 only minimize the orthogonality of GMM, \( \xi^T Z \xi A \xi^T Z \xi. \) Compared to the maximal possible MSE: 81, 400, and 4 respectively, the MSE on \( \hat{\alpha}, \hat{\beta}, \) and \( \sigma_{\beta} \) are relatively large in S9-S11. All the simplified model instances were solved to global optimum with an objective value 0. Since the advantage of solving the simplified estimation model is that no parameters are assumed known in advance, the estimated \( \hat{\alpha} \) obtained from the simplified model can be used as a complement for the \( \hat{\alpha} \) fixed in the estimation models \( \text{QPCC}_{E_sP, NB} \) and \( \text{QPCC}_{E_sP, LUNB}. \)

7 Discussion

We have shown that the estimation of the pure characteristics demand model (PCM) can be formulated as a mathematical program. In the framework of mathematical programming, there is great flexibility to consider the market level observed data when identifying the estimators of PCM. We embed a Nash-Bertrand game, the number of sold products, current products price and the distribution of parameters in the program, yielding a quadratic program with complementarity constraints (QPCC). Four issues arise with this method of estimation:
1. The solution to this QPCC is not unique. To enforce uniqueness, we have imposed the least sum square of deviation (SSD) criterion of an incumbent set of parameters on the estimation. Thus the need of an incumbent set of parameters becomes important. The availability of the incumbent parameters, however, can be a problem in reality.

2. We assume one of the estimators, $\alpha$, fixed, so the QPCC has only linear complementarity constraints. Thus the program can be solved by existing solvers including SNOPT and others. We did not address the problem of estimating $\alpha$ along. Two ways are suggested to compensate for this drawback. One is to do the one-dimensional grid-search for different fixed values of $\alpha$, and the second is to find an estimator $\bar{\alpha}$ from the simplified estimation model that does not contain the Nash-Bertrand game.

3. We use solver SNOPT to solve this QPCC without guaranteeing global optimality. Only when the objective value is 0 do we know that a global optimal estimator is found. The global optimization algorithms for QPCC are time-consuming.

4. We have proposed procedures for generating the instrumental and weight matrices that work perfectly for our game-simulated data, but the best selection of these matrices in reality can be very difficult.

Table 27: Results of the simplified parameters estimation.

<table>
<thead>
<tr>
<th>Name</th>
<th>Setting</th>
<th>Bounds for $\bar{\alpha}$</th>
<th>Instance Size Identified by the Solver</th>
<th>Objective Function</th>
<th>SNOPT Solving Statistics</th>
<th>MSE</th>
<th>$\xi$ and $\omega$ Sol.</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1</td>
<td>as in E1</td>
<td>[0.0001,10]</td>
<td>75</td>
<td>226</td>
<td>4.10E-16</td>
<td>0.06</td>
<td>326</td>
</tr>
<tr>
<td>S2</td>
<td>as in E2</td>
<td></td>
<td>75</td>
<td>226</td>
<td>3.13E-13</td>
<td>0.05</td>
<td>341</td>
</tr>
<tr>
<td>S3</td>
<td>as in E3</td>
<td></td>
<td>75</td>
<td>226</td>
<td>2.70E-16</td>
<td>0.05</td>
<td>341</td>
</tr>
<tr>
<td>S4</td>
<td>as in E4</td>
<td></td>
<td>500</td>
<td>5514</td>
<td>3.08E-11</td>
<td>0.67</td>
<td>281</td>
</tr>
<tr>
<td>S5</td>
<td>as in E5</td>
<td></td>
<td>250</td>
<td>400</td>
<td>1.22E-13</td>
<td>0.53</td>
<td>2685</td>
</tr>
<tr>
<td>S6</td>
<td>as in E6</td>
<td></td>
<td>75</td>
<td>226</td>
<td>1.61E-12</td>
<td>0.05</td>
<td>212</td>
</tr>
<tr>
<td>S7</td>
<td>as in E7</td>
<td></td>
<td>75</td>
<td>226</td>
<td>3.00E+00</td>
<td>12.31</td>
<td>3885</td>
</tr>
<tr>
<td>S8</td>
<td>as in E8</td>
<td></td>
<td>75</td>
<td>226</td>
<td>2.95E-12</td>
<td>0.17</td>
<td>602</td>
</tr>
<tr>
<td>S9</td>
<td>as in E17</td>
<td>[0.0001,10]</td>
<td>500</td>
<td>5514</td>
<td>0.00E+00</td>
<td>1.00</td>
<td>53.90</td>
</tr>
<tr>
<td>S10</td>
<td>as in E18</td>
<td></td>
<td>75</td>
<td>226</td>
<td>1.61E-12</td>
<td>0.05</td>
<td>212</td>
</tr>
<tr>
<td>S11</td>
<td>as in E19</td>
<td></td>
<td>250</td>
<td>400</td>
<td>2.95E-12</td>
<td>0.17</td>
<td>602</td>
</tr>
</tbody>
</table>
In the numerical simulation, SNOPT is able to solve the problem efficiently. The exact parameters can be recovered if the instrumental and weight matrices of the generalized method of moment (GMM) is properly defined, and the SSD term is added to the objective function of the QPCC.
Chapter 5: Future Directions

This chapter aims to explicitly point out the weakness of the research presented in Chapters 2, 3 and 4. Note that in the last sections of these chapters, we have summarized and discussed the pros and cons of the main models and algorithms separately. We refer the readers to these concluding sections as areas for improvement within the framework of the proposed methodologies in each chapter. The issue we address here, however, is more substantial and can be considered as possible future research. These new problems emerge when we validate the algorithms and models through the numerical experiments.

The domain-partitioning algorithm proposed in Chapter 2 performs fairly well in random generated instances of the bi-parametric linear program with linear complementarity (LPCC) when the size of complementarity constraints is not greater than 300. Starting the iteration with a small enough initial approximation error controllable by a pair of user-chosen scalers \((s, t)\), this algorithm converges in all the instances randomly generated by us. The algorithm converges when the gap between the lowest upper bound and the largest lower bound is below a threshold; thus the solutions obtained are global optimal. Unfortunately, the solution techniques we have proposed for this algorithm are not effective enough to solve the bi-parametric LPCC with a special structure of the Support Vector Machine Regression (SVR) parameter selection (See Section 1.1 in Chapter 2). We believe
that the extra variable $u$ in equation (11) of Chapter 2:

$$\begin{align*}
\min_{w,x,y,u} & \quad c^T x + d^T y + g^T u \\
\text{subject to} & \quad A x + B y + E u \geq f, \\
& \quad y \perp w := N x + M y + q, \\
& \quad y_j \geq 0, \forall j = 1, \ldots, 3n_d, \\
& \quad w_j \geq 0, \forall j = 1, \ldots, 3n_d, \\
& \quad y_{3n_d+1} : f r e e, \\
& \quad w_{3n_d+1} = 0,
\end{align*}$$

which is involved in the side constraints but not in the complementarity constraints, has destroyed the simple structure of the model without $u$. Variable $u$ is introduced to linearize the absolute outer objective function. As a result, the variables involved in the complementarity, $x$ and $y$, are completely absent in the linearized objective function. The branch-and-bound process of our algorithm reduces the complementarity approximation error along with the increase of the number of iterations, but the objective value consisting of $u$, involved only in the side constraints, is hardly improved. The domain-partitioning algorithm designed for solving a problem without $u$ fails to extend directly to the model with $u$. For a model as special as the SVR parameter selection model (the equation (11), Chapter 2), other solution techniques need to be developed.

On the other hand, the rectangle search algorithm developed for solving the cross-validated SVR parameter selection in Chapter 3 indeed solved many instances of the synthetic and real-world data to global optimum. This algorithm terminates when every invariancy region (See Definition 7 in Chapter 3) in the parameter space has been found. Compared to the strategies used in [113] and [52] for searching invariancy regions, our algorithm distinguishes itself by always maintaining the search area as a rectangle and searching along the boundaries of the rectangles. We have already mentioned in the last section of Chapter 3 that one crucial improvement for the rectangle search would be to derive additional sufficient conditions so the realization of the invariancy regions inside a rectangle can be claimed. Other than this, we have also found that complementarity degeneracy has caused many trou-
When the solution to a fixed \((C_\epsilon, \epsilon_\epsilon)\) lower-level problem obtained by the semismooth method is degenerate, say

\[
\{y_{d_{j'}} = (x^j_\alpha)^T w_s + b_s - \epsilon_\epsilon, \alpha_{j'} = 0, \beta_{j'} = 0\}, \text{ for some index } j'
\]

there is actually more than one grouping vector associated with this degenerate solution. Should we take care of only one of the associated groupings or every possible groupings? If we take care of only one of the associated groupings, do we know whether the other groupings are retrieved again in the search afterwards? If we take care of all the groupings associated with one degenerate solution, how to enumerate them? Is the process time consuming? The answers to these questions are lacking in the current algorithm and are worth further research, based on which some smart techniques could be developed to accommodate the degeneracy.

The multi-parametric mathematical program with nonlinear complementarity constraints (MPNCC) developed in Chapter 4, the pure characteristics demand model estimation, has not yet been solved to global optimum. Are there any solution techniques being employed in Chapters 2 and 3 that can be again used to develop a global optimization algorithm for this multi-parametric MPNCC? This topic would be a challenge because the parameter space is no longer two dimensional, and the complementarity constraints are nonlinear. Besides these new algorithmic directions, to further validate the effectiveness of this estimation model, work extending it to a specific type of real-world product in the market is desperately needed.

This thesis presents models and methods for optimal parameter selections in the specific model type of the parametric MPCC. We believe that there are abundant source problems of model/parameter selection form many fields of application that have not been introduced in this thesis.
Bibliography


