SOLVING SUPPORT VECTOR MACHINE CLASSIFICATION PROBLEMS AND THEIR APPLICATIONS TO SUPPLIER SELECTION

by

GITAE KIM

B.S., Hanyang University, 1998
M.S., Seoul National University, 2000

AN ABSTRACT OF A DISSERTATION

submitted in partial fulfillment of the requirements for the degree

DOCTOR OF PHILOSOPHY

Department Of Industrial & Manufacturing Systems Engineering
College of Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

2011

Abstract

Recently, interdisciplinary (management, engineering, science, and economics) collaboration research has been growing to achieve the synergy and to reinforce the weakness of each discipline. Along this trend, this research combines three topics: mathematical programming, data mining, and supply chain management. A new pegging algorithm is developed for solving the continuous nonlinear knapsack problem. An efficient solving approach is proposed for solving the ν-support vector machine for classification problem in the field of data mining. The new pegging algorithm is used to solve the subproblem of the support vector machine problem. For the supply chain management, this research proposes an efficient integrated solving approach for the supplier selection problem. The support vector machine is applied to solve the problem of selecting potential supplies in the procedure of the integrated solving approach.

In the first part of this research, a new pegging algorithm solves the continuous nonlinear knapsack problem with box constraints. The problem is to minimize a convex and differentiable nonlinear function with one equality constraint and box constraints. Pegging algorithm needs to calculate primal variables to check bounds on variables at each iteration, which frequently is a time-consuming task. The newly proposed dual bound algorithm checks the bounds of Lagrange multipliers without calculating primal variables explicitly at each iteration. In addition, the calculation of the dual solution at each iteration can be reduced by a proposed new method for updating the solution.

In the second part, this research proposes several streamlined solution procedures of ν-support vector machine for the classification. The main solving procedure is the matrix splitting method. The proposed method in this research is a specified matrix splitting method combined with the gradient projection method, line search technique, and the incomplete Cholesky decomposition method. The method proposed can use a variety of methods for line search and parameter updating. Moreover, large scale problems are solved with the incomplete Cholesky decomposition and some efficient implementation techniques.

To apply the research findings in real-world problems, this research developed an efficient integrated approach for supplier selection problems using the support vector machine and the mixed integer programming. Supplier selection is an essential step in the procurement

processes. For companies considering maximizing their profits and reducing costs, supplier selection requires seeking satisfactory suppliers and allocating proper orders to the selected suppliers. In the early stage of supplier selection, a company can use the support vector machine classification to choose potential qualified suppliers using specific criteria. However, the company may not need to purchase from all qualified suppliers. Once the company determines the amount of raw materials and components to purchase, the company then selects final suppliers from which to order optimal order quantities at the final stage of the process. Mixed integer programming model is then used to determine final suppliers and allocates optimal orders at this stage.

SOLVING SUPPORT VECTOR MACHINE CLASSIFICATION PROBLEMS AND THEIR APPLICATIONS TO SUPPLIER SELECTION

by

GITAE KIM

B.A., Hanyang University, 1998
M.S., Seoul National University, 2000

A DISSERTATION

submitted in partial fulfillment of the requirements for the degree

DOCTOR OF PHILOSOPHY

Department of Industrial & Manuafacturing Systems Engineering
College of Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

2011

Approved by:

Major Professor
Chih-Hang (John) Wu

Abstract

Recently, interdisciplinary (management, engineering, science, and economics) collaboration research has been growing to achieve the synergy and to reinforce the weakness of each discipline. Along this trend, this research combines three topics: mathematical programming, data mining, and supply chain management. A new pegging algorithm is developed for solving the continuous nonlinear knapsack problem. An efficient solving approach is proposed for solving the ν-support vector machine for classification problem in the field of data mining. The new pegging algorithm is used to solve the subproblem of the support vector machine problem. For the supply chain management, this research proposes an efficient integrated solving approach for the supplier selection problem. The support vector machine is applied to solve the problem of selecting potential supplies in the procedure of the integrated solving approach.

In the first part of this research, a new pegging algorithm solves the continuous nonlinear knapsack problem with box constraints. The problem is to minimize a convex and differentiable nonlinear function with one equality constraint and box constraints. Pegging algorithm needs to calculate primal variables to check bounds on variables at each iteration, which frequently is a time-consuming task. The newly proposed dual bound algorithm checks the bounds of Lagrange multipliers without calculating primal variables explicitly at each iteration. In addition, the calculation of the dual solution at each iteration can be reduced by a proposed new method for updating the solution.

In the second part, this research proposes several streamlined solution procedures of ν-support vector machine for the classification. The main solving procedure is the matrix splitting method. The proposed method in this research is a specified matrix splitting method combined with the gradient projection method, line search technique, and the incomplete Cholesky decomposition method. The method proposed can use a variety of methods for line search and parameter updating. Moreover, large scale problems are solved with the incomplete Cholesky decomposition and some efficient implementation techniques.

To apply the research findings in real-world problems, this research developed an efficient integrated approach for supplier selection problems using the support vector machine and the mixed integer programming. Supplier selection is an essential step in the procurement

processes. For companies considering maximizing their profits and reducing costs, supplier selection requires seeking satisfactory suppliers and allocating proper orders to the selected suppliers. In the early stage of supplier selection, a company can use the support vector machine classification to choose potential qualified suppliers using specific criteria. However, the company may not need to purchase from all qualified suppliers. Once the company determines the amount of raw materials and components to purchase, the company then selects final suppliers from which to order optimal order quantities at the final stage of the process. Mixed integer programming model is then used to determine final suppliers and allocates optimal orders at this stage.

Table of Contents

List of Figures

List of Tables

Acknowledgements

I would like to express my deep-felt and sincere gratitude to my advisor, Dr. John Wu. Throughout my research in PhD program at Kansas State University, he provided sound advice, encouragement, thoughtful guidance, lots of good ideas. His broad knowledge and logical way of thinking have been of great value for me. I could not have done my dissertation without his help.

I am deeply grateful to my other advisory committee members: Dr. Todd Easton, Dr. B. Terry Beck, and Dr. Chwen Sheu, for their detailed and constructive comments. I wish to specially thank to Dr. Orlen Grunewald for being my final examination outside chairperson.

I wish to warmly thank Dr. Bradley A. Kramer, Dr. E. Stanley Lee, Dr. David Ben-Arieh, Dr. Shing I. Chang for their valuable encouragement and comments.

I am grateful to the department of Industrial & Manufacturing Systems Engineering for giving me the opportunities, resources, and financial support to finish my research.

I owe my most loving thanks to my wife Mijung Lee, my daughters Youngji and Youngsuh for their loving support, encouragement, and understanding during this work. I also deeply thank my families in Korea for their constant support, encouragement, and advice.

Dedication

To my loving family,

CHAPTER 1 - Introduction

1.1 Introduction

In real world applications, many decision problems have to be solved in daily operations. Among them, classification is one of important class of decision problems. One may need to classify the things that they did for a day as value added and non-value added tasks. Marketing team may classify the company's markets into several different tiers or segments using pre-defined criteria or performance metrics. Companies may classify their clients into important customers and normal ones in terms of value added contributions to the company's revenue. Companies may classify their suppliers into qualified or potential supplier groups based on various supplier evaluation criteria. Countries may classify other countries into friendly-nations (allies) and the others. When one makes a decision for any classification problem, intuitive solutions to any classification problem are easy and simple. However, such solutions are sometimes wrong because the decisions are too subjective. To avoid this happening, researchers have suggested a variety of systematic methods for making decisions about problems in classification.

One good scientific method for classification problems is the machine learning method in the field of data mining (Vapnik, 1995). The classification problem is sometimes referred as the pattern recognition problem. The support vector machine is a machine learning tool for regression and classification problems. This dissertation focuses on the development of a solution algorithm for the support vector machine (SVM) and its applications. Compared with other statistical methods, the SVM does not require any parameters. Thus, the SVM is sometimes called a non-parametric method. Moreover, the SVM can handle large-scale problems. Before the SVM methods were proposed in 1995, machine learning method using neural networks are the popular approaches for attacking classification problems. The neural network had two main drawbacks of the generalization and the slow convergence because the performance of the neural networks is data dependent and the method consumes a lot of memory and processing time to run. The SVM has overcome these drawbacks in both theoretical and practical aspects.

In this research, an efficient solution approach for the SVM is proposed using symmetric kernel method. The method consists of the matrix splitting method, the gradient projection method, and the incomplete Cholesky decomposition. The proposed method enables us to use several options for both the line search and updating parameters.

In addition, this research proposes a solution algorithm for the subproblem of the SVM and suggests an efficient solution procedure of the supplier selection problem using the SVM. In solving a classification problem using SVM, a quadratic knapsack subproblem needs to be solved repeatedly and they are frequently the most time consuming task in the solution processes. A new pegging algorithm is proposed for solving the nonlinear knapsack subporblem arising in the SVM. This newly proposed method for solving the continuous nonlinear knapsack problem can significantly reduce the time consuming steps in the solution processes.

For applications to the classification problems using SVM, an efficient integrated solution method is suggested for the supplier selection problem. The selection of qualified suppliers is an important issue for many companies because it directly affects the quality of products as well as the potential profits. The SVM can classify suppliers into two groups such as the qualified suppliers and the potential suppliers. The final suppliers out of potential suppliers are then selected based on other considerations such as the requirements products, due dates, consolidations, and final costs. In section 2, the motivation of this research is described. The contribution of this dissertation is presented in section 3. An overview of this dissertation is in section 4.

1.2 Research Motivations

The motivation of this research started with the technical issues that arise in the SVM. The SVM has been a popular machine learning method for about fifteen years now since many studies (Bennett and Bredensteiner (1997), Suykens and Vandewalle (1999), Joachims (1999), Platt (1999), Crisp and Burges (2000), Bennett and Bredensteiner (2000), Lee and Mangasarian (2001), To et al. (2001), Zhou et al. (2002), Zhan and Shen (2005), Bach and Jordan (2005), Kianmehr and Alhajj (2006), Mavroforakis and Theodoridis (2006), An et al. (2007), Alzate and Suykens (2008)) have proved that the SVM is an efficient method both theoretically and practically. Given its popularity, three types of major studies have focused on its use: formulation, solving algorithm, and applications. One recent formulation of the SVM is the ν-SVM, which is less sensitive to a regularization parameter C than C-SVM (Nehate, 2006). This research focused on the ν-SVM. Most solving algorithms for the SVM have been proposed by using C-SVM formulation. For large-scale data sets, the working set or the sequential minimal optimization (SMO) type method is the only one that can solve problems. Other methods have worked with small to medium sized problems. This motivated us to develop an efficient solving algorithm for the ν-SVM for the typically large-scale classification problem using non-SMO type methods. The advantage of the non-SMO type method is to use more algorithms that have been already developed for quadratic or nonlinear programming problems. Some attempts have used the non-SMO type method, but the size of the data set has been a major limitation. Therefore, this research proposes a non-SMO type solution method for the ν-SVM classification problem.

To solve the ν-SVM classification problem, a quadratic programming problem must be solved. This problem has a quadratic objective function with a dense Hessian matrix, a single linear constraint, and box constraints. The proposed solving algorithm in my research is an iterative method. At each iteration, a subproblem with continuous quadratic objective function and knapsack constraint needs to be solved which frequently is the most time consuming step in the solution processes. If one can improve the algorithm to solve the continuous quadratic knapsack problem, the SVM problem itself can be solved more efficiently. This motivates the development of a new pegging algorithm for the continuous nonlinear knapsack problem. The Bitran-Hax (1981) algorithm is the pegging algorithm used for the continuous nonlinear knapsack problem. However, this research found that the Bitran-Hax algorithm has two time consuming calculations at each iteration. First is the recalculation of the primal solutions after each dual iteration and check their feasibilities. The other time consuming calculation is to calculate the dual solution at each iteration. These two challenging issues are the main motivation for the development of the dual bound pegging algorithm.

This research finally focuses on applying SVM. Many applications exist for the ν-SVM classification problem. This research focuses on the supplier selection problem in supply chain management (SCM). If a company selects non-qualified suppliers, then the quality of the product could be out-of-controlled and the on time deliveries may not be fulfilled at the desired level. This could very likely have significant impact the company's ultimate profit and reputation. Therefore, the supplier selection problem is an important issue in many companies. If a company can find a more efficient and accurate method for selecting suppliers, then the company could easily gain more profits or market shares, which are the goals of most for-profit companies. This issue motivated us to propose an efficient solution approach for the supplier selection problem. It is well-known that the supervised machine learning method is frequently more accurate and more efficient than the unsupervised method through existing literatures. This research used the SVM as a supervised machine learning method for selecting potential suppliers from all suppliers considered. In case of that there exist the past data for the supplier selection of a company, the SVM can solve the supplier selection problem better than unsupervised method. Another application of SVM to financial area is the company credit ratings. Credit rating is a very important factor for investing or loan to companies and also significant measurement of the company. This research applies the newly proposed SVM algorithm to predict credit ratings of companies in Korea.

1.3 Research Contributions

This research combines three major topics: mathematical programming, data mining, and supply chain management. Therefore, the contributions of the research range from theoretical to practical. One of the theoretical contributions is the development of a new pegging algorithm for solving the continuous nonlinear knapsack problem. Another theoretical contribution is to propose an efficient solving method for the ν-SVM classification problem. The practical contribution in this research is to suggest a new integrated solving approach for the supplier selection problem.

The Bitran-Hax algorithm is a famous pegging algorithm for the continuous nonlinear knapsack problem. The algorithm is known to be simple and fast. This research aims to improve on the Bitran-Hax algorithm. This research found that the algorithm does two time consuming calculations at each iteration. These two tasks are the calculation of a dual solution and those of the primal variables. The dual solution is calculated by the summation of the gradient of the functions at each iteration. The new algorithm splits this calculation and reduces re-calculations. Calculating primal variables is simple, but must be done for all free variables at each iteration. The new algorithm uses a dual variable instead of all free primal variables. The reason the Bitran-Hax algorithm calculates primal variables at each iteration is to check the feasibility of each variable. The main idea of the new algorithm is to use the bounds of the dual variable instead of the primal variables to check the feasibilities. The contributions of solving the continuous nonlinear knapsack problem are as follows.

- This research developed a new pegging algorithm for the continuous nonlinear knapsack problem.
- The new algorithm has two advantages: it removes the calculations of primal variables at each iteration and updates the dual solution instead of re-calculating.
- The solution time of the new algorithm is overall faster than the Bitran-Hax algorithm.
- The new algorithm is faster when the size of the problem is large.

The ν-SVM classification problem has a quadratic objective function, two linear constraints, and box constraints. First, this research gets one of linear constraints on the objective function using the augmented Lagrangian method. Then the problem becomes a singly linearly constrained quadratic convex programming problem. The Hessian matrix is a dense positive semi-definite matrix. The proposed method in this research splits the dense positive semi-definite Hessian matrix as the sum of two matrices. The algorithm solves a subproblem with a simple diagonal Hessian matrix, one of these two matrices, and can choose the Hessian matrix for the subproblem with any simple and nonsingular matrix. The subproblem is a continuous quadratic knapsack problem and can be solved by the new method proposed in this research. The current solution and the solution of the subproblem are used for calculating the direction vector. In the next step, the line search is conducted to find the best step size. Then the algorithm updates solutions and a parameter. In the line search and updating parameter steps, the algorithm can take advantage of several options like monotone or nonmonotone line search for the line search and Barzilai & Borwein (BB) (1988) rule for updating a parameter. Even if the algorithm splits the Hessian matrix, there are a few steps to calculate the Hessian matrix. To facilitate the calculation, the incomplete Cholesky decomposition method is applied to decompose the Hessian matrix if kernel method is applied. The contributions for the ν-SVM classification problem can be summarized as follows:

- This research proposes a different approach for solving ν-SVM classification.
- The method is a combination of matrix splitting, gradient projection, and incomplete Cholesky decomposition.
- The subproblem can be solved with an efficient solving algorithm such as dual bound algorithm.
- The algorithm can use a variety of combinations of methods for line search and parameter updating.

The supplier selection problem is an important issue for a company purchasing raw materials or some components for production from other companies. In the supplier selection problem, three major issues contribute the decision: the definition of criteria, the quantification of the criteria, and the selection method for potential suppliers and final suppliers. This research focuses on the method for selecting suppliers. Most methods for selecting suppliers do not need historical data for the selection. These methods use and check only the suppliers currently considered. This method is the unsupervised method. On the other hand, if a company has historical data for selecting suppliers, a supervised method like the SVM can be used. It is well known the supervised method is more accurate than the unsupervised method. The main idea of this research is to use SVM classification for selecting potential suppliers. The potential supplier denotes a supplier eligible to contract with a company. The company selects the final suppliers out of all potential suppliers. The potential suppliers can be considered the candidates for final suppliers. Selecting potential suppliers is a classification problem, and the SVM classification can be applied. To select the final suppliers, this research used a mixed integer programming model. The summary of the contribution of the supplier selection problem is as follows:

- This research suggests an integrated solving approach for the supplier selection problem.
- The SVM classification is used to select potential suppliers.
- A supervised method like the SVM is more accurate than an unsupervised method.
- The proposed method is simpler than other methods.

In the last part of this research, it applies the proposed method of SVM to a financial problem. Banks or investment companies want to measure eligible companies with appropriate and objective criteria. One well-known measurement is the company credit rating. The ratings are A through D. This research classifies companies into qualified company and others using SVM. Experimental results show that the newly proposed SVM solution method is good potential method for predicting the company credit ratings.

1.4 Dissertation Overview

In Chapter 2 of this dissertation, it describes the development of a new pegging algorithm for the continuous nonlinear knapsack problem introducing the new concept of the dual bound. The algorithm is compared with the Bitran-Hax algorithm because the new algorithm is an extension of the Bitran-Hax algorithm. Experimental results are shown as well.

In Chapter 3, the solution method for ν-SVM classification problem is presented. This research describes the history of SVM problems first and solving algorithms. Then, it proposes a method consisting of several mathematical methods. The incomplete Cholesky decomposition method and an efficient data storage method for a large scale lower triangular matrix are introduced. Some options for the line search and the parameter updating method are shown with some experimental results.

In Chapter 4, the supplier selection problem is described. The integrated solving approach for the supplier selection problem combines the SVM classification method with mathematical programming model for selecting suppliers. To compare with the SVM classification method, the analytic hierarchy process (AHP) for selecting potential suppliers is described. A mixed integer programming model is presented for selecting the final suppliers from the potential suppliers.

In Chapter 5, an application of financial problem using SVM is presented with the prediction of company credit ratings with real company data. The overall conclusion and future work of this research will be in the last Chapter 6.

CHAPTER 2 - Continuous Nonlinear Knapsack Problem

In this chapter, this research proposes an efficient pegging algorithm for solving continuous nonlinear knapsack problems with box constraints. The problem is to minimize a convex and differentiable nonlinear function with one equality constraint and bounds on the variables. One of the main approaches for solving this problem is the variable pegging method. The Bitran-Hax algorithm is a well-known pegging algorithm that has been shown to be a preferred choice especially when dealing with large-scale problems. However, it needs to calculate an optimal dual variable and update all free primal variables at each iteration, which frequently is the most time-consuming task. This research proposed a Dual Bound algorithm that checks the box constraints implicitly using the bounds on the Lagrange multiplier without explicitly calculating primal variables at each iteration and updating the dual solution in a more efficient manner. The results from the computational experiments have shown that the proposed new algorithm constantly outperforms the Bitran-Hax algorithm in all the baseline testing and two real-time application models. The proposed algorithm shows significant potentials to be used in practice for many other mathematical models in real-world applications with straight-forward extensions.

2.1 Introduction

The knapsack problem, also known as the resource allocation problem, is that a hitchhiker wants to pack his knapsack by selecting from among various possible objects those which give him maximum comfort, which can be formulated by a mathematical model with the objective function is to maximize the total comfort, one knapsack constraint of the capacity of knapsack, and binary variables which are defined in Martello and Toth, (1980).

If its objective function is nonlinear, then the problem is a nonlinear knapsack problem. There are some classes of the nonlinear knapsack problem and the review of these can be found in Bretthauer and Shetty, (2002). This research is interested in the problem which has a convex, differentiable, and nonlinear objective function, and box constraints for all variables, which is a convex, separable, and continuous type of problem. There are many applications for problems of this type which is described in Robinson et al. (1992) such as portfolio selection problem in Markowitz (1952), multi-commodity network flow problem Ali et al. (1980), transportation problem in Ohuchi and Kaji (1984), support vector machine in Nehate (2006), production planning in Tamir (1980), and convex quadratic programming in Dussault et al. (1986). The problem also can be considered as a subproblem for many optimization models. Ibaraki and Katoh (1988) discussed comprehensively the algorithmic aspects of resource allocation problem and its variants in their book. Twenty years later, Patriksson (2008) surveyed the history and applications of the problem as well as solving algorithms. Therefore these literatures are not reviewed here.

This research considers a continuous nonlinear knapsack problem with box constraints as follows.

(P1) $$Min \quad \sum_{i=1}^{n} f_i(x_i) \tag{2.1}$$

$$s.t. \quad \sum_{i=1}^{n} g_i(x_i) = b \tag{2.2}$$

$$l_i \le x_i \le u_i, \quad i = 1, \dots, n \tag{2.3}$$

where $f_i(x_i)$ is a nonlinear, convex, and differentiable function, $g_i(x_i)$ is linear referred as the knapsack constraint in the rest of the thesis, $x_i \in R^n$, $l_i \in R^n, u_i \in R^n$ for all $i = 1, \dots, n$, $b \in R^1$ and this research assumes all coefficients of $g_i(x_i)$ are not zero and $\sum_{i=1}^{n} g_i(l_i) \le b \le \sum_{i=1}^{n} g_i(u_i)$ and $f_i'(x_i)$ is invertible.

The Lagrangian dual formulation of P1 by relaxing the knapsack constraint (2.2) is as follows.

(D1) $$Max \quad \pi(\alpha) \tag{2.4}$$

$$\text{where } \pi(\alpha) = Min \quad \sum_{i=1}^{n} f_i(x_i) + \alpha\left(\sum_{i=1}^{n} g_i(x_i) - b\right) \tag{2.5}$$

$$s.t. \quad l_i \le x_i \le u_i, \quad i = 1, \dots, n \tag{2.6}$$

and $\alpha \in R^1$ is the Lagrange multiplier corresponding to the knapsack constraint (2.2).

The nonlinear knapsack problem P1 is frequently solved via an iterative manner. There are more than a handful of algorithms proposed to solve this problem and they can be generally divided into two main categories (Patriksson, 2008): the Lagrange multiplier search method and the variable pegging method. The basic ideas of these two methods are in the following pictures.

Figure 2.1 Lagrange multiplier search method

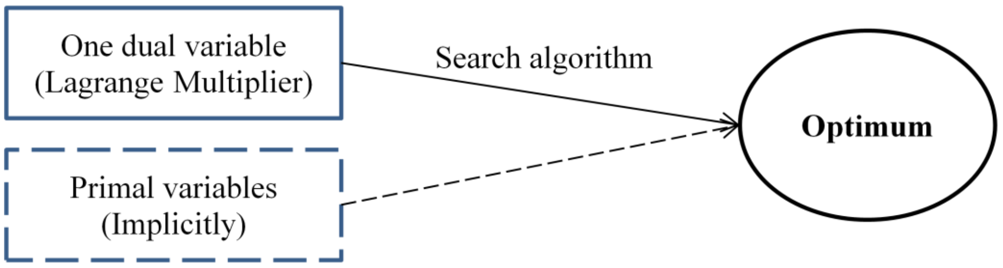
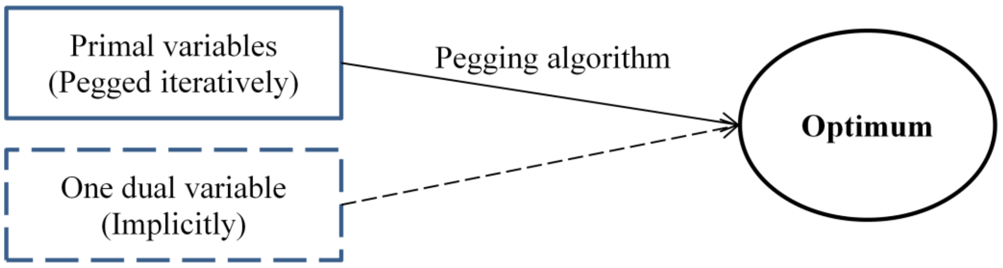


Figure 2.2 Variable pegging method



In Figure 2.1 and 2.2, solid boxes denote a variable or variables explicitly used to find the optimal solution in each algorithm and dashed boxes represent a variable or variables implicitly optimized as the other variable or variables are explicitly optimized. As Bretthauer and Shetty (2002) mentioned in their paper, while the Lagrange multiplier search method maintains all Karush-Kuhn-Tucker (KKT) conditions during its iterations, except the one knapsack constraint and its corresponding complementary slackness condition, the pegging method maintains all KKT conditions during its iterations except box constraints. That means the multiplier search method focuses on one dual variable achieving the optimal point, and the pegging method aims to find the optimal solution satisfying the feasibility of all primal variables. In Figure 2.1, the Lagrange multiplier search method uses the dual variable to find the optimal dual solution using a search algorithm because there is only one dual variable in the dual problem D1 which is described in Bazaraa et. al (1993) and Martello & Toth (1990). As the dual variable is optimized, the corresponding primal variables are implicitly optimized as well. On the other hand, in Figure 2.2, the variable pegging method is a type of primal algorithm finds the optimal primal solution by pegging some variables to their lower or upper bounds as their optimal value each iteration. In this algorithm, the dual variable can be also optimized implicitly as well. In his literature review, Patriksson (2008) did not make clear which approach was better in terms of computational complexity or average solution time. The biggest drawback of the pegging algorithm is that the relaxed problem should have an optimal solution and its efficiency depends on whether the optimal solution of the relaxed problem can be obtained in closed form. However, the pegging algorithm requires the objective function to be convex at least for the linear explicit constraints convergence of the method while the multiplier search method requires the objective function to be strictly convex (Patriksson 2008). In addition, Bretthauer et al. (2003) mentioned that the pegging algorithm was typically faster than the multiplier search algorithm when the relaxed subproblem can be solved in closed form. This research focuses on the pegging method since it has nicer finite convergence properties and has good potential to be streamlined for great performances.

The pegging method utilized a relaxed problem of P1 by ignoring the box constraints in (2.3), which has an optimal solution and the corresponding Lagrange multiplier can be solved in a closed form. This relaxed problem can be used to develop an efficient procedure to improve the solution efficiency. In addition, the pegging method generally guarantees a finite convergence. One of the well-known pegging methods is the Bitran-Hax algorithm (1981). Bitran and Hax (1981) developed the algorithm for solving continuous knapsack problem with a convex separable objective function and the coefficients of the equality constraints in (2.2) are ones. The Bitran-Hax algorithm has some very attractive features including the excellent convergence behavior, easy to implement and generally very efficient. There have been many extensions of this algorithm and the reviews of them are referred to (Patriksson, 2008).

In their research, Bitran and Hax (1981) introduced various resource allocation problems that could be formulated with this type of P1 problem. Patriksson (2008) referred to many extensions of this algorithm and reviews them.

Cottle at al. (1986) applied this algorithm to the constrained matrix problem. Eu (1991) formulated the sampling resource allocation problem with the nonlinear knapsack problem and solved the problem with the Bitran-Hax algorithm. Bretthauer et al. (1999) also studied the stratified sampling problem with integer variables, which was solved by the branch and bound algorithm for the main problem and the variable pegging algorithm for its subproblem.

Extensions have been applied to more general problems. Ventura (1991) extended the Bitran-Hax algorithm to a problem with non-unit coefficients in the knapsack constraint. Kodialam and Luss (1998) developed the algorithm to solve a nonlinear knapsack problem with non-negativity constraints on variables and a nonlinear convex knapsack constraint. The RELAX algorithm they proposed only checks the lower bound for pegging. Bretthauer and Shetty (2002) proposed a pegging branch and bound algorithm for more general problems with integer variables and a nonlinear convex objective function and knapsack constraint. Bretthauer et al. (2003) extended the pegging branch and bound algorithm to problems with additional block diagonal constraints.

Using the relationship between the restricted projection problem and the nonlinear knapsack problem, a projected pegging algorithm has been proposed. Robinson et al. (1992) introduced a pegging algorithm incorporated with the restricted projection method. If the objective function is the sum of squared variables and there are one linear knapsack constraint and box constraints, then the problem is equivalent to a problem finding the orthogonal projection of the origin on the feasible region. Robinson et al. (1992) used this projection method in the pegging algorithm to calculate the primal solutions of a relaxed problem with only the knapsack constraint and then checked the box constraints. Stefanov (2004) considered a problem with the objective function as a sum of squared subtraction of two variables and also used the concept of projection in the pegging algorithm and extended the algorithm that considered the case of some zero coefficients in the knapsack constraint.

As the literature shows, most extensions of the pegging algorithm focused on improving the calculation of the primal solution of the relaxed problem. On the other hand, the new algorithm in this research does not use the primal solutions of the relaxed problem, using instead the dual bound to check bound constraints.

This research presents another extension of the Bitran-Hax algorithm. Based on the preliminary computational experiments, this research discovered the efficiency of the Bitran-Hax algorithm suffers from two time consuming tasks. Firstly, in the Bitran-Hax algorithm, all primal free variables have to be recalculated at each iteration, where a free variable means the unpegged variable. Secondly, in the Bitran-Hax algorithm, the dual variable, α, must be searched and reevaluated several times to determine its optimal value. These are usually the two most time consuming procedures in the algorithm and they are the main motivations of this research. In the Bitran-Hax, the algorithm checks the feasibility of the solution after it solves the relaxed problem by ignoring the box constraints in (2.3) at each iteration. Solving the relaxed problem is the calculation of a newly trial dual variable at each iteration. Then, the algorithm calculates the primal variables again and then rechecks their box constraints for the feasibility. Basically, the newly proposed algorithm establishes a set of bounds as the predicted range of the optimal dual solution, which can be defined initially using only the input data. This is then used as the criterion for feasibility instead of checking the feasibility of the primal variables at each iteration. The dual bound is calculated only once initially and need not be updated during the solution process. For calculating the optimal dual solution, the new algorithm divides the calculations of the dual variables into several smaller components, and just updating the required components during the pegging process.

The objective of this chapter is to develop a new pegging algorithm based on the concepts of the Bitran-Hax algorithm to solve the continuous separable nonlinear knapsack problem with one linear equality knapsack constraint and box constraints. In the newly proposed algorithm, the new algorithm introduces the concept of the dual bound and how the dual bound can speed up the solution process. The computational results on randomly generated problems and applications embedded test models show that the new algorithm consistently outperforms the Bitran-Hax algorithm.

In the rest of the chapter, the concept of the Bitran-Hax algorithm as a current state-of-the-art pegging method is described in section 2. The new pegging algorithm for continuous nonlinear knapsack problem is presented in section 3. The new algorithm applies to the continuous quadratic knapsack problem as a special case of the nonlinear knapsack problem and the experimental results are shown in section 4. In the conclusion, the contributions of this research are reviewed.

2.2 The Bitran-Hax algorithm

This section presents the Bitran-Hax Algorithm to solve the problem P1. The problem P1 is a convex problem with linear constraints, so the following Karush-Kuhn-Turker (KKT) conditions are necessary and sufficient for the optimality as described in Mangasarian (1969). The Karush-Kuhn-Turker (KKT) conditions of the problem P1 are

$$f_i'(x_i) + \alpha g_i'(x_i) + v_i - w_i = 0, \text{ for all } i = 1, \dots, n \tag{2.7}$$

$$w_i(l_i - x_i) = 0, \text{ for all } i = 1, \dots, n \tag{2.8}$$

$$v_i(x_i - u_i) = 0, \text{ for all } i = 1, \dots, n \tag{2.9}$$

$$\sum_{i=1}^{n} g_i(x_i) = b, \tag{2.10}$$

$$l_i \le x_i \le u_i, \text{ for all } i = 1, \dots, n \tag{2.11}$$

$$v_i \ge 0, w_i \ge 0, \text{ for all } i = 1, \dots, n \tag{2.12}$$

where α is the Lagrange multiplier for the knapsack constraint (2.2), w_i and v_i, for all $i = 1, \dots, n$ are the Lagrange multiplier for the lower and upper bounds in (2.3), respectively.

If one relaxed the box constraint in (2.3) in the problem P1, the equation (2.7) becomes $f_i'(x_i) + \alpha g_i'(x_i) = 0$, for all $i = 1, \dots, n$ and x_i and α can be solved in closed form. Since $g_i(x_i)$ is linear, its gradient $g_i'(x_i)$ is merely a constant defined as gc_i. Assuming $f_i'(x_i)$, for all $i = 1, \dots, n$, is invertible, x_i can be calculated as,

$$x_i(\alpha) = -(f_i')^{-1}(\alpha \cdot gc_i) \text{ , for all } i = 1, \dots, n \tag{2.13}$$

where $(f_i')^{-1}$ denotes the inverse of f_i', the gradient of i objective function f_i, and α the Lagrange multiplier corresponding to the knapsack constraint in (2.2) can be obtained from the KKT conditions of P1 without the box constraints

$$\alpha = -\frac{\sum_{i=1}^{n} f_i'(x_i)}{\sum_{i=1}^{n} g_i'(x_i)} = -\frac{\sum_{i=1}^{n} f_i'(x_i)}{\sum_{i=1}^{n} gc_i} \tag{2.14}$$

If the solution in (2.13) satisfies the box constraints (2.3) in P1, then it is also optimal to P1. If not, then one can set x_i for some i to their upper or lower bounds and then the value of α can be recalculated. To determine which variables are to be fixed at their bounds, one defines following two terms as the sums of over and under limits, respectively.

$$P(\alpha) = \sum_{i \in A}(x_i(\alpha) - u_i), \text{ where } A = \{i : x_i(\alpha) \ge u_i, \text{ for all } i = 1, \dots, n\} \tag{2.15}$$

$$Q(\alpha) = \sum_{i \in B}(l_i - x_i(\alpha)), \text{ where } B = \{i : x_i(\alpha) \le l_i, \text{ for all } i = 1, \dots, n\} \qquad (2.16)$$

where $x_i(\alpha)$ is defined in (2.13). $P(\alpha)$ and $Q(\alpha)$ are used for choosing which set of variables (variables in A and B) to be pegged in the algorithm. The following theorem describes how to choose the pegging variables.

Theorem 2.2.1

If $P(\alpha) = Q(\alpha) = 0$, then $x_i^* = x_i(\alpha)$, for all $i = 1, \dots, n$ (2.17)

If $P(\alpha) > Q(\alpha)$, then $x_i^* = u_i$, for all $i \in A$ (2.18)

If $P(\alpha) < Q(\alpha)$, then $x_i^* = l_i$, for all $i \in B$ (2.19)

If $P(\alpha) = Q(\alpha) \neq 0$, then $x_i^* = u_i$, for all $i \in A$, $x_i^* = l_i$, for all $i \in B$ (2.20)

where x_i^* is the optimal solution of the problem P1.

Proof

This theorem can be proved using Bitran and Hax's (1981) work. The proof uses the KKT condition of the problem (P1). The Lemma 1, Lemma 2, Theorem 1, and Theorem 2 in Bitran and Hax (1981) described that each case of $P(\alpha)$ and $Q(\alpha)$ in (2.17) ~ (2.20) is related with the inequalities among the first derivatives $f_i'(x_i)$, for all i=1,…,n (See details in Bitran and Hax (1981)). Using this result and the KKT condition, Theorem 3 in their paper showed that setting the optimal value at each case at the upper bound or the lower bound or the optimal value as obtained by (2.13) in the relaxed problem is optimal in the original problem (P1). Ventura (1992) also showed the relationship of these cases at Theorem 6 in his paper. □

The related computational experiments can be found in Wu (1993). The detail steps of Bitran-Hax algorithm are as follows.

Bitran-Hax Algorithm

Step 0 (Initialization)

Let $E^r := E = \{i : i = 1, \cdots, n\}$, $b^r := b$, $r = 1$.

Step 1 (Calculating Dual Solution and Primal Solution)

Compute $\alpha^r = -\frac{\sum_{i=1}^{n} f_i'(x_i)}{\sum_{i=1}^{n} a_i}$,

Calculate $x_i(\alpha^r) = -(f_i')^{-1}(\alpha^r a_i)$ for all $x_i \in E^r$

Step 2 (Check feasibility)

For all $x_i \in E^r$.

If $l_i \leq x_i(\alpha^r) \leq u_i$ for all $x_i \in E^r$, then it is optimal and go to Step 6.

Otherwise, go to Step 3.

Step 3 (Calculate Pegging Sums & Check Stopping Criterion)

Compute $P(\alpha^r)$ and $Q(\alpha^r)$

$P(\alpha^r) = \sum_{i \in A^r}(x_i(\alpha^r) - u_i)$, where $A^r = \{i: x_i(\alpha^r) \geq u_i,\ i \in E^r\}$.

$\mathrm{Q}(\alpha^r) = \sum_{i \in B^r}(l_i - x_i(\alpha^r))$, where $B^r = \{i: x_i(\alpha^r) \leq l_i,\ i \in E^r\}$.

If A^r and B^r are empty,

then it is optimal and calculate $x_i(\alpha^r)$ for all $i \in E^r$ from (13) and go to Step 6.

Step 4 (Pegging Variables)

If $P(\alpha^r) > Q(\alpha^r)$, then

set $x_i = u_i$ for all $i \in A^r$,

let $E^{r+1} := E^r \backslash A^r$ and $b^{r+1} := b^r - \sum_{i \in A^r} u_i$.

If $P(\alpha^r) < Q(\alpha^r)$, then

set $x_i = l_i$ for all $i \in B^r$,

let $E^{r+1} := E^r \backslash B^r$ and $b^{r+1} := b^r - \sum_{i \in B^r} l_i$.

If $P(\alpha^r) = Q(\alpha^r) \neq 0$, then

set $x_i = u_i$ for all $i \in A^r$, set $x_i = l_i$ for all $i \in B^r$,

let $E^{r+1} := E^r \backslash (A^r \cup B^r)$ and $b^{r+1} := b^r - \sum_{i \in A^r} u_i - \sum_{i \in B^r} l_i$.

Step 5 (Check Stopping Criterion)

If $E^{r+1} = \emptyset$, then go to Step 6.

Else, set $r := r + 1$ and go to Step 1.

Step 6 (Optimum Found)

Set $x_i(\alpha^r)$ for all $i \in E$ as the optimal solution and terminate.

The Bitran-Hax Algorithm guarantees that at least one variable is pegged (or fixed) at each iteration since if there is no variable to be pegged at the current iteration, then the current solution on hand is optimal. Therefore, the algorithm can reduce the dimension of the problem as it progresses, and thus, guarantee the finite convergence. For the solution time, Wu (1993) has shown that the Bitran-Hax algorithm outperforms the Helgason et al.'s sequential line search and the random search by 25%~48% for quadratic network flow problems. It is, however, significantly slower than these two methods during the later stage of the solution process. With these empirical insights, this research has discovered several unnecessary procedures in the Bitran-Hax algorithm, which means the algorithm can be further streamlined. The calculation of a dual solution and its corresponding primal free variables at each iteration in Step 1 are the two most time consuming tasks, and these are the main foci to be improved in this research. In the next section, this research will show how to streamline these tasks.

2.3 The Dual Bound Algorithm (DBA)

In this section, a new pegging algorithm for continuous nonlinear knapsack problem with box constraints is proposed. The following a definition and two theorems demonstrate the basic ideas of the new algorithm.

Definition 2.3.1

The dual bound is the set of upper and lower bound of Lagrange multiplier corresponding to the solution of the relaxed problem, where the relaxed problem is the problem P1 ignoring the box constraint in (2.3).

Theorem 2.3.1

$x_i(\alpha)$, for $i = 1,2,\dots,n$, from (2.13) is a solution of the relaxed P1 problem if it satisfies its box constraint $l_i \le x_i(\alpha) \le u_i$ if and only if α is within the dual bound corresponding to the variable x_i as follows: $-\frac{f_i'(u_i)}{gc_i} \le \alpha \le -\frac{f_i'(l_i)}{gc_i}$, for $i = 1,2,\dots,n$, where $gc_i \neq 0$.

Proof

The optimal solution of the relaxed P1 problem is $x_i(\alpha) = -(f_i')^{-1}(\alpha \cdot gc_i)$, for $i = 1,2,\dots,n$, from (2.13). If one replaces $x_i(\alpha)$ by $-(f_i')^{-1}(\alpha \cdot gc_i)$, then the box constraint of P1 $l_i \leq x_i(\alpha) \leq u_i$, for $i = 1,2,\dots,n$ become $l_i \leq -(f_i')^{-1}(\alpha \cdot gc_i) \leq u_i$, for $i = 1,2,\dots,n$. Hence, the bounds on the Lagrange multiplier α according to the box constraint can be described by:

$$-\frac{f_i'(u_i)}{gc_i} \leq \alpha \leq -\frac{f_i'(l_i)}{gc_i}, \text{ for } i = 1,2,\dots,n.$$

Conversely, if one solves $-\frac{f_i'(u_i)}{gc_i} \leq \alpha \leq -\frac{f_i'(l_i)}{gc_i}$ for $-(f_i')^{-1}(\alpha \cdot gc_i)$, then one can get $l_i \leq -(f_i')^{-1}(\alpha \cdot gc_i) = x_i \leq u_i$ which is the same as bound constraint of the variable x_i . □

Theorem 2.3.1 provides a novel perspective to check the box constraint in (2.3) using the dual bound and shows that the box constraint in the primal problem can be replaced by the dual bound as defined in Definition 2.3.1. Each primal variable $x_i(\alpha^r)$ has its box constraint, and it can be transformed into the dual bound corresponding to each $x_i(\alpha^r)$. In the knapsack problem, the coefficient of the knapsack constraint in (2.3) denotes the weight of the each item. Therefore, if one of coefficients is zero, then the corresponding x_i does not need to be taken into account the knapsack constraint and it can be fixed to its upper bound, which is the reason this research can assume $gc_i \neq 0$, for all $i = 1,2,\dots,n$. As stated in Theorem 2.3.1, the calculation of the dual bound only requires the input values: gc_i, u_i, and l_i, which are known parameters. This property implies one does not have to update the dual bound at each iteration after it has been calculated initially.

Theorem 2.3.2

The solution $x_i(\alpha)$, for $i = 1,\dots,n$, obtained from (2.13) is an optimal solution of the problem P1 for a given dual solution α to D1, if the following inequality holds true:

$$Max\left\{-\frac{f_i'(u_i)}{gc_i}, i = 1,2,\dots,n\right\} \leq \alpha \leq Min\left\{-\frac{f_i'(l_i)}{gc_i}, i = 1,2,\dots,n\right\}.$$

Proof

In the Bitran-Hax algorithm, if the solution obtained from using the equation (2.13) satisfies the box constraints (2.3) of P1, then the solution is also optimal to P1. That is, if

$l_i \leq x_i(\alpha) \leq u_i$, for all $i = 1, \dots, n$, then $x_i(\alpha)$ is also the optimal solution of the problem P1. From the Theorem 2.3.1, one can easily replace all n inequalities of $l_i \leq x_i(\alpha) \leq u_i$, for all $i = 1, \dots, n$ with the following:

$$Max\left\{-\frac{f_i'(u_i)}{gc_i}, i = 1,2,\dots,n\right\} \leq \alpha \leq Min\left\{-\frac{f_i'(l_i)}{gc_i}, i = 1,2,\dots,n\right\}. \qquad \square$$

Theorem 2.3.2 shows if the dual solution α satisfies all the dual bounds, then the current solution is then optimal. The primal solution $x_i(\alpha^r)$ for all $i = 1, \dots, n$ satisfies all box constraints in the Bitran-Hax algorithm is the same that the dual solution α satisfies all dual bounds in the new algorithm. From these properties, the new algorithm is called Dual Bound algorithm (DBA). The DBA uses a correction value τ_i at each iteration when the algorithm calculates the values of $P(\alpha^r)$ and $Q(\alpha^r)$, so these values are the same as the values in the Bitran-Hax algorithm. Therefore, DBA and Bitran-Hax algorithm select the same variables to be pegged at each iteration. The pseudo-code of the proposed algorithm is now summarized below.

Dual Bound Algorithm (DBA)

Step 0 Initialization

Let $E^r := E = \{i : i = 1, \cdots, n\}$, $b^r := b$, $r = 1$.

Step 1 Calculating Dual Bounds

For all $x_i \in E^r$.

Compute $[LR_i, UR_i] = \left[-\frac{f_i'(u_i)}{gc_i}, -\frac{f_i'(l_i)}{gc_i}\right]$

Calculate $S_f^r = \sum_{i \in E^r} f_i'(x_i)$ and $S_g^r = \sum_{i \in E^r} gc_i$

Step 2 Update Dual Variable

Compute

$\alpha^r = -\frac{S_f^r}{S_g^r}$

Step 3 Calculate Pegging Sums & Check Stopping Criterion

Compute $P(\alpha^r)$ and $Q(\alpha^r)$

$P(\alpha^r) = \sum_{i \in A^r} (LR_i - \alpha^r)(\tau_i)$, where $A^r = \{i: \alpha^r \le LR_i,\ i \in E^r\}$ and τ_i is correction value.

Let $SA_f^r = \sum_{i \in A^r} f_i'(x_i),\ SA_g^r = \sum_{i \in A^r} gc_i$

$\mathrm{Q}(\alpha^r) = \sum_{i \in B^r} (\alpha^r - UR_i)(\tau_i)$,

where $B^r = \{i: \alpha^r \ge UR_i,\ i \in E^r\}$ and τ_i is correction value.

Let $SB_f^r = \sum_{i \in B^r} f_i'(x_i),\ SB_g^r = \sum_{i \in B^r} gc_i$

If A^r and B^r are empty, then it is optimal,

and calculate $x_i(\alpha^r)$ for all $i \in E^r$ from (2.13) and go to Step 6.

Step 4 Pegging Variables

If $P(\alpha^r) > Q(\alpha^r)$, then

set $x_i = u_i$ for all $i \in A^r$,

let $E^{r+1} := E^r \setminus A^r$ and $b^{r+1} := b^r - \sum_{i \in A^r} u_i$.

update $S_f^{r+1} = S_f^r - SA_f^r,\ S_g^{r+1} = S_g^r - SA_g^r$.

If $P(\alpha^r) < Q(\alpha^r)$, then

set $x_i = l_i$ for all $i \in B^r$,

let $E^{r+1} := E^r \setminus B^r$ and $b^{r+1} := b^r - \sum_{i \in B^r} l_i$.

update $S_f^{r+1} = S_f^r - SB_f^r,\ S_g^{r+1} = S_g^r - SB_g^r$.

If $P(\alpha^r) = Q(\alpha^r) \ne 0$, then

set $x_i = u_i$ for all $i \in A^r$, set $x_i = l_i$ for all $i \in B^r$,

let $E^{r+1} := E^r \setminus (A^r \cup B^r)$ and $b^{r+1} := b^r - \sum_{i \in A^r} u_i - \sum_{i \in B^r} l_i$.

update $S_f^{r+1} = S_f^r - SA_f^r - SB_f^r,\ S_g^{r+1} = S_g^r - SA_g^r - SB_g^r$.

Step 5 Check Stopping Criterion

If $E^{r+1} = \emptyset$, then go to Step 6.

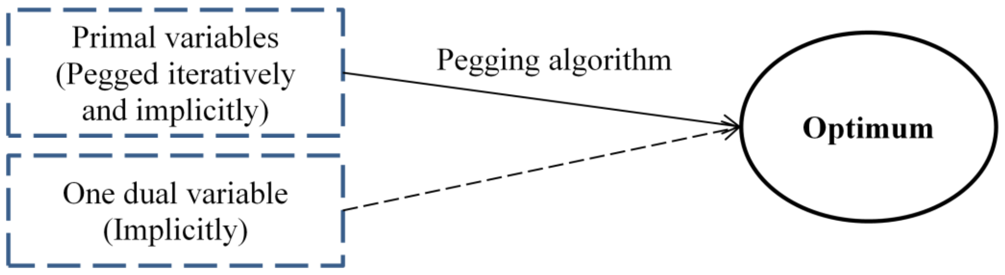
Else, set $r := r + 1$ and go to Step 2.

Step 6 Optimum Found

Set $x_i(\alpha^r)$ for all $i \in E$ is the optimal solution and terminate.

The above proposed DBA has two main potential advantages for improving the solution times: (1) eliminating the calculations of all the primal variables $x_i(\alpha^r)$ in every iteration and (2) only update α^r instead of recalculation of α^r. Compared with Bitran-Hax algorithm in the previous section, while the loop of Bitran-Hax algorithm is Step 1 to Step 5, the DBA loop is Step 2 to Step 5. The DBA loop does not include the calculation of dual bounds and primal variables. The algorithm uses the dual bounds on α^r (i.e., calculated in Step 1 to check the feasibility of box constraints implicitly instead of calculating the primal variable $x_i(\alpha^r)$ explicitly in Step 3. Although the algorithm should calculate dual bounds for each x_i, for $i \in E^r$ in Step 1, it is not necessary to the update dual bounds at each iteration because calculations of the dual bounds requires only the input data. Furthermore, in the DBA, the update of α^r is divided into two parts, updating of S_f^r and S_g^r which are calculated once in Step 1 and their values are updated in Step 4. The decrement of S_f^r and S_g^r are calculated in Step 3 with SA_f^r and SA_g^r or SB_f^r and SB_g^r. When some variables are gradually pegged in Step 4, the values of S_f^r and S_g^r are updated since the number of free variables in E^r decreases at least by one at each iteration. In Step 3, the term τ_i is multiplied to make the values of $P(\alpha^r)$ and $Q(\alpha^r)$ in the similar manner as those of the Bitran-Hax algorithm. Therefore, the values of $P(\alpha^r)$ and $Q(\alpha^r)$ in the DBA have the similar effects as those of the Bitran-Hax Algorithm. The DBA can get the same solution and the number of iteration as the Bitran-Hax algorithm. The only difference of the results between the Bitran-Hax and the DBA is the solution time. The basic idea of the Dual Bound algorithm is the following picture.

Figure 2.3 Dual Bound algorithm



Although the DBA does not calculate primal variables in every iteration, at least one primal variable is pegged at each iteration. Therefore, in the DBA, both primal and dual variables are optimized implicitly as illustrated Figure 2.3.

For the original Bitran-Hax algorithm, in the worst case, only one variable is set to its upper or lower bound at each iteration. The computational complexity of this process is $O(n)$. In addition, there are two calculations of the primal variables and the dual variable at each iteration and the evaluation of the feasibility for all remaining free variables. The computational complexity of this process is $O(n)$. Therefore, the overall computational complexity of the Bitran-Hax algorithm is $O(n^2)$. In the DBA, the pegging procedure has the same complexity $O(n)$ as the Bitran-Hax algorithm and there is the evaluation of the feasibility of all remaining free variables at each iteration of which is the calculation of pegging sums $P(\alpha^r)$ and $Q(\alpha^r)$ and its complexity is $O(n)$. The overall computational complexity of the DBA seems to be similar to the Bitran-Hax algorithm. However, except the pegging process, the DBA has only the evaluation of the feasibility process and does not have two calculations of the primal variables and the dual variable. For instance, let us consider the worst case problem that only one variable is pegged at each iteration. In the Bitran-Hax algorithm, the calculation of the primal variables in (2.13) is $n - k + 1$ at each iteration and the overall calculation is the same as the calculation of the sum of 1 to n because the number of iteration is n (the worst case), that is, $\frac{n(n+1)}{2}$. The calculation of the dual variable in (2.14) is also $\frac{n(n+1)}{2}$. The total variable updating effort can be as bad as $n(n+1)$. On the other hand, the DBA only needs to calculate the dual bounds for all variables initially, that is, n, but does not need to calculate two $\frac{n(n+1)}{2}$. In this respect, it is obvious that the DBA could be more efficient than the Bitran-Hax algorithm unless the problem has only one or two iterations to get the optimum. In the next section, the computational experiments show the practical performance of the DBA.

2.4 Numerical Examples

This section shows the Dual Bound Algorithm for a continuous quadratic knapsack problem as a special case of nonlinear knapsack problem as follows.

(P2)

$$Min \quad \frac{1}{2}\sum_{i=1}^{n} m_i x_i^2 + \sum_{i=1}^{n} q_i x_i \tag{2.21}$$

$$s.t. \quad \sum_{i=1}^{n} a_i x_i = b \tag{2.22}$$

$$l_i \le x_i \le u_i, \quad i = 1, \dots, n \tag{2.23}$$

where $m_i > 0 \in R^n, q_i \in R^n, a_i \in R^n, l_i \in R^n, u_i \in R^n$ for all $i = 1, \dots, n$, $b \in R^1$ and this research assumes $a_i \neq 0$ for all $i = 1, \dots, n$ and $\sum_{i=1}^{n} a_i l_i \leq b \leq \sum_{i=1}^{n} a_i u_i$.

To simplify the implementation, a variable transformation is first performed to let all the coefficients of the equality constraint become one. This requires the following change of variables.

Let $y_i = a_i x_i$, for all $i = 1, \dots, n$ and the bounds become

$$s_i = \begin{cases} a_i u_i, & \text{if } a_i > 0 \\ a_i l_i, & \text{otherwise} \end{cases}, \quad \text{for all } i = 1, \dots, n \tag{2.24}$$

$$t_i = \begin{cases} a_i l_i, & \text{if } a_i > 0 \\ a_i u_i, & \text{otherwise} \end{cases}, \quad \text{for all } i = 1, \dots, n \tag{2.25}$$

The problem is reformulated as

(P3) $$Min \ \ \frac{1}{2} \sum_{i=1}^{n} \frac{m_i y_i^2}{a_i^2} + \sum_{i=1}^{n} \frac{q_i y_i}{a_i} \tag{2.26}$$

$$s.t. \quad \sum_{i=1}^{n} y_i = b \tag{2.27}$$

$$t_i \leq y_i \leq s_i \text{ , for all } i = 1, \dots, n \tag{2.28}$$

The Karush Kuhn Turker (KKT) conditions of the problem P3 are

$$m_i y_i + a_i q_i + \alpha a_i^2 + v_i a_i^2 - w_i a_i^2 = 0, \text{ for all } i = 1, \dots, n \tag{2.29}$$

$$w_i (t_i - y_i) = 0, \text{ for all } i = 1, \dots, n \tag{2.30}$$

$$v_i (y_i - s_i) = 0, \text{ for all } i = 1, \dots, n \tag{2.31}$$

$$\sum_{i=1}^{n} y_i = b, \tag{2.32}$$

$$t_i \leq y_i \leq s_i, \text{ for all } i = 1, \dots, n \tag{2.33}$$

$$v_i \geq 0, w_i \geq 0, \text{ for all } i = 1, \dots, n \tag{2.34}$$

With this equation, the variable y_i is calculated as follows:

$$y_i(\alpha) = a_i x_i(\alpha) = -\frac{a_i q_i + a_i^2 \alpha}{m_i} \text{, for all } i = 1, \dots, n \tag{2.35}$$

where α is obtained from the KKT conditions of P2 without the bound constraints

$$\alpha = -\frac{\sum_{i=1}^{n} \frac{a_i q_i}{m_i} + b}{\sum_{i=1}^{n} \frac{a_i^2}{m_i}} \tag{2.36}$$

The equations (2.35) and (2.36) are corresponding to (2.13) and (2.14) respectively. The detail algorithm of Dual Bound is as follows:

Algorithm (Dual Bound : Quadratic Knapsack Problem)

Step 0 Initialization

Let $E^r := E = \{i : i = 1, \cdots, n\}$, $b^r := b,\ r = 1$.

Transform x_i into $y_i = a_i x_i$ and compute s_i and t_i for all $i \in E^r$ from (2.24) and (2.25).

Step 1 Calculating Dual Bounds

Compute dual bounds for all y_i, for $i \in E$.

$$[LR_i, UR_i] = \left[-\left(\frac{a_i q_i}{a_i^2} + \frac{m_i s_i}{a_i^2}\right), -\left(\frac{a_i q_i}{a_i^2} + \frac{m_i t_i}{a_i^2}\right)\right]$$

Calculate $S_{ff}^r = \sum_{i \in E^r} \frac{a_i q_i}{m_i}$ and $S_g^r = \sum_{i \in E^r} \frac{a_i^2}{m_i}$

Step 2 Update Dual Variable

Compute α^r

$\alpha^r = -\frac{S_{ff}^r + b^r}{S_g^r}$

Step 3 Calculate Pegging Sums & Check Stopping Criterion

Compute $P(\alpha^r)$ and $Q(\alpha^r)$

$P(\alpha^r) = \sum_{i \in A^r} (LR_i - \alpha^r)\left(\frac{a_i^2}{m_i}\right) = \sum_{i \in A^r}\left(-\frac{a_i q_i}{m_i} - \frac{a_i^2}{m_i}\alpha^r - s_i\right)$,

where $A^r = \{i: \alpha^r \le LR_i,\ i \in E^r\}$

let $SA_f^r = \sum_{i \in A^r} \frac{a_i q_i}{m_i},\ SA_g^r = \sum_{i \in A^r} \frac{a_i^2}{m_i}$

$Q(\alpha^r) = \sum_{i \in B^r} (\alpha^r - UR_i)\left(\frac{a_i^2}{m_i}\right) = \sum_{i \in B^r}\left(t_i + \frac{a_i q_i}{m_i} + \frac{a_i^2}{m_i}\alpha^r\right)$,

where $B^r = \{i: \alpha^r \ge UR_i,\ i \in E^r\}$

let $SB_f^r = \sum_{i \in B^r} \frac{a_i q_i}{m_i},\ SB_g^r = \sum_{i \in B^r} \frac{a_i^2}{m_i}$

If A^r and B^r are empty, then it is optimal,

and calculate $y_i(\alpha^r)$ for all $i \in E^r$ from (2.35) and go to Step 6.

Step 4 Pegging Variables

Pegging variables

If $P(\alpha^r) > Q(\alpha^r)$, then set $y_i = s_i$ for all $i \in A^r$,

let $E^{r+1} := E^r \backslash A^r$ and $b^{r+1} := b^r - \sum_{i \in A^r} s_i$.

update $S_{ff}^{r+1} = S_{ff}^r - SA_f^r,\ S_g^{r+1} = S_g^r - SA_g^r$.

If $P(\alpha^r) < Q(\alpha^r)$, then set $y_i = t_i$ for all $i \in B^r$,

let $E^{r+1} := E^r \backslash B^r$ and $b^{r+1} := b^r - \sum_{i \in B^r} t_i$.

update $S_{ff}^{r+1} = S_{ff}^r - SB_f^r,\ S_g^{r+1} = S_g^r - SB_g^r$.

If $P(\alpha^r) = Q(\alpha^r) \neq 0$, then set $y_i = s_i$ for all $i \in A^r$, set $y_i = t_i$ for all $i \in B^r$,

let $E^{r+1} := E^r \backslash (A^r \cup B^r)$ and $b^{r+1} := b^r - \sum_{i \in A^r} s_i - \sum_{i \in B^r} t_i$.

update $S_{ff}^{r+1} = S_{ff}^r - SA_f^r - SB_f^r,\ S_g^{r+1} = S_g^r - SA_g^r - SB_g^r$.

Step 5 Check Stopping Criterion

If $E^{r+1} = \emptyset$, then go to Step 6.

Else, set $r := r + 1$ and go to Step 2.

Step 6 Optimum Found

Set $x_i(\alpha^r) = y_i(\alpha^r)/a_i$ for all $i \in E$ is the optimal solution and terminate.

In the above algorithm, $S_{ff}^r + b^r$ is the S_f^r in the previous section. The term $\left(\frac{a_i^2}{m_i}\right)$ in step 3 is the τ_i which makes $P(\alpha^r)$ and $Q(\alpha^r)$ the same as those of Bitran-Hax algorithm. This section provides a simple example to show how the algorithm proposed in this research works. The simple example is solved with the methods of both the Bitran-Hax and the Dual Bound.

Example 4.1

$Min\ \ x_1^2 + 4\,x_2^2 - 2\,x_1 + 8\,x_2$

$s.t.\ \ 2\,x_1 + 3\,x_2 = 2$

$0 \le x_1, x_2 \ \le 2$

$\Longrightarrow$ $Min \ \ \frac{1}{2}(2x_1^2 + 8x_2^2) - 2\,x_1 + 8\,x_2$

$s.t. \ 2x_1 + 3x_2 = 2$

$0 \le x_1 \le 2$

$0 \le x_1 \le 2$

By the formulation, $m_1 = 2, m_2 = 8, q_1 = -2, q_2 = 8, a_1 = 2,$

$a_2 = 3, b = 2, l_1 = 0, u_1 = 2, l_2 = 0, u_2 = 2$

< **Bitran-Hax Algorithm** >

- Iteration 1

Step 0

$E^r := E = \{1,2\}, \ \ b^r := b = 2, \ r = 1$

$y_i = a_i x_i \Longrightarrow y_1 = 2x_1, \ \ y_2 = 3x_2, \ \ s_1 = 4, \ s_2 = 6, \ t_1 = 0, \ t_2 = 0$

Step 1

$$\alpha^1 = -\frac{\sum_{i\in E^1}\frac{a_i q_i}{m_i} + b^1}{\sum_{i\in E^1}\frac{a_i^2}{m_i}} = -\frac{\frac{2\cdot(-2)}{2} + \frac{3\cdot 8}{8} + 2}{\frac{4}{2} + \frac{9}{8}} = -\frac{\frac{24}{8}}{\frac{25}{8}} = -\frac{24}{25}$$

$$y_1 = a_1 x_1(\alpha^1) = -\frac{a_1 q_1 + a_1^2\alpha}{m_1} = -\frac{2\cdot(-2) + 4\cdot(-\frac{24}{25})}{2} = 2 + \frac{48}{25} = \frac{98}{25}$$

$$y_2 = a_2 x_2(\alpha^1) = -\frac{a_2 q_2 + a_2^2\alpha}{m_2} = -\frac{3\cdot 8 + 9\cdot(-\frac{24}{25})}{8} = -3 + \frac{27}{25} = -\frac{48}{25}$$

Step 2

$0 \le y_1 = \frac{98}{25} \le 4$: Yes

$0 \le y_2 = -\frac{48}{25} \le 6$: No

Step 3

$A = \emptyset, \ \ B = \{2\}$

$P(\alpha^1) = \sum_{i\in A}(y_i(\alpha^1) - s_i) = 0$

$$Q(\alpha^1) = \sum_{i\in B}\left(t_i - y_i(\alpha^1)\right) = 0 - \left(-\frac{48}{25}\right) = \frac{48}{25}$$

Step 4

Since $P(\alpha^1) < Q(\alpha^1)$, $y_2 = 0$, $E^2 := E^1 \backslash B = \{1\}$

$$b^{r+1} := b^2 = b^1 - \sum_{i\in B} t_i = 2 - 0 = 2$$

Step 5

Since $E^2 \neq \emptyset$, set $r := r + 1 = 2$, go to Step 1

- Iteration 2

Step 1

$$\alpha^2 = -\frac{\sum_{i\in E^2}\frac{a_i q_i}{m_i} + b^2}{\sum_{i\in E^2}\frac{a_i^2}{m_i}} = -\frac{\frac{2\cdot(-2)}{2}+2}{\frac{4}{2}} = 0$$

$$y_1 = a_1 x_1(\alpha^2) = -\frac{a_1 q_1 + a_1^2\alpha^2}{m_1} = -\frac{2\cdot(-2)+4\cdot 0}{2} = 2$$

Step 2

$0 \leq y_1 = 2 \leq 4$: Yes

Go to Step 6.

Step 6

Set $x_1(\alpha^2) = \frac{y(\alpha^2)}{a_1} = \frac{2}{2} = 1$, $x_2(\alpha^2) = \frac{y(\alpha^2)}{a_2} = \frac{0}{3} = 0$

Therefore, $(x_1,\ x_2)^* = (1,\ 0)$ is optimal.

Next, the Dual Bound Algorithm is used to solve this problem.

< Dual Bound Algorithm >

- Iteration 1

Step 0

$E^r := E = \{1, 2\}$, $b^r := b = 2$, $r = 1$

$$y_i = a_i x_i \Longrightarrow y_1 = 2x_1, \quad y_2 = 3x_2, \quad s_1 = 4, \; s_2 = 6, \; t_1 = 0, \; t_2 = 0$$

Step 1

$$[LR_1, UR_1] = \left[-\left(\frac{a_1 q_1}{a_1^2} + \frac{m_1 s_1}{a_1^2}\right), -\left(\frac{a_1 q_1}{a_1^2} + \frac{m_1 t_1}{a_1^2}\right)\right]$$

$$= \left[-\left(\frac{2(-2)}{4} + \frac{2\cdot 4}{4}\right), -\left(\frac{2(-2)}{4} + \frac{2\cdot 0}{4}\right)\right] = [-1,1]$$

$$[LR_2, UR_2] = \left[-\left(\frac{a_2 q_2}{a_2^2} + \frac{m_2 s_2}{a_2^2}\right), -\left(\frac{a_2 q_2}{a_2^2} + \frac{m_2 t_2}{a_2^2}\right)\right]$$

$$= \left[-\left(\frac{3\cdot 8}{9} + \frac{8\cdot 6}{9}\right), -\left(\frac{3\cdot 8}{9} + \frac{8\cdot 0}{9}\right)\right] = \left[-8, -\frac{8}{3}\right]$$

$$S_{ff}^1 = \sum_{i\in E^1} \frac{a_i q_i}{m_i} = \frac{2\cdot(-2)}{2} + \frac{3\cdot 8}{8} = \frac{8}{8}$$

$$S_g^1 = \sum_{i\in E^1} \frac{a_i^2}{m_i} = \frac{4}{2} + \frac{9}{8} = \frac{25}{8}$$

Step 2

$$\alpha^1 = -\frac{S_{ff}^1 + b^1}{S_g^1} = -\frac{\frac{8}{8}+2}{\frac{25}{8}} = -\frac{\frac{24}{8}}{\frac{25}{8}} = -\frac{24}{25}$$

Step 3

$P(\alpha^1) = \sum_{i\in A^1}\left(-\frac{a_i q_i}{m_i} - \frac{a_i^2}{m_i}\alpha^r - s_i\right) = 0$, A^1 is empty.

$\mathrm{Q}(\alpha^1) = \sum_{i\in B^1}\left(t_i + \frac{a_i q_i}{m_i} + \frac{a_i^2}{m_i}\alpha^1\right) = 0 + \frac{3\cdot 8}{8} + \frac{9}{8}\left(-\frac{24}{25}\right) = \frac{48}{25}$, $B^1 = \{2\}$

$SB_f^1 = \sum_{i\in B^1} \frac{a_i q_i}{m_i} = \frac{24}{8} = 3, \; SB_g^1 = \sum_{i\in B^1} \frac{a_i^2}{m_i} = -\frac{27}{25}$

Step 4

Since $P(\alpha^1) < Q(\alpha^1), \; y_2 = t_2 = 0, \quad E^2 := E^1 \backslash B^1 = \{1\}$

$S_{ff}^2 = S_{ff}^1 - SB_f^1 = 1 - 3 = -2 \qquad b^2 := b^1 - \sum_{i\in B^1} t_i = 2 - 0 = 2$

Step 5

Since $E^2 \neq \emptyset$, set $r := r + 1 = 2$, go to Step 2

- Iteration 2

Step 2

$$\alpha^2 = -\frac{S_{ff}^2 + b^2}{S_g^2} = -\frac{-2+2}{\frac{25}{8}} = 0$$

Step 3

$P(\alpha^2) = \sum_{i \in A^2}\left(-\frac{a_i q_i}{m_i} - \frac{a_i^2}{m_i}\alpha^r - s_i\right) = 0$, A^2 is empty.

$\mathrm{Q}(\alpha^1) = \sum_{i \in B^2}\left(t_i + \frac{a_i q_i}{m_i} + \frac{a_i^2}{m_i}\alpha^r\right) = 0$, B^2 is empty.

It is optimal.

Calculate $y_1 = a_1 x_1(\alpha) = -\frac{a_1 q_1 + a_1^2 \alpha}{m_1} = -\frac{2\cdot(-2)+4\cdot 0}{2} = 2$

and go to Step 6.

Step 6

Set $x_1(\alpha) = \frac{y(\alpha)}{a_1} = \frac{2}{2} = 1$, $x_2(\alpha) = \frac{y(\alpha)}{a_2} = \frac{0}{3} = 0$

Therefore, $(x_1, x_2)^* = (1, 0)$ is optimal.

From this numerical example, two methods have the same optimal solution and the number of iteration. The Bitran-Hax Algorithm calculates α and $y_i(\alpha)$ in step 1 at every iteration. On the other hand, The Dual Bound Algorithm calculates dual bounds at the beginning and calculates three components of α calculation in step 1. The loop of the Dual Bound Algorithm starts from step 2. The Dual Bound Algorithm does not calculate $y_i(\alpha)$ at every iteration. Furthermore, the Dual Bound Algorithm updates two components of α calculation (S_g^r, S_{ff}^r, and b^r) instead of calculating all it again. In this simple problem, the Dual Bound Algorithm is not much attractive. However, if the size of the problem or the number of iteration is increasing, the solution time would be different. Some experimental results for various large scale problems are shown in the next section.

2.5 Experimental results

In this section, the computational experiments for both the Dual Bound algorithm (DBA) and the Bitran-Hax algorithm are conducted on randomly generated problems with different sizes, and then tested on different types of applications with common data sets. Both algorithms are implemented in C programming language, complied using gcc and ran on a Fedora 7, 64 bit Red Hat Linux machine with 2 GB memory and Intel Duo Core^{TM} 2 CPU running 2.66 GHz. Experiments on different types of problems demonstrated a comparison between the Dual Bound and the Bitran-Hax algorithms.

In the first round of tests, the continuous quadratic knapsack test problems with various sizes were randomly generated to establish a baseline comparison between the Bitran-Hax algorithm and the DBA.

First, for testing the continuous quadratic knapsack problem, data sets were randomly generated with the following distribution: $a_i, m_i \sim U[10,25]$, $q_i \sim U[1,1000]$, and $l_i, u_i \sim U[1,10000]$, where $U[a, b]$ denotes the uniform distribution with range from a to b. For generating the bound values, two values are generated from $U[1,10000]$ and this research puts the larger value to u_i and the smaller one to l_i to satisfy the inequality $l_i < u_i$. The b value in (22) is generated by $b \sim U[\sum_{i=1}^{n} a_i l_i , \sum_{i=1}^{n} a_i u_i]$. The experimental results are as follows.

Table 2.1 Problem size (500,000 variables)

	Solution time (seconds)		
500k	**DBA**	**Bitran-Hax**	**Improved (%)**
1	1.05	1.13	7.07
2	1.08	1.17	7.69
3	1.04	1.15	9.56
4	1.05	1.12	6.25
5	1.06	1.19	10.92
6	1.05	1.15	8.70
7	1.08	1.17	7.69
8	1.03	1.12	8.04
9	1.04	1.16	10.34
10	1.06	1.16	8.62
		Average	**8.49%**

Table 2.2 Problem size (1,000,000 variables)

	Solution time (seconds)		
1000k	**DBA**	**Bitran-Hax**	**Improved (%)**
1	2.10	2.35	10.64
2	2.09	2.28	8.33
3	2.13	2.34	8.97
4	2.08	2.27	8.37
5	2.13	2.34	8.97
6	2.14	2.40	10.83
7	2.12	2.34	9.40
8	2.11	2.35	10.21
9	2.09	2.30	9.13
10	2.12	2.36	10.17
		Average	**9.5%**

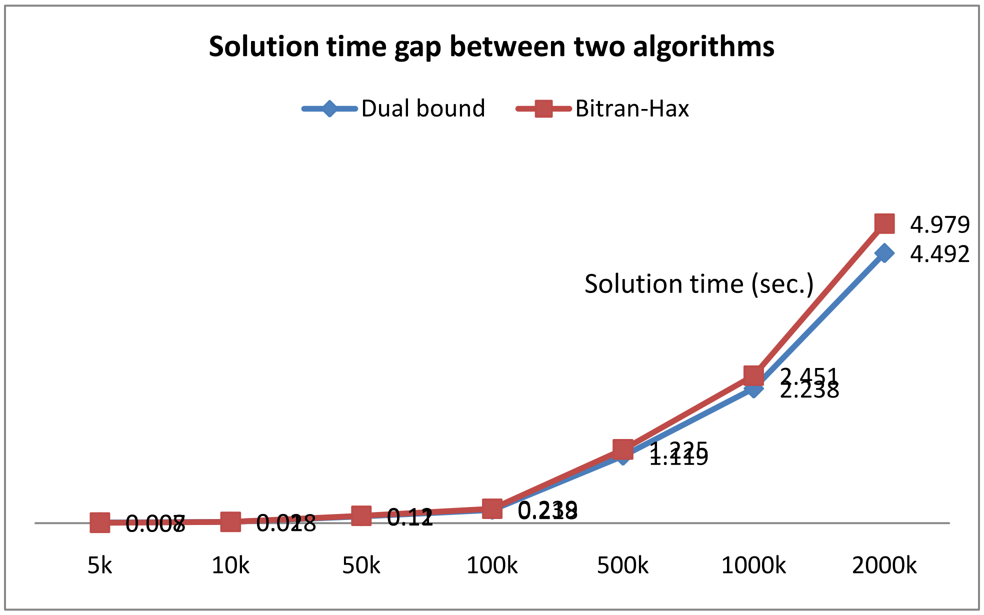
Table 2.3 Problem size (2,000,000 variables)

	Solution time (seconds)		
2000k	**DBA**	**Bitran-Hax**	**Improved (%)**
1	4.20	4.61	8.89
2	4.22	4.67	9.63
3	4.20	4.85	13.40
4	4.26	4.66	8.58
5	4.27	4.82	11.41
6	4.26	4.82	11.62
7	4.28	4.83	11.39
8	4.19	4.67	10.28
9	4.26	4.75	10.36
10	4.18	4.68	10.68
		Average	**10.62%**

The experimental results from Tables 2.1 to 2.3 have shown that the DBA outperforms the Bitran-Hax algorithm by 8 ~ 10%. In this set of test problems, around 30~40% of the remaining free variables (i.e., variables having their optimal values strictly between their bounds) are in the optimal solution.

The next set of test problems were randomly generated with the following distributions: $a_i, m_i \sim U[1.0,1.5]$, $q_i \sim U[0.0,5.5]$, $l_i \sim U[-2.0,-1.0]$, and $u_i \sim U[1.0,4.0]$.

Figure 2.4 The gap of solution time between Bitran-Hax and Dual Bound



In Figure 2.4, the horizontal axis denotes the problem sizes ranging from 5,000 to 2,000,000 variables. From the results illustrated in Figure 2.4, this research discovered that when the problem size increases, the gap of the solution times between the Bitran-Hax and the DBA also increases. When the problem size is small (i.e., ranging from 5,000 to 100,000 variables), the gap of the solution time is small. On the other hand, for the large size problems (i.e., from 500,000 variables and beyond), the gap is large. The percentages of free variables at the achieved optimal solution are about 70% in these test problems. The percentages of free variables at the final optimal solution are larger than the previous experimental results in Tables 2.1~3. However, the improvements on solution times are not much different (i.e., around 8~10% of improvement) because the number of iteration in the results of Figure 2.4 is less than those presented in Tables 2.1~3. Therefore, the DBA outperforms the Bitran-Hax algorithm regardless the number of optimal free variables or required total number of iterations to achieve the optimal solutions.

Second, this research examined the test cases for the some real-world applications having embedded convex knapsack problems in its optimization problems. Two types of optimization problems: quadratic network flow problem and portfolio optimization problem have been tested to assess the effeteness of the DBA. The quadratic network flow problems were randomly generated using a modified version of the generator NETGEN in Klingman et al. (1974) to generate nonlinear separable cost functions. The quadratic and linear cost coefficients are distributed by $U[0.01,10]$. The detail information for this data set and solving algorithm is in (Arasu, 2000). The algorithm framework for solving this data set is a hybrid dual algorithm which combined the conjugate gradient and the dual preflow algorithms in Arasu (2000). The quadratic knapsack problem is a well-known line-search subproblem using in this algorithm. The computational results are presented in Table 2.4. Table 2.4 reports the problem sizes tested (denoted by the numbers of nodes and arcs in the tested networks), and solution times for the Bitran-Hax algorithm and the DBA in seconds.

Table 2.4 Computational Results for Quadratic Network Problems

Problem Sizes		Solution time (seconds)		
# of nodes	# of arcs	Bitran-Hax	DBA	Improved (%)
200	400	0.16	0.13	18.75
200	1000	0.07	0.05	28.57
250	1000	0.11	0.07	36.36
300	600	0.21	0.15	28.57
400	800	0.72	0.50	30.56
400	2000	0.19	0.16	15.79
400	2400	0.33	0.17	48.48
450	2400	0.40	0.23	42.50
500	1000	0.55	0.49	10.91
500	2000	0.31	0.19	38.71
500	2400	0.39	0.28	28.21
500	2500	0.41	0.34	17.07
1000	40000	5.51	3.67	33.39
4000	20000	9.77	5.46	44.11
4500	50000	11.89	6.71	43.57
5000	50000	11.71	6.60	43.64

The percentage of improving solution time is around 10~48%. It can be also seen from Table 2.4 that the speedups of the solution times between the Bitran-Hax algorithm and the DBA increases as the problem size increases.

The tested financial problems in this research are the stochastic portfolio optimization problems. These problems have been modeled as the two-stage stochastic programming problems. These problems were solved using progressive hedging algorithm with potential reduction function (Arasu, 2000). The detail information of the data set and the progressive hedging algorithm can be found in Arasu (2000). The results are summarized in Table 2.5 below.

Table 2.5 Computational Results for the Portfolio Optimization Problems

Problem Sizes			Solution time (seconds)		
Asset	Periods	Scenarios	Bitran-Hax	DBA	Improved (%)
15	8	18	2.74	2.14	21.90
15	6	52	7.80	5.96	23.59
15	8	80	7.17	5.53	22.87
15	8	72	11.14	8.43	24.33
15	4	70	3.02	2.34	22.52
15	8	48	7.60	5.79	23.82
15	8	40	6.53	4.95	24.24
15	8	60	10.14	7.62	24.85
15	8	100	8.79	6.83	22.30
15	8	120	10.59	8.21	22.47
15	8	124	11.00	8.45	23.18
15	8	125	11.34	8.68	23.46
15	8	200	17.87	13.85	22.50
15	8	250	22.35	17.21	23.00
15	8	400	35.88	27.58	23.13
15	8	500	44.71	34.73	22.32

This financial problem has a line-search subproblem in the similar format to the quadratic network flow problem, which is also a quadratic knapsack problem. In this case, the continuous quadratic knapsack problem is the subproblem of the subproblem of the progressive hedging algorithm as Arasu (2000) referred the problem as two-stage stochastic network model. From this round of the computational experiments, the solution times can be improved around 21~25% if the DBA is used. From the results depicted in Table 2.4 and 2.5, if the continuous quadratic knapsack problem is a subproblem of other optimization problems and the size of these optimization problems is large and the number of iteration in solution procedure is large, then the DBA is more attractive than the Bitran-Hax algorithm.

The differences in practical computations for the two algorithms follow. The DBA should calculate the dual bounds for all variables once at the initial step and does not calculate them again during iterations. At the last iteration, the DBA should calculate the remaining primal free variables. On the other hand, the Bitran-Hax algorithm does not calculate the dual bounds, but it should calculate all primal free variables not pegged at the iteration during each iteration. Therefore, even if the problem is large, the Bitran-Hax algorithm would be faster than the DBA when the number of iterations is small and the calculations of the dual bounds and the final remaining free variables in DBA are larger than the calculations of free variables during iterations in the Bitran-Hax algorithm. In the experiments, the results of the quadratic network and financial optimization problems are more improved than the quadratic knapsack problem. The reason is that the number of iterations in Tables 2.4 and 2.5 is larger than that in Table 2.1~3 even though the size of the problem in Table 2.1~3 is larger. The average number of iterations in the quadratic knapsack problems is 8. On the other hand, the quadratic network problems have approximately 110 iterations for the main problem, which means algorithm calls the quadratic knapsack problem as a subproblem 110 times during the solving process. Thus, the total number of iterations for the quadratic knapsack problem in the quadratic network problem is approximately 110 times the average number of iterations of a quadratic knapsack problem. In summary, the DBA is more attractive if three conditions are met: the size of the problem is large, the number of iteration is large, and the number of free variables is large.

2.5 Conclusions

This chapter proposed a new pegging algorithm, the Dual Bound algorithm, to solve the separable continuous nonlinear knapsack problem with box constraints. The nonlinear knapsack problem has many real-world applications and is frequently embedded as a subproblem in many large-scale mathematical models. The main motivation of the new algorithm is to reduce the time consuming variable updating procedures in the Bitran-Hax algorithm to improve the overall efficiency. The Bitran-Hax algorithm must recalculate the dual variable and primal variables at each iteration, which is frequently the most computationally involved procedure. In the Dual Bound algorithm (DBA), once dual bounds are initially calculated, they can be used throughout the solution procedure while the Bitran-Hax algorithm must recalculate all remaining free primal variables at each iteration. To update the dual variable α, the DBA divides the calculation of α

into several smaller components and updates the each component, individually only when necessary.

The results of the two types of experiments with quadratic objective functions show that the DBA can solve such problems faster than the Bitran-Hax algorithm. The first type of experiment used randomly generated, continuous quadratic knapsack problems with sizes ranging from 500 to 2,000,000 variables. The computational results revealed that the DBA improves the average solution times by approximately 8~10% over the Bitran-Hax algorithm while obtaining the same optimal solutions. When problem sizes increased, the solution time of the DBA became even faster than the Bitran-Hax algorithm. The second type of experiment involved the continuous quadratic knapsack problem as a subproblem of other optimization models. The quadratic network flow problems and the portfolio optimization problems were tested in this round of computational comparisons. The quadratic network flow problem has a line-search subproblem as a continuous quadratic knapsack problem and the portfolio management problem is modeled by two-stage stochastic network problem with a similar line-search routine. The results of these problem sets show that the DBA can achieve approximately 10~48% faster results for the quadratic network flow problems and 21-25% faster results for the portfolio optimization problems than the Bitran-Hax algorithm. The results of the extensive computational experiments reveal that the DBA is an attractive alternative for the Bitran-Hax algorithm for large-scale problems. In addition, the computational experiments suggest the DBA provides an edge when used to solve an embedded subproblem, in which a large number of nonlinear knapsack problems are repeatedly resolved with possible warm-starts and when significantly large number of variables are bounded at the optimum.

In the future, the DBA can be extended to handle broader problems such as the non-separable objective functions with a dense Hessian matrix or problems with generalized upper-bounding constraints. The DBA optimizes both primal variables and a dual variable implicitly and using smaller components updates to achieve the maximal efficiency. With these concepts, more efficient algorithms could be possible because checking feasibility uses only one dual variable instead of all the primal variables. If the dual variable could initially be chosen or estimated with better insight, the number of iterations would be significantly reduced. This algorithm also can be applied to more complicated, larger problems in the real-life applications.

The problem in this chapter is separable, convex, continuous, and bounded nonlinear knapsack problem. The separable problem denotes the Hessian matrix is a diagonal matrix. In the next chapter, the problem arising in support vector machine has nonseparable Hessian matrix. The solution algorithm for the nonseparable problem is different from separable case in this chapter. However, the method in chapter 3 will split the Hessian matrix into sum of simple diagonal matrices and solve a separable nonlinear knapsack problem iteratively. Thus, the nonlinear knapsack problem in this chapter will be a subproblem of the problem in the next chapter and be solved iteratively.

CHAPTER 3 - ν-Support Vector Machine

This chapter proposes a solving approach for the ν-support vector machine (SVM) for classification problems using the modified matrix splitting method and incomplete Cholesky decomposition. The SVM problem is solved by solving its dual problem because there is only one dual variable. Using the augmented Lagrange method, the dual formulation of the ν-SVM classification becomes a singly linearly constrained convex quadratic program with box constraints. The Kernel Hessian matrix of the SVM problem is dense and large. The matrix splitting method combined with the projection gradient method solves the subproblem with a diagonal Hessian matrix iteratively until the solution reaches the optimum. The subproblem is a nonlinear knapsack problem with a diagonal Hessian matrix described in chapter 2. Thus, the Bitran-Hax or Dual Bound algorithm is used for solving this subproblem. The method can choose one of several line search and updating alpha method in the projection gradient method. The incomplete Cholesky decomposition is used for the calculation of the large scale Hessian and vectors. The experimental results show that the newly proposed method has a potential for the alternative of the solution method for the ν-SVM classification problem even if the size of the problem is medium or large.

Section 1 introduces a brief history of machine learning and SVM. In section 2, the solving algorithm for the ν-SVM is described. The decomposition method and the data structure of Hessian matrix are showed in section 3. Section 4 presents the experimental results. In the conclusion, the contributions of this chapter are reviewed.

3.1 Introduction

The machine learning truly began with Rosenblatt's perceptron from the research of neurodynamics in Rosenblatt (1962). Rosenblatt constructed the perceptron to solve pattern recognition problems and described the concept can be generalized. The problem was to find a rule to separate data into two groups using given examples. The learning theory aims to find the rule from data observed to predict the future. Finding the rule is to find the pattern of the data. For example, let us assume that there are training data set $(\boldsymbol{x}_1, y_1)$, ..., $(\boldsymbol{x}_N, y_N)$, where $\boldsymbol{x}_i \in R^m$ and $y_i \in R^1$ for all $i = 1, \dots, N$. If the pattern of the data set is known, one can estimate the response y'_1, ..., y'_N for $\boldsymbol{x}'_1$, ..., $\boldsymbol{x}'_N$. To find the rule, the perceptron uses the hierarchical network with the data set on the bottom of the network and the result on the top of the network. There are arcs between the layers. The arcs have weights. The perceptron aims to find the weights which perfectly explain how to get the results. The artificial intelligence group also got involved in this research in 1980s. The name of percepton was changed into neural network. The neural network has been used to find the rule in the learning theory. However, the neural network method was dependent on the data set and it took much time to find the weights in the large scale problem. In 1986, there was the second breakthrough in learning theory. The backpropagation method was introduced by A and B independently. The backpropagation method significantly increased the speed of finding the weights in neural network. The backpropagation neural network has been a popular method since then. However, the backpropagation neural network still had two problems: slow convergence and less generalization. In 1995, Vapnik introduced the SVM that has been a popular method of learning theory so far. The SVM based on the concept of the structural risk minimization principle have gained popularity with many attractive features such as statistical background, good generalization, and promising performance. This research focuses on the solving algorithm for the SVM for the classification problem.

The SVM maps the data set into another space called a feature space and classifies or does a regression the data set using separating hyperplane. Using the SVM, it is necessary to solve a quadratic programming problem that has a dense Hessian matrix. In this chapter, this research proposes an approach to solve the SVM for classification problems. The matrix splitting method with the nonmonotone line search technique is used to solve the quadratic programming problem and a penalty method is used to move one of constraints to the objective function. Bitran-Hax or Dual Bound algorithm in chapter 2 is used for solving the subproblem that is a quadratic nonlinear knapsack problem. This research also uses an incomplete Cholesky decomposition method for the dense Hessian matrix for large scale problems.

3.2 Support Vector Optimization

The SVM originally was developed for classification problems also called pattern recognitions and extended for regression problems. This research focuses on the classification problem. When one wants to classify certain data into two groups, one can think three possible cases. The first case is that the two groups are known and can be separated trivially. For example, there are ten pets: five dogs and five cats. One knows the information of two groups:

dog group and cat group. Then one can easily classify ten pets into two groups. The second case is that the two groups are known, but can not be separated trivially. For example, there are ten dogs. One wants to classify the dogs into two groups: biting a thief or not when the dog confronts a thief. One knows the information of the two groups: biting group and not biting group. However, one can not classify the group of dogs trivially. The last case is that the two groups are unknown and cannot be separated trivially. For example, there are ten dogs. One wants to classify the dogs, but one does not know how to classify the dogs and do not have any information of the groups. The second and the third case can be considered in the field of learning theory. The supervised learning is related to the second case and the unsupervised learning is the third case. The SVM is one of supervised learning methods. Thus, it is assumed that the information of group and the data of history are known in the SVM.

Let us assume there are two groups and sample data. The training data $\boldsymbol{x} \in R^m$ is a vector and its result is $y \in \{-1,1\}$ which denotes two groups. If it assumes that there are n training data, then the pairs of training data are $(\boldsymbol{x}_i, y_i) \in R^{n\times m} \times \{-1,1\}$, for $i = 1, \dots, n$. The goal of this research is to find the pattern of the data using these pairs of training data. Suppose $P(\boldsymbol{x}, y)$ is an unknown probability distribution of data set and $f(\boldsymbol{x}, \alpha)$ is defined a mapping from input $\boldsymbol{x}$ to output y. The function $f(\alpha)$ is referred to hypothesis and the set of functions $\{f(\alpha) : \alpha \in \Lambda\}$ is called the hypothesis space denoted by H. The parameter α is an adjustable parameter and specifies a particular function in the hypothesis space. The symbol Λ denotes an index set. The expected risk or expected error is

$$R(\alpha) = \int \frac{1}{2} |(f(\boldsymbol{x}, \alpha) - y)| dP(\boldsymbol{x}, y) \tag{3.1}$$

However, $R(\alpha)$ cannot be calculated exactly because the probability distribution is unknown. Instead one calculates the bound of the expected risk. If one has n data observed, the empirical risk is defined as

$$R_{emp}(\alpha) = \frac{1}{2n} \sum_{i=1}^{n} |(f(\boldsymbol{x_i}, \alpha) - y_i)| \tag{3.2}$$

If one assumes the confidence level is $(0 \le \eta \le 1)$, then Vapnik introduced the bound of the expected risk is

$$R(\alpha) \le R_{emp}(\alpha) + \sqrt{\frac{h\left(\log\left(\frac{2n}{h}\right)+1\right)-\log\left(\frac{\eta}{4}\right)}{n}} \tag{3.3}$$

where h is defined the VC dimension for the hypothesis space. Therefore, if one minimizes the right hand side, the bound of the expected risk, then the bound is close to the original expected risk. The second term of the bound denotes the confidence term. When one minimizes the empirical risk, the confidence term is increased. Similarly, the empirical risk increases if one reduces the confidence term. Therefore, one should have a tradeoff between the empirical risk and the confidence term. However, it is hard to get an appropriate VC dimension and to minimize the problem. The structural risk minimization (SRM) principle is to minimize the risk functional with respect to both the empirical risk and the confidence term, where the functional is a function of which variables are functions. In the above inequality, the first term of the right hand side means how the data chosen is good and the second term is for the complexity of the model. There are two approaches to minimize the right hand side of the inequality: neural network and SVM. The neural network keeps the confidence term fixed and minimizes the empirical risk. On the other hand, the SVM keeps the value of empirical risk fixed and minimizes the confidence term.

To minimize the empirical risk functional, a set of linear indicator functions is defined as follows.

$$f(x, w) = \text{sign}\{(w \cdot x)\}, w \in R^n \tag{3.4}$$

where $(w \cdot x)$ denotes an inner product between vectors w and x.

Assuming the number of data is n, the goal is to find the coefficient w_0 that minimize the empirical risk functional

$$R_{emp}(w) = \frac{1}{n} \sum_{i=1}^{n} \left(y_i - f(x_i, w)\right)^2 \tag{3.5}$$

If the training set is separable without error which means the empirical risk can become zero, there exists a finite step procedure to find the vector w. On the other hand, if the training set is not separable, the problem becomes NP-complete. Furthermore, one cannot use the regular gradient based method since the gradient of the functional is either equal to zero or undefined. With these facts, one needs to approximate the indicator functions so called sigmoid function as follows.

$$\bar{f}(x, w) = S\{(w \cdot x)\} \tag{3.6}$$

where $S(u)$ is a smooth monotonic function as follows.

$$S(-\infty) = -1, S(\infty) = 1 \tag{3.7}$$

The neural network approach has some problems. The quality of the solution depends on many factors, in particular on the initialization of weight matrices. The convergence of the method is slow. The choice of the scaling factor in the sigmoid function is a trade-off between the quality of approximation of indicator function and the rate of convergence.

The SVM approach uses an optimal separating hyperplane as described in Vapnik and Chervonenkis (1974), Vapnik (1979) which separates the data with maximum distance (margin) between the data and the hyperplane.

Suppose that there are the training data

$$(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n), \text{ where } \boldsymbol{x} \in R^m \text{ and } y \in \{-1, 1\} \tag{3.8}$$

The data has two classes: the one is the class the target value y is -1 and the other class is the target value y is 1. The separating hyperplane is defined as

$$(\boldsymbol{w} \cdot \boldsymbol{x}) + b = 0, \text{ where } \boldsymbol{w} \in R^n \text{ and } b \in R \tag{3.9}$$

The decision function $f(\boldsymbol{x})$ is

$$f(\boldsymbol{x}) = \operatorname{sign}\big((\boldsymbol{w} \cdot \boldsymbol{x}) + b\big). \tag{3.10}$$

When the input data are separable, the hyperplane has the following conditions

$$(\boldsymbol{w} \cdot \boldsymbol{x}_i) + b \geq 1, \quad \text{if } y_i = 1 \tag{3.11}$$

$$(\boldsymbol{w} \cdot \boldsymbol{x}_i) + b \leq -1, \quad \text{if } y_i = -1. \tag{3.12}$$

These two constraints (3.11) and (3.12) can be combined as

$$y_i\big((\boldsymbol{w} \cdot \boldsymbol{x}_i) + b\big) \geq 1 \tag{3.13}$$

The distance between the data and the hyperplane is $\frac{2}{\|\boldsymbol{w}\|}$, where $\|\cdot\|$ denotes the norm of the vector. The optimal separating hyperplane is the hyperplane with the maximum distance. Therefore, to obtain the optimal separating hyperplane, one needs to solve the following problem:

$$\min_{\boldsymbol{w},b} \frac{1}{2}\|\boldsymbol{w}\|^2 \tag{3.14}$$

$$s.t. \quad y_i\big((\boldsymbol{w} \cdot \boldsymbol{x}_i) + b\big) \geq 1, i = 1, \ldots, n \tag{3.15}$$

In this model, variables are $\boldsymbol{w}$ and b while $\boldsymbol{x}_i$ and y_i are input data. One can consider the dual problem of this problem to solve it efficiently. The Lagrangian dual with multiplier $\boldsymbol{\alpha}$ is as follows.

$$\max_{\boldsymbol{\alpha}}\big(\min_{\boldsymbol{w},b} L(\boldsymbol{w}, b, \boldsymbol{\alpha})\big) \tag{3.16}$$

where $L(\boldsymbol{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}\|\boldsymbol{w}\|^2 - \sum_{i=1}^{n} \alpha_i\big(y_i\big((\boldsymbol{w} \cdot \boldsymbol{x}_i) + b\big) - 1\big)$.

The Lagrange function $L(\boldsymbol{w}, b, \boldsymbol{\alpha})$ is convex. With the strong duality condition, the primal and the dual optimal solution is the same in this case. One can solve this dual problem instead of the primal. For formulating the dual problem, the derivatives of the Lagrangian function are follows.

$$\frac{\partial L(\boldsymbol{w},b,\boldsymbol{\alpha})}{\partial b} = 0 \text{ leads to } \sum_{i=1}^{n} \alpha_i y_i = 0, \tag{3.17}$$

$$\frac{\partial L(\boldsymbol{w},b,\boldsymbol{\alpha})}{\partial \boldsymbol{w}} = 0 \text{ leads to } \boldsymbol{w} = \sum_{i=1}^{n} \alpha_i y_i \boldsymbol{x}_i \tag{3.18}$$

The dual problem can be written as follows.

$$\max_{\boldsymbol{\alpha}} -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j (\boldsymbol{x}_i \cdot \boldsymbol{x}_j)\,(\boldsymbol{x}_i \cdot \boldsymbol{x}_j) + \sum_{i=1}^{n} \alpha_i \tag{3.19}$$

$$s.t. \quad \sum_{i=1}^{n} \alpha_i y_i = 0, i = 1, \dots, n \tag{3.20}$$

$$\alpha_i \geq 0, i = 1, \dots, n \tag{3.21}$$

The decision function is

$$f(\boldsymbol{x}) = \text{sign}(\sum_{i=1}^{n} \alpha_i y_i (\boldsymbol{x} \cdot \boldsymbol{x}_i) + b) \tag{3.22}$$

The problem is a quadratic programming problem with one equality constraint. The solution $\boldsymbol{\alpha}$ of this problem specifies the training patterns. The vectors $\boldsymbol{w}$ corresponding to the non-zero elements of $\boldsymbol{\alpha}$ are called support vectors which only effects to form the separating hyperplane, which also means a subset of constraints in the primal problem play a role to make the classifier.

In the SVM, the input data is mapped to a higher dimensional space called as the feature space. A nonlinear mapping function $\varphi(\boldsymbol{x})$ maps the input data to the feature space. The kernel function is defined as follows. $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \varphi(\boldsymbol{x}_i) \cdot \varphi(\boldsymbol{x}_j)$. All kernel functions can be expressed with dot products of $(\boldsymbol{x}_i \cdot \boldsymbol{x}_j)$. Therefore, the dual problem can be rewritten as follows.

$$\max_{\boldsymbol{\alpha}} -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j K(\boldsymbol{x}_i, \boldsymbol{x}_j) K(\boldsymbol{x}_i, \boldsymbol{x}_j) + \sum_{i=1}^{n} \alpha_i \tag{3.23}$$

$$s.t. \quad \sum_{i=1}^{n} \alpha_i y_i = 0, i = 1, \dots, n \tag{3.33}$$

$$\alpha_i \geq 0, i = 1, \dots, n \tag{3.34}$$

Then, the decision function becomes $f(\boldsymbol{x}) = \text{sign}\left(\sum_{i=1}^{n} \alpha_i y_i K(\boldsymbol{x}, \boldsymbol{x}_j) + b\right)$. There are several types of kernel function used in the SVM:

Linear kernel : $K(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{x} \cdot \boldsymbol{y}$

Polynomial kernel : $K(\boldsymbol{x}, \boldsymbol{y}) = (\boldsymbol{x} \cdot \boldsymbol{y} + 1)^d$

Radial basis function (RBF) kernel : $K(\boldsymbol{x}, \boldsymbol{y}) = e^{-\|\boldsymbol{x}-\boldsymbol{y}\|^2/2\sigma^2}$

Two layer neural network kernel : $K(\boldsymbol{x},\boldsymbol{y}) = \tanh(\kappa \boldsymbol{x} \cdot \boldsymbol{y} - \delta)$.

These kernels are expressed as the dot product between two vectors. Using the kernel function, one does not need to worry about the high dimensionality of the feature space. The input data is implicitly mapped to the feature space. Therefore, the dimension of the kernel Hessian matrix in the objective function is the same dimension as the linear kernel even if other kernels are used.

In the real world, the input data may not be separable with the separating hyperplane because the data can be inconsistent, missing, incomplete, noisy, and so on. To fix the non-separable case, one can introduce additional slack variables $\boldsymbol{\xi}$ in the primal problem:

$$y_i\big((\boldsymbol{w}\cdot\boldsymbol{x}_i)+b\big) \geq 1-\xi_i,\ i=1,\dots,n \tag{3.35}$$

The primal problem becomes

$$\min_{\boldsymbol{w},b,\boldsymbol{\xi}} \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{n}\xi_i \tag{3.36}$$

$$s.t.\quad y_i\big((\boldsymbol{w}\cdot\boldsymbol{x}_i)+b\big) \geq 1-\xi_i,\ i=1,\dots,n \tag{3.37}$$

$$\xi_i \geq 0,\ i=1,\dots,n \tag{3.38}$$

In this problem, $\frac{1}{2}\|\boldsymbol{w}\|^2$ represents the model complexity because it shows how much the classifier is accurate. $\sum_{i=1}^{n}\xi_i$ denotes the measure of the training errors, which can be seen as the empirical risk $R_{emp}(w)$. The constant C controls the trade-off between the complexity and the training errors. Since the slack variables $\boldsymbol{\xi}$ make the margin smooth, the margin is called as the soft margin and the problem is called as C support vector classification (C-SVC). The dual problem of this problem is as follows.

$$\max_{\boldsymbol{\alpha}} -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j K(\boldsymbol{x}_i,\boldsymbol{x}_j)K(\boldsymbol{x}_i,\boldsymbol{x}_j) + \sum_{i=1}^{n}\alpha_i \tag{3.39}$$

$$s.t.\quad \sum_{i=1}^{n}\alpha_i y_i = 0,\ i=1,\dots,n \tag{3.40}$$

$$0 \leq \alpha_i \leq C,\ i=1,\dots,n \tag{3.41}$$

The only difference from the separable case is that the dual problem has box constraints for all variables. However, this C-SVC has two broad range of the parameter C and the solution is very sensitive to the value C. To fix these problems, Schölkopf et al. (2000) proposed another model so called ν-support vector classification (ν-SVC) or ν-support vector machine (ν-SVM). ν-SVC uses a new parameter ν instead of C. The parameter ν has a range of zero to one, that is $\nu \in [0,1]$, and provides the lower bound of the fraction of the support vectors and the upper bound of the fraction of the margin errors. The primal problem of ν-SVC or ν-SVM is as follows.

$$\min_{\boldsymbol{w},b,\xi,\rho} \frac{1}{2}\|\boldsymbol{w}\|^2 - \nu\rho + \frac{1}{n}\sum_{i=1}^{n} \xi_i \tag{3.42}$$

$$s.t. \quad y_i\big((\boldsymbol{w}\cdot\boldsymbol{x}_i) + b\big) \geq \rho - \xi_i,\ i = 1, \dots, n \tag{3.43}$$

$$\xi_i \geq 0,\ i = 1, \dots, n \tag{3.44}$$

$$\rho \geq 0 \tag{3.45}$$

To see the function of the additional variable ρ, if the variable ξ_i equals zero, then the margin becomes $\frac{2\rho}{\|\boldsymbol{w}\|}$ instead of $\frac{2}{\|\boldsymbol{w}\|}$. The Lagrangian function with additional multipliers $\boldsymbol{\beta}$ and δ is as follows.

$$L(\boldsymbol{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}\|\boldsymbol{w}\|^2 - \sum_{i=1}^{n} \alpha_i\big(y_i\big((\boldsymbol{w}\cdot\boldsymbol{x}_i) + b\big) - \rho + \xi_i\big) + \sum_{i=1}^{n} \beta_i\xi_i - \delta\rho \tag{3.46}$$

With partial derivatives, one can get the following conditions:

$$\sum_{i=1}^{n} \alpha_i y_i = 0 \tag{3.47}$$

$$\boldsymbol{w} = \sum_{i=1}^{n} \alpha_i y_i \boldsymbol{x}_i \tag{3.48}$$

$$\alpha_i + \beta_i = \frac{1}{n} \tag{3.49}$$

$$\sum_{i=1}^{n} \alpha_i - \delta = \nu \tag{3.50}$$

Therefore, the dual problem of the ν-SVC or ν-SVM is as follows.

$$\max_{\boldsymbol{\alpha}} -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i\alpha_j y_i y_j K(\boldsymbol{x}_i, \boldsymbol{x}_j) K(\boldsymbol{x}_i, \boldsymbol{x}_j) \tag{3.51}$$

$$s.t. \quad \sum_{i=1}^{n} \alpha_i y_i = 0,\ i = 1, \dots, n \tag{3.52}$$

$$0 \leq \alpha_i \leq \frac{1}{n},\ i = 1, \dots, n \tag{3.53}$$

$$\sum_{i=1}^{n} \alpha_i \geq \nu,\ i = 1, \dots, n \tag{3.54}$$

Comparing with C-SVC, the objective function does not have the first order term $\sum_{i=1}^{n} \alpha_i$ and there is additional constraint. The decision function is the same as the previous one (3.22):

$$f(\boldsymbol{x}) = \text{sign}\big(\sum_{i=1}^{n} \alpha_i y_i K(\boldsymbol{x}, \boldsymbol{x}_j) + b\big) \tag{3.55}$$

For the calculation of b and ρ, one can use a KKT condition of the ν-SVC. One of KKT condition is as follows.

$$\alpha_i\big[y_i\big((\boldsymbol{w}\cdot\boldsymbol{x}_i) + b\big) - \rho + \xi_i\big] = 0,\ i = 1, \dots, n \tag{3.56}$$

By (3.48), the equation (3.56) can be rewritten as

$$\alpha_i\big[y_i\big(\sum_{\boldsymbol{j=1}}^{\boldsymbol{n}} \alpha_{\boldsymbol{j}} K(\boldsymbol{x}_i, \boldsymbol{x}_j) + b\big) - \rho + \xi_i\big] = 0,\ i = 1, \dots, n \tag{3.57}$$

At the optimal solution, if $\alpha_i \neq 0$, then $y_i\left(\sum_{\boldsymbol{j=1}}^{\boldsymbol{n}} \alpha_{\boldsymbol{j}} K\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right) + b\right) - \rho + \xi_i$ should be zero. Since $\xi_i \geq 0$, the slack variable $\xi_i = 0$. The remaining term $y_i\left(\sum_{\boldsymbol{j=1}}^{\boldsymbol{n}} \alpha_{\boldsymbol{j}} K\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right) + b\right) - \rho$ should be also zero. One can consider two cases of $y_i \in \{-1, 1\}$. Then two equalities are obtained as follows.

$$\rho - b = \sum_{i \in \{i: y_i = 1\}} \alpha_{\boldsymbol{j}} K\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right) \tag{3.58}$$

$$\rho + b = \sum_{i \in \{i: y_i = -1\}} \alpha_{\boldsymbol{j}} K\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right) \tag{3.59}$$

Therefore,

$$\rho = \frac{\sum_{i \in \{i: y_i = 1\}} \alpha_{\boldsymbol{j}} K\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right) + \sum_{i \in \{i: y_i = -1\}} \alpha_{\boldsymbol{j}} K\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right)}{2} \tag{3.60}$$

$$b = \frac{\sum_{i \in \{i: y_i = -1\}} \alpha_{\boldsymbol{j}} K\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right) - \sum_{i \in \{i: y_i = 1\}} \alpha_{\boldsymbol{j}} K\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right)}{2} \tag{3.61}$$

To solve SVM problems, it is necessary to solve the quadratic programming problem to find the decision function. However, in the SVM problems, Hessian matrix in the quadratic programming problem is dense and the size of the problem is large. Therefore, traditional optimization methods cannot be applied directly. Nonetheless, there are many approaches proposed so far for solving SVM problems.

Suykens and Vandewalle (1999) proposed the least squares support vector machine (LS-SVM) which is a function estimation problem. LS-SVM solves a set of linear system from Kuhn Tucker condition for training the data instead of solving a quadratic programming problem. Lee and Mangasarian (2001) proposed the reduced support vector machine (RSVM) that uses the reduced data set (about 1% out of the data) for training the data. The reduced data is chosen by the way that the distance between the data exceeded a certain tolerance. Zhan and Shen (2005) proposed an iterative method to reduce the size of support vectors so that the calculations of testing can be reduced. Kianmehr and Alhajj (2006) suggested an integrated method for the classification using the association rule based method and the SVM. The association rule based method generates the best set of rules from the data with the form that can be used in the support vector machine. The SVM is then used to classify the data.

Gradient projection based approaches are used to solve the SVM problems. To et al. (2001) proposed a method for solving SVM problem using space transformation method based on surjective space transformation introduced in Evtushenko and Zhadan (1994) and steepest descent method for solving the transformed problem. Serafini et al. (2005) proposed the generalized variable projection method which has a new step length rules. Dai and Fletcher (2006) suggested an efficient projected gradient algorithm to solve the singly linearly constrained quadratic programs with box constraints and tested some medium scale quadratic programs arising in the training the SVM.

A geometric approach also has been studied by Mavroforakis and Theodoridis (2006), Bennett and Bredensteiner (2000), Crisp and Burges (2000), Bennett and Bredensteiner (1997), and Zhou et al. (2002). In the geometric approach, if the data are separable, one can find two convex hulls for two classes of the data and the minimum distance line between two convex hulls. The separating hyperplane can be found as the hyperplane passing through the mid point of the minimum distance line and being orthogonal to the line. If the data are not separable, one can reduce the two convex hulls until they are separable, which is called as a reduced convex hull.

One of important issues in solving SVM problem is to solve a quadratic programming problem with dense Hessian matrix which is a positive semi-definite matrix. Due to the dense Hessian matrix, the decomposition method is essential to solve large scale problems in SVM. For example, if the problem has one thousand data points, then one need to have a storage of the matrix of $1{,}000 \times 1{,}000$ which means one million bytes size of units are required when one solves the problem with a computer program. Osuna et al. (1997) introduced a decomposition method that one solves smaller sized subproblems sequentially with some selected variables until the KKT condition of the original problem is satisfied. The set of variables selected in this type of approach is called as a working set. Joachims (1999) proposed an efficient decomposition method to shrink the size of the problem by fixing some variables to their optimal values. Platt (1999) described a new algorithm for training the SVM called Sequential Minimal Optimization (SMO). The size of the working set in SMO is only two. Since the working set is small, the algorithm does not require any quadratic programming solvers to solve the subproblem of the working set. In addition to that, it requires less matrix storage. Platt showed the SMO does particularly well for the sparse data sets. The SMO has been a popular method for the SVM.

Another approach to decompose this problem is to decompose the kernel Hessian matrix. The most concerning computational issue in solving the SVM problem is how to handle the dense kernel Hessian matrix. While SMO type method is trying to reduce the dimension of the problem and to solve subproblems with smaller variables, the matrix decomposition approach is trying to decompose the kernel Hessian matrix and to reduce computational burden with the same number of variables. There are various approaches for kernel matrix approximation such as spectral decomposition, incomplete Cholesky decomposition, tridiagonalization, Nyström method, and Fast Gauss Transform in Kashima et al. (2009).

In this chapter, this research focuses on the incomplete Cholesky decomposition (ICD) method. The ICD is used in solving the SVM problem with several ways. Bach and Jordan (2005), An et al. (2007), and Alzate and Suykens (2008) used the ICD for solving the LS-SVM problem. The interior point method has been used in Fine and Scheinberg (2001), Ferris and Munson (2003), and Goldfarb and Scheinberg (2008). The ICD has been used for solving the normal equation in the interior point method. Lin and Saigal (2000) used the ICD as a preconditioner in the SVM problem. Louradour et al. (2006) proposed a new kernel method using the ICD. Debnath and Takahashi (2006) suggested a solving method for the SVM using the second order cone programming and the ICD. Camps-Valls et al. (2009) used the ICD for solving the semi-superviesd SVM.

This research uses the projected gradient approach for solving the SVM problem. In this approach, the matrix splitting method is used for the projection and splitting the Hessian matrix. The ICD is used for reducing the computational burden and storage. Since most problems of the SVM have the dense Hessian matrix and are large scale, dimension reduction methods such as working set method or SMO type method can be attractive. However, traditional methods like Newton's method or gradient methods have advantages such as rapid local convergence. The main drawback of traditional methods is the size of the problem. If one can remove or reduce the curse of dimensionality, one may use advantages of traditional methods. With this motivation, this research uses the matrix splitting method and the ICD for reducing the computational and storage problem of handling the large scale problems. In addition, the Bitran-Hax or Dual Bound algorithm is used for solving the subproblem iteratively after splitting the Hessian matrix.

3.3 Solving approach for the ν-support vector machine

This section provides a new solving approach for ν-SVM problem (ν-SVC). The basic idea is to make the Hessian matrix simple using the matrix splitting method, and to solve the problem with the projected gradient and incomplete Cholesky decomposition methods.

The dual problem of ν-SVM is as follows.

$$\max_{\boldsymbol{\alpha}} -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j K(\boldsymbol{x}_i,\boldsymbol{x}_j)K(\boldsymbol{x}_i,\boldsymbol{x}_j) \tag{3.62}$$

$$s.t. \quad \sum_{i=1}^{n} \alpha_i y_i = 0, i = 1, \dots, n \tag{3.63}$$

$$0 \le \alpha_i \le \frac{1}{n}, i = 1, \dots, n \tag{3.64}$$

$$\sum_{i=1}^{n} \alpha_i \le \nu, i = 1, \dots, n \tag{3.65}$$

Crisp and Burges (2000), Chang and Lin (2001) proved the constraint (3.65) can be changed to an equality constraint. Changing the problem to a minimization problem, the problem can be rewritten as follows.

$$\min_{\boldsymbol{\alpha}} \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j K(\boldsymbol{x}_i, \boldsymbol{x}_j) K(\boldsymbol{x}_i, \boldsymbol{x}_j) \tag{3.66}$$

$$s.t. \quad \sum_{i=1}^{n} \alpha_i y_i = 0, i = 1, \dots, n \tag{3.67}$$

$$0 \le \alpha_i \le \frac{1}{n}, i = 1, \dots, n \tag{3.68}$$

$$\sum_{i=1}^{n} \alpha_i = \nu, i = 1, \dots, n \tag{3.69}$$

The scaled and vector version of the problem is as follows.

$$\min_{\boldsymbol{\alpha}} \frac{1}{2} \boldsymbol{\alpha}^T Q \boldsymbol{\alpha} \tag{3.70}$$

$$s.t. \quad \boldsymbol{y}^T \boldsymbol{\alpha} = 0, \tag{3.71}$$

$$0 \le \boldsymbol{\alpha} \le 1, \tag{3.72}$$

$$\boldsymbol{e}^T \boldsymbol{\alpha} = \nu n \tag{3.73}$$

This problem is a quadratic programming problem with two equality constraints and box constraints on variables. If one removes the last equality constraint from this problem, the problem becomes a singly linearly constrained convex quadratic problem which is well known to be able to applied many practical applications introduced in Pardalos and Kovoor (1990), Dai and Fletcher (2006), Lin et al. (2009). In this thesis, the augmented Lagrangian method is used to remove the last constraint and put it on the objective function as follows.

$$\min_{\boldsymbol{\alpha}} \frac{1}{2} \boldsymbol{\alpha}^T Q \boldsymbol{\alpha} + w(\boldsymbol{e}^T \boldsymbol{\alpha} - \nu n) + \mu(\boldsymbol{e}^T \boldsymbol{\alpha} - \nu n)^2 \tag{3.74}$$

$$s.t. \quad \boldsymbol{y}^T \boldsymbol{\alpha} = 0, \tag{3.75}$$

$$0 \le \boldsymbol{\alpha} \le 1 \tag{3.76}$$

The parameter w and μ denote the Lagrangian multiplier and the penalty parameter respectively. The problem can be reorganized as follows.

$$\min_{\boldsymbol{\alpha}} \frac{1}{2} \boldsymbol{\alpha}^T H \boldsymbol{\alpha} + (w - 2\mu\nu n) \boldsymbol{e}^T \boldsymbol{\alpha} \tag{3.77}$$

$$s.t. \quad \boldsymbol{y}^T \boldsymbol{\alpha} = 0, \tag{3.78}$$

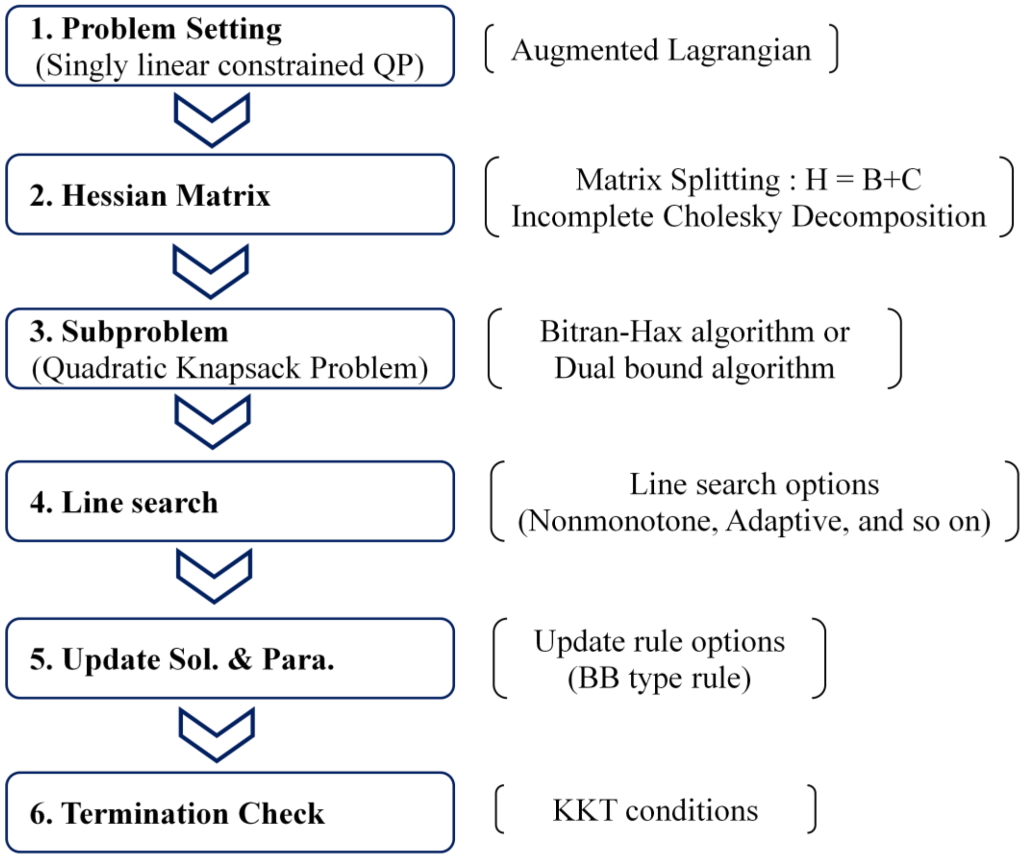
$$0 \le \boldsymbol{\alpha} \le 1 \tag{3.79}$$

where $H = Q + \mu U$, U is a $n \times n$ matrix which all elements are one.

The constant term $\mu(\nu n)^2 - w\nu n$ in the objective function can be removed because it does not affect the solution. Now, the problem is a singly linearly constrained convex quadratic problem. The Hessian matrix H is dense, symmetric, and positive semi-definite matrix. Since the Hessian matrix H is not diagonal and large in the SVM, a standard solving method for the general quadratic programming problem can be applied to small or medium size problems. Therefore, a specified method which can handle the dense Hessian matrix and the large scale problem is essential for solving this problem. In this research, the algorithm uses matrix splitting method with nonmonotone line search and gradient projection method. In addition, the incomplete Cholesky decomposition method is used for handling large scale problems. The structure of the solution procedure is as follows.

Solving Procedure

Figure 3.1 Solving Procedure for the ν-SVM



The original problem becomes a singly linearly constrained convex quadratic problem with Augmented Lagrangian method by putting one of constraints to the objective function. The Hessian matrix is split into the sum of two matrices by the matrix splitting method. In addition to that, the incomplete Cholesky decomposition is performed for the Hessian matrix to facilitate the calculation of the Hessian matrix and the variable vectors. The method solves the subproblem that has a simple Hessian matrix in procedure 3.

The Bitran-Hax or Dual bound method is used to solve this subproblem which is a separable continuous quadratic knapsack problem. The direction vector is calculated with the solution of the subproblem and the current solution. The line search technique finds the step length along the direction in procedure 4. There is a type of scale parameter where the coefficient of the linear term of the subproblem in procedure 3. The solution of the problem and the scale parameter are updated in procedure 5. In procedure 6, the termination is checked with KKT conditions. For the procedure 4 and 5, the algorithm can use several options that will be described in the later section.

3.3.1 Matrix splitting with Gradient Projection method

In this section, the matrix splitting method is described. The problem (3.77) ~ (3.79) can be rewritten as follows. For the convenience, this research uses the term $\boldsymbol{x}$ instead of $\boldsymbol{\alpha}$ and simple terms.

$$\text{(MP)} \qquad \min_{\boldsymbol{x}} \frac{1}{2}\boldsymbol{x}^T H\boldsymbol{x} + \boldsymbol{c}^T\boldsymbol{x} \tag{3.80}$$

$$s.t. \quad \boldsymbol{a}^T\boldsymbol{x} = b, \tag{3.81}$$

$$0 \le \boldsymbol{x} \le 1 \tag{3.82}$$

where H is a $n \times n$ positive semi-definite matrix.

The matrix splitting method is an iterative method to solve the quadratic programming problem. The most concern of the quadratic problem in the SVM is about the Hessian matrix. The dense and large Hessian matrix needs a large space for the storage and a lot of burden for the calculation. The matrix splitting method splits the Hessian matrix into the sum of two matrices that have certain properties. The properties are based on the P-regular matrix splitting in Ortega (1972), Keller (1965), Lin and Pang (1987) as follows.

Definition 3.1

For a given matrix A, a splitting $A = M + N$ with M nonsingular is called P-regular splitting if the matrix $M - N$ is positive definite. Then the splitting iterative method is convergent: $\rho(M^{-1}N) < 1$, where $\rho(A)$ denotes the spectral radius of a matrix A and $M^{-1}N$ is called a complementarity matrix.

P-regular splitting has been used for solving the linear systems such as linear complementarity problem (LCP). Pang (1982) proposed an iterative method for LCP using P-regular splitting. Luo and Tseng (1992) established the linear convergence for the matrix splitting method and analyzed error bound for the LCP problem. The matrix splitting method for LCP requires solving a subproblem using successive overrelaxation (SOR) and calculates the complementarity matrix at each iteration. Therefore, the matrix splitting method cannot be directly applied to the SVM problem because the problem has a large and dense Hessian matrix. In his dissertation, Nehate (2006) modified the matrix splitting method for the SVM problem. The new method in this research is based on this modified matrix splitting method which combines with the gradient projection method. In this method, the original Hessian matrix H is splitted into the sum of two matrices $(B + C)$ using P-regular splitting. The matrix B is chosen to be a simple nonsingular matrix. In this research, the identity matrix is used. Then a subproblem is formulated with B matrix as the Hessian. In the subproblem, the coefficient of the linear term in the objective function is a little different, but the constraints are the same as the original problem. The method solves the subproblem iteratively until it has the optimal solution. The detail algorithm is as follows.

Main Algorithm

Step 1 Initialization

Let $(B,\ H - B)$ be a splitting of the Hessian matrix H and

$\boldsymbol{x}^0$ be a feasible initial solution. Let B be an identity matrix,

$0 < \alpha_{min} < \alpha_{max}$, $\alpha^0 \in [\alpha_{min}, \alpha_{max}]$. Set $k = 0$.

Perform the Incomplete Cholesky decomposition for the Hessian matrix :

$H \cong LL^T$, where L is a lower triangular matrix.

Step 2 Solving the subproblem

Solve

$\text{Min}_z \frac{1}{2}\boldsymbol{z}^T B\boldsymbol{z} + \boldsymbol{q}^T\boldsymbol{z}$

$s.t. \quad \boldsymbol{a}^T\boldsymbol{z} = b,$

$\quad\quad 0 \leq \boldsymbol{z} \leq 1$

where $\boldsymbol{q} = \alpha^k\boldsymbol{g}^k - B\boldsymbol{x}^k,\ \boldsymbol{g}^k = LL^T\boldsymbol{x}^k + \boldsymbol{c}$.

Bitran-Hax algorithm or Dual bound algorithm is used.

Direction vector is calculated as $\boldsymbol{d} = \boldsymbol{z} - \boldsymbol{x}^k$.

Step 3 Line search

$\boldsymbol{x}^{k+1} = \boldsymbol{x}^k + \lambda\boldsymbol{d}^k$, Find λ^* using line search techniques.

Step 4 Update solution

Update solution $\boldsymbol{x}^{k+1} = \boldsymbol{x}^k + \lambda^*\boldsymbol{d}^k$,

Calculate $\boldsymbol{\alpha}^{k+1}$ using Brailai-Borwein (BB) type methods.

Step 5 Check termination

If some appropriate stopping rule is satisfied, stop.

else set $k := k + 1$ and go to Step 2.

Compared with Nehate's method, the newly proposed solving algorithm uses the incomplete Cholesky decomposition method for the Hessian matrix in step 1 and tries several methods for the line search and the updating the parameter value in step 3 and 4. In step 2, the parameter α plays a role in this algorithm as Nehate (2006) described. The algorithm solves the subproblem instead of the original problem at each iteration using the Bitran-Hax or Dual Bound algorithm. The value of α is changed iteratively. This parameter rescales the gradient and makes the subproblem to be closer to the original problem. If $\alpha = 1$, the algorithm is just the matrix splitting method. If $\alpha > 1$, then the algorithm is the combination of the matrix splitting method and the gradient projection method. If the parameter α increases, the algorithm is getting closer to the gradient projection method.

The parameter α can be obtained by solving a problem to minimize the gap between the sequential gradients which are the kth gradient and the $k-1$th gradient. The α at the kth iteration is derived as follows. The details for the derivation are in Nehate (2006).

$$\alpha^k = \frac{(s^k)^T B s^k}{(s^k)^T y^k} \tag{3.83}$$

where $\boldsymbol{s}^k = \boldsymbol{x}^k - \boldsymbol{x}^{k-1}$, $\boldsymbol{y}^k = g(\boldsymbol{x}^k) - g(\boldsymbol{x}^{k-1})$, $g(\cdot)$ is the gradient of the objective function.

The equation (3.83) is called as the two point step size gradient method or Barzilai and Borwein (BB) type rule. In this research, the new algorithm uses several other methods for calculating α^k.

The line search in step 3 determines the best step size λ to get the solution of the original problem forward to the optimal solution. The new algorithm also uses several line search methods. The next section describes the details for the methods in step 3 and 4.

3.3.2 Line search and update parameter α methods

This section presents some methods for the line search and updating α in the algorithm proposed in the previous section. The problem in this research is a singly linearly constrained quadratic convex problem. Due to its numerous applications, there have been many studies for this problem such as Pardalos and Kovoor (1990), Dai and Fletcher (2006), Lin et al. (2009), Fu and Dai (2010), and so on. This research uses four methods that applied for the SVM problem. The details for the methods are as follows.

3.3.2.1 SPGM (Spectral Projected Gradient Methods)

Birgin et al. (2000) proposed a solving algorithm which extended the classical projected gradient method to use additional methods including the nonmonotone line search technique and the spectral step length known as the BB type rule. The algorithm was proposed for the problem of the minimization of differentiable functions on nonempty closed and convex sets. In their paper, Birgin et al. (2000) proved the convergence of the algorithm and showed good experimental results comparing to the LANCELOT package in Conn et al. (1988). The detail algorithm for the line search and the updating method are as follows.

SPGM Algorithm

Step 1 Initialization

Calculate direction $\boldsymbol{d}^k$ and set $\lambda = 1$

Step 2 Set new value

Set $\boldsymbol{x}^+ = \boldsymbol{x}^k + \lambda \boldsymbol{d}^k$

Step 3 Line search

If $f(\boldsymbol{x}^+) \leq \max_{0 \leq j \leq min\{k, M-1\}} f\left(\boldsymbol{x}^{k-j}\right) + \lambda \langle \boldsymbol{d}^k, g(\boldsymbol{x}^k) \rangle$, then define $\lambda^k = \lambda$, $\boldsymbol{x}^{k+1} = \boldsymbol{x}^+$, $\boldsymbol{s}^k = \boldsymbol{x}^{k+1} - \boldsymbol{x}^k$, $\boldsymbol{y}^k = g(\boldsymbol{x}^{k+1}) - g(\boldsymbol{x}^k)$, and go to step 4.

else define $\lambda_{new} \in [\sigma_1\lambda, \sigma_2\lambda]$ and set $\lambda = \lambda_{new}$ and go to step 2.

Step 4 Update parameter

Calculate $b^k = \langle \boldsymbol{s}^k, \boldsymbol{y}^k \rangle$

If $b^k \leq 0$, then set $\alpha^{k+1} = \alpha_{max}$,

else calculate $a^k = \langle \boldsymbol{s}^k, \boldsymbol{s}^k \rangle$ and $\alpha^{k+1} = \min\left\{\alpha_{\max}, \max\left\{\alpha_{\min}, \frac{a^k}{b^k}\right\}\right\}$

The line search in step 3 is the nonmonotone line search which means the objective function value is allowed to increase on some iterations. The calculation of λ_{new} is from one dimensional quadratic interpolation. If the minimum of the one dimensional quadratic lies outside $[\sigma_1\lambda, \sigma_2\lambda]$, then the algorithm sets $\lambda \leftarrow \frac{1}{2}\lambda$. The parameter σ_1 and σ_2 are fixed constants. The method for updating α^k is BB rule. Birgin et al. (2000) showed that the use of the parameter α^k is more important than line search method to improve the performance of the algorithm in their experimental results.

3.3.2.2 GVPM (Generalized Variable Projection method)

Serafini et al. (2005) proposed a generalized version of variable projection method (GVPM) using a new adaptive steplength alternating rule. Their research has compared the GVPM with SPGM described in the previous section. The GVPM focuses on the method of updating parameter α^k. The algorithm uses two steplength rules adaptively and a limited minimization rule as a line search method. The two types of steplength rules are as follows.

$$\alpha_{k+1}^1 = \frac{(\boldsymbol{d}^k)^T \boldsymbol{d}^k}{(\boldsymbol{d}^k)^T H \boldsymbol{d}^k} \tag{3.84}$$

$$\alpha_{k+1}^2 = \frac{(\boldsymbol{d}^k)^T H \boldsymbol{d}^k}{(\boldsymbol{d}^k)^T H^2 \boldsymbol{d}^k} \tag{3.85}$$

The algorithm switches the rules (3.84) and (3.85) if certain criteria are satisfied. Let $0 < n_{min} < n_{max}$ are fixed constants. The number n_α denotes the number of iterations that use the same steplength rule. There are two definitions for this algorithm as follows.

Definition 3.2

Let $\boldsymbol{x}^k \in \Omega$(feasible set) and $(\boldsymbol{d}^k)^T H \boldsymbol{d}^k > 0$ and $\alpha^k \in [\alpha_{min}, \alpha_{max}]$.

If $\alpha_{k+1}^2 < \alpha^k < \alpha_{k+1}^1$, then α^k is called a separating steplength.

Definition 3.3

Let $\boldsymbol{x}^k \in \Omega$(feasible set) and $(\boldsymbol{d}^k)^T H \boldsymbol{d}^k > 0$, $\alpha^k \in [\alpha_{min}, \alpha_{max}]$,

and $\lambda_{opt} = \arg\min_{\lambda \in [0,1]} f(\boldsymbol{x}^k + \lambda \boldsymbol{d}^k) = \frac{-\left(g(\boldsymbol{x}^k)\right)^T \boldsymbol{d}^k}{\left(\boldsymbol{d}^k\right)^T H \boldsymbol{d}^k}$.

Given two constants λ_l and λ_u such that $0 < \lambda_l \leq 1 \leq \lambda_u$, α^k is called a bad descent generator if one of following conditions is satisfied:

$$\lambda_{opt} < \lambda_l \text{ and } \alpha^k = \alpha_{k+1}^1 \qquad (3.86)$$

$$\lambda_{opt} > \lambda_u \text{ and } \alpha^k = \alpha_{k+1}^2 \qquad (3.87)$$

The detail algorithm is as follows.

GVPM algorithm

```
Step 1  Initialization
        Calculate direction $\boldsymbol{d}^k$
Step 2  Line search
        Calculate $\boldsymbol{x}^{k+1} = \boldsymbol{x}^k + \lambda^k \boldsymbol{d}^k$, $\lambda^k \in (0,1]$
        with $\lambda^k$ given by $\lambda^k = \arg\min_{\lambda \in [0,1]} f(\boldsymbol{x}^k + \lambda \boldsymbol{d}^k) = \frac{-\left(g(\boldsymbol{x}^k)\right)^T \boldsymbol{d}^k}{\left(\boldsymbol{d}^k\right)^T H \boldsymbol{d}^k}$.
Step 3 Update parameter
        If $(\boldsymbol{d}^k)^T H \boldsymbol{d}^k \leq 0$, then set $\alpha^{k+1} = \alpha_{max}$,
        else calculate $\alpha_{k+1}^1$ and $\alpha_{k+1}^2$ from (3.78) and (3.79)
          If $n_\alpha \geq n_{min}$,
            then if $n_\alpha \geq n_{max}$ or $\alpha^k$ is a separating steplength or a bad descent generator,
              then set $i_\alpha = mod(i_\alpha.2) + 1$, $n_\alpha = 0$.
          Calculate $\alpha^{k+1} = \min\left\{\alpha_{\max}, \max\left\{\alpha_{\min}, \alpha_{k+1}^{i_\alpha}\right\}\right\}$
        Set $k = k + 1$, $n_\alpha = n_\alpha + 1$ and go to step 2
```

The key idea of this algorithm is to switch the steplength rules with certain criteria. Serafini et al. (2005) showed that this adaptive steplength change played an important role to perform well in the experimental results comparing with the SPGM and the VPM. For the SVM problem, Serafini et al. (2005) applied this algorithm for solving the subproblem of the large scale SVM problems while the SVM problem was solved by the SMO type algorithm.

3.3.2.3 MSM (Matrix Splitting Method)

Nehate (2006) proposed several efficient solving algorithms for the SVM problems. One of the algorithms uses the matrix splitting method combined with gradient projection method. The parameter α^k plays a role to focus on the algorithm forward either the matrix splitting or the gradient projection method. Nehate (2006) used a simple line search and updating method in the algorithm. The detail is as follows.

MSM algorithm

Step 1 Initialization

Calculate direction $\boldsymbol{d}^k$ and set $\lambda = 1$

Step 2 Set new value

Set $\boldsymbol{x}^+ = \boldsymbol{x}^k + \lambda \boldsymbol{d}^k$

Step 3 Line search

If $f(\boldsymbol{x}^+) \leq \max_{0 \leq j \leq 2} f\left(\boldsymbol{x}^{k-j}\right) + \lambda \langle \boldsymbol{d}^k, g(\boldsymbol{x}^k) \rangle$,

then define $\lambda^k = \lambda$,

else $\lambda^k = \frac{-\left(g(\boldsymbol{x}^k)\right)^T \boldsymbol{d}^k}{\left\|(\boldsymbol{d}^k)^T H \boldsymbol{d}^k\right\|} = \arg\min_{\lambda \in [0,1]} f(\boldsymbol{x}^k + \lambda \boldsymbol{d}^k)$, $0 < \lambda^k < 1$.

Step 4 Update solution and parameter

$\boldsymbol{x}^{k+1} = \boldsymbol{x}^k + \lambda \boldsymbol{d}^k$,

$\boldsymbol{s}^k = \boldsymbol{x}^{k+1} - \boldsymbol{x}^k$, $\boldsymbol{y}^k = g(\boldsymbol{x}^{k+1}) - g(\boldsymbol{x}^k)$,

$\alpha^{k+1} = \frac{(\boldsymbol{s}^k)^T B \boldsymbol{s}^k}{(\boldsymbol{s}^k)^T \boldsymbol{y}^k}$,

$\alpha^{k+1} = \min\left\{\alpha_{\max}, \max\left\{\alpha_{\min}, \frac{a^k}{b^k}\right\}\right\}$.

Nehate (2006) showed that the use of the parameter α^k speeds up the convergence of the algorithm, but it makes the algorithm nonmonotone. Therefore the algorithm uses the nonmonotone line search technique. In step 3, the line search simply assigns the λ^k value and uses the range of index j from zero to two. This simple assignment is very useful to solve the large scale problems. Nehate (2006) focuses on the SVM for regression problem. As the other methods, the algorithm can solve up to medium sized SVM problems.

3.3.2.4 AA (Adaptive step size method and Alternate step length (alpha) method)

Dai and Zhang (2001) suggested an adaptive nonmonotone line search method. The nonmonotone line search such as $f(\boldsymbol{x}^{+}) \leq \max_{0 \leq j \leq min\{k,M-1\}} f\left(\boldsymbol{x}^{k-j}\right) + \lambda\langle \boldsymbol{d}^{k}, g(\boldsymbol{x}^{k})\rangle$ in SPGM has a fixed integer M. The performance of the algorithm depends on the choice of M described in Raydan (1997). To resolve this problem, Dai and Zhang (2001) uses a new method to change the value of M adaptively. Let f_{min} be the current minimum objective value over all past iterations and f_{max} be the maximum objective value in recent M iterations. These are denoted as follows.

$$f_{min} = \min_{0 \leq i \leq k} f(\boldsymbol{x}_i) \tag{3.88}$$

$$f_{max} = \max_{0 \leq i \leq min\{k,M-1\}} f(\boldsymbol{x}_{k-i}) \tag{3.89}$$

The value l denotes the number of iterations since the f_{min} is obtained and L is a fixed constant. The value p denotes the largest integer such that $\left\{\lambda_1^{k-i} : i = 1, \dots, p\right\}$ are accepted but λ_1^{k-p-1}not and λ_1^{k} is the first trial step size at the kth iteration and P denotes a fixed constant. The values $\gamma_1 \geq 1$ and $\gamma_2 \geq 1$ are fixed constants.

Dai and Fletcher (2006) proposed an efficient gradient projection method using a new formula for updating the parameter. The algorithm uses the adaptive line search method proposed by Dai and Zhang (2001). Dai and Fletcher (2006) introduced a new formula for updating the α^k. As the previous section defined, let $\boldsymbol{s}^k = \boldsymbol{x}^{k+1} - \boldsymbol{x}^k$, $\boldsymbol{y}^k = g(\boldsymbol{x}^{k+1}) - g(\boldsymbol{x}^k)$. The BB rule can be as follows.

$$\alpha^{k+1} = \frac{\left(s^k\right)^T s^k}{\left(s^k\right)^T y^k} \tag{3.90}$$

This formula can be obtained by solving a one dimensional problem as follows.

$$\min\|\alpha_{k+1}^{-1}\boldsymbol{s}^k - \boldsymbol{y}^k\|_2 \tag{3.91}$$

If one replaces the pair $(\boldsymbol{s}^k, \boldsymbol{y}^k)$ with $\left(\boldsymbol{s}^{k-i}, \boldsymbol{y}^{k-i}\right)$ for each integer $i \geq 0$, then the similar formula can be obtained as follows as described in Friedlander et al. (1998).

$$\alpha^{k+1} = \frac{\left(S^k\right)^T S^k}{\left(S^k\right)^T Y^k} = \frac{\sum_{i=0}^{m-1}\left(\boldsymbol{s}^{k-i}\right)^T \boldsymbol{s}^{k-i}}{\sum_{i=0}^{m-1}\left(\boldsymbol{s}^{k-i}\right)^T \boldsymbol{y}^{k-i}} \tag{3.92}$$

where $S^k = ((\boldsymbol{s}^k)^T, \dots, (\boldsymbol{s}^{k-m+1})^T)^T$ and $Y^k = ((\boldsymbol{y}^k)^T, \dots, (\boldsymbol{y}^{k-m+1})^T)^T$.

In the formula (3.92), if $m = 1$, then the formula is the same as the BB rule in (3.90). Dai and Fletcher (2006) found the best value of m is 2 with their experimental results.

In this research, the algorithm combines the adaptive steplength algorithm from Dai and Zhang (2001) with the alternative updating formula from Dai and Fletcher (2006) and calls this method as AA (adaptive and alternative) algorithm. Combining these two algorithms, the line search is the adaptive nonmonotone line search and the updating the parameter α^k is the method of alternative formula (3.92). The detail algorithm is as follows.

AA algorithm

```
Step 1  Initialization
    Calculate direction $\boldsymbol{d}^k$ and set $\lambda = 1$,
    Set $l = 0, p = 0, f_{min} = f_r = f_c = f(\boldsymbol{x}^0)$
Step 2  Line search
    Step 2.1  Reset reference value
      If $l = L$,
        then set $l = 0$ and calculate $f_r = \begin{cases} f_c, & \text{if } \frac{f_{max} - f_{min}}{f_c - f_{min}} > \gamma_1 \\ f_{max}, & \text{otherwise} \end{cases}$
      If $p > P$,
        then calculate $f_r = \begin{cases} f_{max}, & \text{if } \mathrm{f_{max}} > f(\boldsymbol{x}^k) \text{ and } \frac{f_r - f(\boldsymbol{x}^k)}{f_{max} - f(\boldsymbol{x}^k)} \geq \gamma_2 \\ f_r, & \text{otherwise} \end{cases}$
    Step 2.2  Test first trial step size
      If $f(\boldsymbol{x}^k + \lambda_1^k \boldsymbol{d}^k) \leq f_r + \delta \lambda_1^k \langle \boldsymbol{d}^k, g(\boldsymbol{x}^k) \rangle$
        then let $\lambda^k = \lambda_1^k$, $p = p + 1$, and go to step 2.4
      else $p = 0$
    Step 2.3  Test other trial step sizes
      Set $\lambda_{old} = \lambda_1^k$
      Calculate $\lambda_{new} \in [\sigma_1 \lambda_{old}, \sigma_2 \lambda_{old}]$
      If $f(\boldsymbol{x}^k + \lambda_{new} \boldsymbol{d}^k) \leq \min\{f_{max}, f_r\} + \delta \lambda_{new} \langle \boldsymbol{d}^k, g(\boldsymbol{x}^k) \rangle$,
        then set $\lambda^k = \lambda_{new}$ and go to step 2.4
      else set $\lambda_{old} = \lambda_{new}$ and repeat step 2.3
    Step 2.4
        If $f(\boldsymbol{x}^{k+1}) < f_{min}$,
            then set $f_c = f_{min} = f(\boldsymbol{x}^{k+1})$ and $l = 0$.
        else $l = l + 1$.
        If $f(\boldsymbol{x}^{k+1}) > f_c$,
            then set $f_c = f(\boldsymbol{x}^{k+1})$.
        Calculate $f_{max}$ from (3.83).
Step 3 Update parameter
    $\boldsymbol{s}^k = \boldsymbol{x}^{k+1} - \boldsymbol{x}^k, \boldsymbol{y}^k = g(\boldsymbol{x}^{k+1}) - g(\boldsymbol{x}^k)$,
    If $(\boldsymbol{s}^k)^T \boldsymbol{y}^k \leq 0$,
        then $\alpha^{k+1} = \alpha_{max}$
    else $\alpha^{k+1} = \min\left\{\alpha_{\max}, \max\left\{\alpha_{\min}, \frac{\sum_{i=0}^{m-1}(\boldsymbol{s}^{k-i})^T \boldsymbol{s}^{k-i}}{\sum_{i=0}^{m-1}(\boldsymbol{s}^{k-i})^T \boldsymbol{y}^{k-i}}\right\}\right\}$
```

The value of λ_{new} is obtained by a quadratic interpolation method in step 2.3. Dai and Fletcher (2006) tested some medium sized SVM problem with their new algorithm and obtaining good results.

From these sections 3.3.2.1 ~ 3.3.2.4, four different line search and updating methods were reviewed for the gradient method. The new solving algorithm can take one of the methods for step 3 and 4 in the main algorithm in section 3.3.1. As it can be seen in these four methods, since the test problems are limited to the medium size for the SVM problems, the methods cannot be directly applied to the large scale problems. In these four algorithms, the SMO type or other methods should be used for large scale problems and the four methods are used for solving the subproblem within the solution procedure. To overcome this issue, this research proposes a new solving approach to apply these algorithms directly to large scale problems. The method proposed in this research is the incomplete Cholesky decomposition method. The details will be described in the next section.

3.3.3 Incomplete Cholesky decomposition

Cholesky A. developed a method for solving a linear system in 1910. The method has been published by Jensen H. in 1944. The Cholesky factorization or decomposition named after his name. More than sixty years now, the Cholesky decomposition method has been an attractive method for solving large scale systems. If a matrix A is a symmetric and positive definite, then A can be expressed as $A = LL^T$ and L is a lower triangular matrix with positive diagonal elements. The elements of L are called as Cholesky factors. The Cholesky factors can be obtained as a simple way of the following example.

Let $A = \begin{bmatrix} a_{11} & a_{21} \\ a_{21} & a_{22} \end{bmatrix}$, $L = \begin{bmatrix} l_{11} & 0 \\ l_{21} & l_{22} \end{bmatrix}$, and $L^T = \begin{bmatrix} l_{11} & l_{21} \\ 0 & l_{22} \end{bmatrix}$.

By the definition of the Cholesky decomposition $A = LL^T$,

$$\begin{bmatrix} a_{11} & a_{21} \\ a_{21} & a_{22} \end{bmatrix} = \begin{bmatrix} l_{11} & 0 \\ l_{21} & l_{22} \end{bmatrix} \begin{bmatrix} l_{11} & l_{21} \\ 0 & l_{22} \end{bmatrix} = \begin{bmatrix} (l_{11})^2 & l_{11}l_{21} \\ l_{11}l_{21} & (l_{21})^2 + (l_{22})^2 \end{bmatrix} \quad (3.93)$$

Therefore,

$$l_{11} = \sqrt{a_{11}} \quad \text{from} \quad a_{11} = (l_{11})^2, \quad (3.94)$$

$$l_{21} = a_{21}/\sqrt{a_{11}} \quad \text{from} \quad a_{21} = l_{11}l_{21}, \quad (3.95)$$

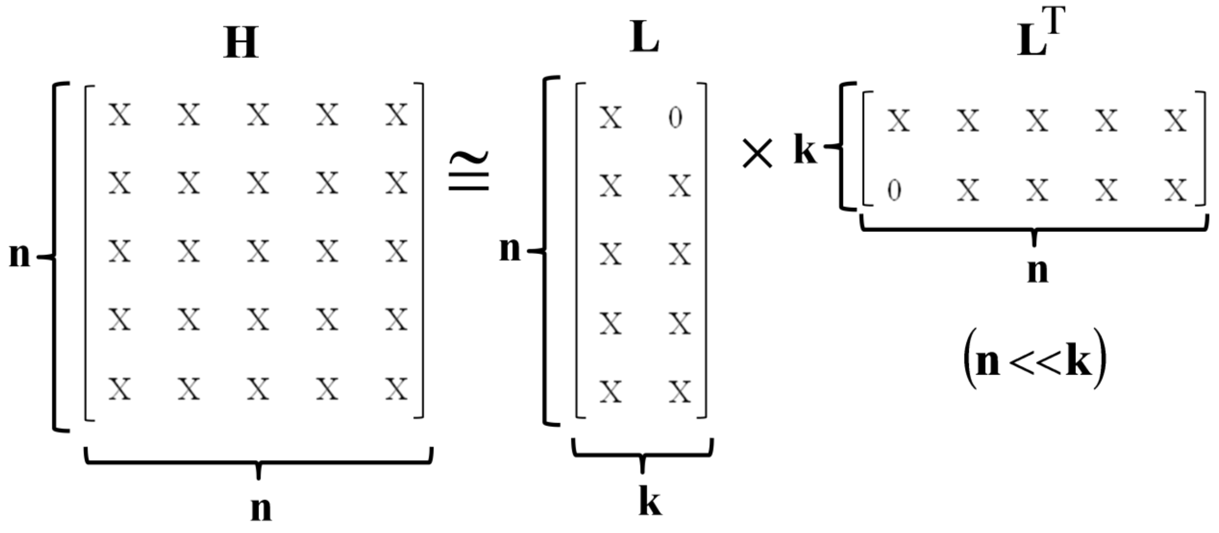
$$l_{21} = \sqrt{a_{22} - (a_{21})^2/a_{11}} \quad \text{from} \quad a_{22} = (l_{21})^2 + (l_{22})^2. \quad (3.96)$$

Practically, the Cholesky decomposition is implemented by the way of the above example. The linear system $A\boldsymbol{x} = \boldsymbol{b}$ can be solved by using the Cholesky decomposition. The problem can be rewritten as $LL^T\boldsymbol{x} = \boldsymbol{b}$. Two substitutions are needed to solve this system. The forward substitution $L\boldsymbol{y} = \boldsymbol{b}$ is calculated first. Then the backward substitution $L^T\boldsymbol{x} = \boldsymbol{y}$ is calculated. However, this solving procedure can be used only if the matrix A is symmetric and positive definite. If the matrix A is a symmetric and positive semi-definite, some of diagonal values of L are zero. If any diagonal elements of L are zero, the rest of Cholesky factors cannot be calculated any more. In the above example, if $l_{11} = 0$ in (3.94), l_{21} cannot be calculated in (3.95) because the denominator $\sqrt{a_{11}}$ is zero. In this case, the incomplete Cholesky decomposition method can be used.

The incomplete Cholesky decomposition is the method for the symmetric and positive semi-definite matrix. During the Cholesky decomposition procedure, when one encounters a diagonal element of L that is zero, the procedure cannot progress. The idea of the incomplete Cholesky decomposition is that a permutation is performed to move the largest diagonal element on the active pivot position so that the decomposition procedure can stop when the largest diagonal element is zero. First, the algorithm finds the largest diagonal element. Then a permutation is performed to move the largest diagonal element on the active position in which column is working on calculating the Cholesky factors. This permutation process is called as a pivot. After the incomplete Cholesky decomposition, some of columns of the lower triangular matrix L are all zeros. Therefore, the matrix A is decomposed as $A \cong LL^T$ which is an approximation of the matrix A. Let $\tilde{A} = LL^T$. Then there is an approximation error $\Delta A = A - \tilde{A}$. If the approximation error is bounded by a certain value and the difference between the optimal values of the original and approximated problem is small, then the decomposition can be acceptable. Higham (1990) also showed the incomplete Cholesky decomposition is a stable method. Fine and Scheinberg (2001) derived the bound of the approximation error and showed the error bound is acceptable in the SVM problem.

The SVM problem MP considered in this dissertation in section 3.3.1 has the Hessian matrix which is a symmetric and positive semi-definite. The incomplete Cholesky decomposition method is applied to that system. The Hessian matrix in the SVM problem is too dense and large. Even if the incomplete Cholesky decomposition is applied to the problem, the calculation of $L^T\boldsymbol{x}$ and the space of memory to store the matrix L is still a burden. Fortunately, however, the rank of the Hessian matrix is significantly smaller than its size.

Figure 3.2 Incomplete Cholesky Decomposition



In Figure 3.2, the mark X denotes a non-zero element. This research considers the column-wise decomposition. During the decomposition, the largest diagonal element of the Cholesky factor is moved to the active column. For example, the algorithm calculates all diagonal elements of the L matrix which are the Cholesky factors and finds the largest one. The largest diagonal elements are moved to the first column using a permutation. Then the rest of the elements in the first column are calculated. That is the end of the first iteration or pivoting. The next iteration is for the second column of the L. At the $k+1$ iteration, if the largest diagonal element is zero, then the procedure of is stopped. Therefore the values of diagonal elements are $l_{11} \geq l_{22} \geq \cdots \geq l_{kk}$. In this case, the rank of the Hessian matrix is k. In Figure 3.2, the rank of the Hessian matrix H is two. The incomplete Cholesky decomposition method is sometimes used as a way to calculate the rank of a matrix. The rank k in the SVM problem is significantly smaller than the size of the Hessian matrix n. This property is advantageous for the calculation in the new algorithm because the algorithm needs to calculate the product between the L matrix and the $\boldsymbol{x}$ vector and the computer memory space to store the matrix L. The incomplete Cholesky decomposition in this research is based on the method in Fine and Scheinberg (2001). The algorithm uses the symmetric permutation which is known as symmetric pivoting introduced in Golub and Van Loan (1996). The detail algorithm is as follows.

Incomplete Cholesky Decomposition

```
for $i = 1:n$  : **Column-wise decomposition**
< **Step 1 :** Calculate all remaining diagonal Cholesky factors >
  for $j = i:n$
    $d_{jj} = h_{jj}$
    for $k = 0:i$
      $d_{jj} = d_{jj} - l_{jk}$
    end
  end

  if $\left(\sum_{j=i}^{n} d_{jj} > tol\right)$  : **Check the positiveness of the diagonal factor**
< **Step 2 :** Find the largest diagonal factor >
```

```
        find $j^*$ such that $d_{j^*j^*} = \max_{i \le j \le n} d_{jj}$

< **Step 3 : Swap the largest one and current active element >**
        for $j = i{:}n$  : **Column**
            $l_{ji} = h_{jj^*}$
        end
        for $j = 0{:}i + 1$  : **Row**
            $temp = l_{ij}$
            $l_{ij} = l_{j^*j}$
            $l_{j^*j} = temp$
        end
        Swap the permutation matrix $P$

< **Step 4 :** Calculate the rest of Cholesky factors in the active column >
        for $k = i + 1{:}n$
            for $j = 0{:}i$
                $l_{ki} = l_{ki} - l_{kj} * l_{ij}$
            end
        end
        $d_{ii} = \sqrt{d_{ii}}$
        for $k = i + 1{:}n$
            $l_{ki} = l_{ki}/d_{ii}$
        end
    else
< **Step 5 :** Finish the decomposition >
        $rank = i$
        break
    end
end
```

The elements of L are separately stored into diagonal elements d_{jj} and the others l_{ij} for the convenience of the calculation. h_{ij} denotes an element in the Hessian matrix H and P is a permutation matrix. The first step is to calculate all remaining diagonal Cholesky factors so that the algorithm can find the largest diagonal element among them. If the diagonal element is positive, then the algorithm finds the largest diagonal element and permutes the element and the active element for both the column and the row. Then, the method calculates the rest of the elements of the column. The algorithm goes to the next column and continues this process. If the diagonal element is not a positive or less than a certain tolerance, then the algorithm is stopped and the current number of column is the rank of the Hessian matrix.

If one takes an attention to the access to the original matrix H, the diagonal elements of H are only frequently accessed more than once through the decomposition process in step 1, but the others are accessed only once in step 3. This property is a great advantage for the SVM problem. The access to the Hessian matrix in the SVM problem means to calculate the kernel functions because the Hessian matrix is a kernel matrix as the (3.23) in section 3.2. Therefore one needs to calculate $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \varphi(\boldsymbol{x}_i) \cdot \varphi(\boldsymbol{x}_j)$ whenever the algorithm accesses the Hessian matrix during the decomposition procedure. The incomplete Cholesky decomposition is the most time consuming calculation in the proposed solving procedure and the less calculation of kernel functions is very helpful to the amount of calculations. Moreover, the Hessian matrix does not need to be stored explicitly. Instead, one only needs to store the lower triangular matrix L. The algorithm can save the memory space with the amount of $(n \times n) - (n \times k)$. There are several other issues for the implementation for the new algorithm as will be shown in the next section.

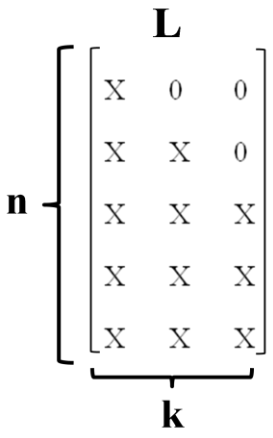
3.3.4 Implementation Issues

This section describes the implementation issues for the new solving procedure in the computer programming. The C programming language is used to implement the algorithm in a Linux system. The most time consuming calculations are related to the incomplete Cholesky decomposition and the calculation of the multiplication between the Hessian matrix H and the variable vector $\boldsymbol{x}$. Another issue is to store the lower triangular matrix L. The details are in the next sections.

3.3.4.1 Storing the lower triangular matrix L

Using the incomplete Cholesky decomposition, the memory space is significantly reduced. Let the Hessian matrix H be a $n \times n$ matrix and the matrix L from the incomplete Cholesky decomposition be $n \times k$. In the SVM, n is significantly larger than k. For example, if $n = 1{,}000$ and $k = 90$, then the amount of saving space is $(1000 \times 1000) - (1000 \times 90) = 910{,}000$. If n is larger, then one can save more space. Nevertheless, the matrix L needs a big memory space for a large scale problem. Sometimes, the rank of the Hessian matrix in the SVM problem is a little large up to about ten percent of the size of Hessian matrix. In this case, if $n = 10{,}000$, then $k = 1000$. then $10{,}000{,}000$ (ten million) memory space is required. If one uses an array to store this matrix in C programming language, then it is not efficient and ends up the program with a core dump.

Figure 3.3 L matrix from the incomplete Cholesky decomposition



Another way one may consider is to store only non-zero elements out of the matrix L. However, the zero elements are only ones at the upper diagonal side because the matrix is too dense. Since there are only a few zero elements in the matrix L as can be seen in Figure 3.3, this method is not very beneficial.

Therefore, this research proposes a simple and efficient method to store the matrix L. The idea is to store the elements of L into several single dimensional arrays.

Figure 3.4 The method of storing L matrix

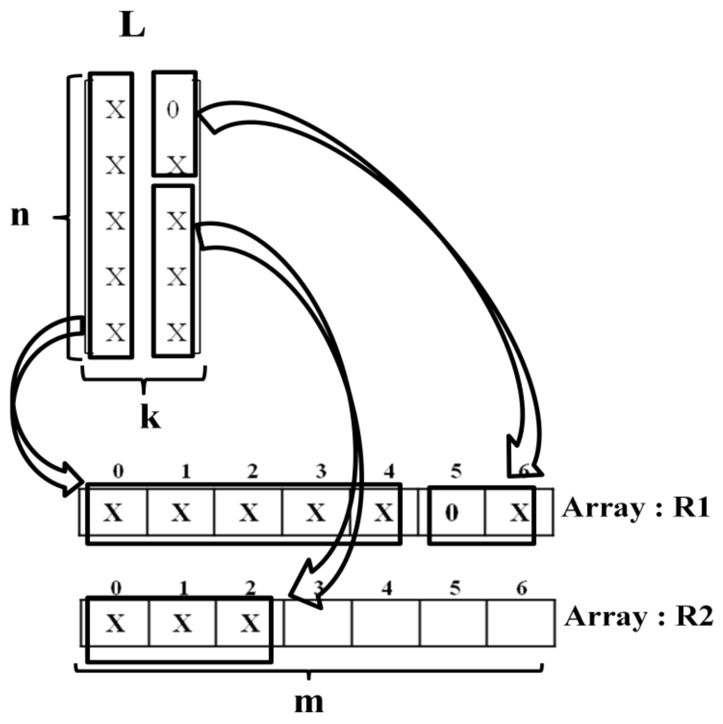


Figure 3.4 describes the way to store the matrix L. If $n \times k$ is too big to be stored in a single array, the method can put them into two arrays separately. Let the size of $R1$ or $R1$ matrix be m which is not so big. The details for storing are as follows.

Insert or Refer the Cholesky factors

Step 1 Calculate the refer number

$$r = (\text{column number} - 1) * n + (\text{row number} - 1)$$

Step 2 Find the position to insert or refer the element

```
If $r < m$,
    insert or refer the Cholesky factor to $r$th position in the array $R1$
else
    insert or refer the Cholesky factor to $(r - m)$th position in the array $R2$
```

For example, one assumes that the element l_{42} should be referred in Figure 3.4. The refer number is calculated as $r = (2 - 1) * 5 + (4 - 1) = 8$. Since $r > m$, the position is $(8 - 7)$th position in array $R2$. There is one calculation for the refer number and one comparison for the size of the value to find the array $R1$ or $R2$. The method can use more than three matrices such as $R1$, $R2$, $R3$, ... In this case, this technique can handle larger scale problems, but the efficiency would be down because the method needs more comparing operations to find the array. This research uses two arrays $R1$ and $R2$.

3.3.4.2 Calculation of Hx

In the main algorithm in section 3.3.1, the calculation of the coefficient of linear term of the objective function in step 2 is a little complicated. The algorithm needs to solve this subproblem at every iteration. The calculation $\boldsymbol{g}^k = LL^T\boldsymbol{x}^k + \boldsymbol{c}$ should be frequently conducted during the solving procedure. The Hessian matrix is decomposed by $H \cong LL^T$ in step 1. The calculation $LL^T\boldsymbol{x}^k$ is the most time consuming task for calculating the coefficient.

Let L be a $n \times k$ matrix. L^T is a $k \times n$ matrix and $\boldsymbol{x}^k$ is a column vector with the size n. It is better to calculate the multiplication between a matrix and a vector than between two matrices. First, one calculates $L^T\boldsymbol{x}^k$. The dimension of calculation is $(k \times n) \times (n \times 1) = (k \times 1)$. Let Ltx be a vector with the size k. The elements are $ltx_1 = \sum_{i=1}^{n} l_{i1}x_i$, $ltx_2 = \sum_{i=2}^{n} l_{i2}x_i$, ..., $ltx_k = \sum_{i=k}^{n} l_{ik}x_i$. The next calculation is $LLtx$ and the dimension is $(n \times k) \times (k \times 1) = (n \times 1)$. There are two cases in this calculation. If $i < k$, the final value $hx_i = \sum_{j=1}^{i} l_{ij}ltx_j$, where hx_i is an element of $H\boldsymbol{x}$. If $i \geq k$, the value $hx_i = \sum_{j=1}^{k} l_{ij}ltx_j$.

This type of calculation $LL^T\boldsymbol{x}^k$ is frequently used in this solving procedure. This research uses that calculation to find ρ and b values in (3.60) and (3.61). The BB rule for updating parameter also uses the calculation of $\boldsymbol{d}^T H\boldsymbol{d}$ where $\boldsymbol{d}$ is the direction vector. The objective function value also needs to use this calculation.

During the line search procedure, $\boldsymbol{x}^k$ is updated by $\boldsymbol{x}^{k+1} = \boldsymbol{x}^k + \lambda\boldsymbol{d}^k$ to find the best λ. Every time the variable is updated, the algorithm checks the objective function value. Then the calculation $LL^T\boldsymbol{x}^k$ needs to be repeated for new λ values. To avoid this burden, one can use an efficient way. At each iteration, the method already has the calculation $LL^T\boldsymbol{x}^k$ for the coefficient of linear term and $LL^T\boldsymbol{d}^k$ for the BB rule. If the algorithm needs to calculate $LL^T\boldsymbol{x}^{k+1}$, the variables can be separated as follows.

$$LL^T\boldsymbol{x}^{k+1} = LL^T(\boldsymbol{x}^k + \lambda\boldsymbol{d}^k) = LL^T\boldsymbol{x}^k + \lambda LL^T\boldsymbol{d}^k \tag{3.97}$$

Using (3.97), the algorithm does not need to calculate $LL^T\boldsymbol{x}^{k+1}$ for all different λ values.

3.3.4.3 Calculation of new λ and initial solution

In SPGM and AA, the methods used a one dimensional quadratic interpolation method to find a new λ. One dimensional optimization method has two different types: search and approximation in Antoniou and Lu (2007). The quadratic interpolation method is an approximation method in one dimensional optimization. While a linear interpolation needs two points because two coefficients need to be calculated, a quadratic interpolation usually needs three points to find three coefficients. Let us consider a one dimensional optimization problem such as $\min f(x)$. One can approximate $f(x)$ a polynomial function $p(x) = a_1x^2 + a_2x + a_3$.

Let $p(x) = a_1x^2 + a_2x + a_3 = f(x) = f_i$. If one knows three different points $\{x_1, x_2, x_3\}$ and their function values $\{f_1, f_2, f_3\}$, then all coefficients $\{a_1, a_2, a_3\}$ can be obtained by solving three simultaneous equations. By the approximation, the minimum value of $p(x)$ is close to the original minimum value of $f(x)$.

In the SVM problem, Only two points are known such as x^k and $x^k + d$ in the line search method and try to find a point $x^k + \lambda d$, where $0 < \lambda < 1$, between the two points. There is a quadratic interpolation method with two points if there is information about two points and a first derivative of the function. Let f_i' denote the first derivative, two points $\{x_1, x_2\}$ known, and $\bar{x}$ be a minimum value. Then, the algorithm has three equations: $p(x_1) = f_1$, $p(x_2) = f_2$, and $p'(x_1) = f_1'$.

Solving these equations, the minimum value is as follows. The details are in Antoniou and Lu (2007).

$$\bar{x} = x_1 + \frac{f_1'(x_2 - x_1)^2}{2[f_1 - f_2 + f_1'(x_2 - x_1)]} \tag{3.98}$$

In the new solving algorithm, $f_1 = f_c$, $f_2 = f_v$, and $f_1' = \langle \boldsymbol{d}^k, g(\boldsymbol{x}^k) \rangle$. Therefore, one can obtain the new value of λ as follows.

$$\lambda = \frac{\langle \boldsymbol{d}^k, g(\boldsymbol{x}^k) \rangle}{2(f_c - f_v + \langle \boldsymbol{d}^k, g(\boldsymbol{x}^k) \rangle)} \tag{3.99}$$

Another issue is to set up an initial feasible solution. Two constraints are used for finding an initial solution. The two constraints are (3.71) and (3.73) : $\boldsymbol{y}^T\boldsymbol{\alpha} = 0$ and $\boldsymbol{e}^T\boldsymbol{\alpha} = \nu n$. The value y_i has only one of two values 1 and -1. Let m be a number of 1's and k be a number of -1's. From the constraint $\boldsymbol{y}^T\boldsymbol{\alpha} = 0$, let $\alpha_i = \alpha_m$ for all $i = \{i | y_i = 1\}$ and $\alpha_j = \alpha_k$ for all $j = \{j | y_j = 1\}$. Then two constraints can be rewritten as follows.

$$m\alpha_m - k\alpha_k = 0 \tag{3.100}$$

$$m\alpha_m + k\alpha_k = \nu n \tag{3.101}$$

Solving these two equations, one can obtain α_m and α_k as follows.

$$\alpha_m = \frac{\nu n}{2m} \text{ and } \alpha_k = \frac{\nu n}{2k} \tag{3.102}$$

Once one gets the number of 1 and -1 values for $\boldsymbol{y}$, if $y_i = 1$, then the initial $\alpha_i = \frac{\nu n}{2m}$, and otherwise $\alpha_i = \frac{\nu n}{2k}$.

3.4 Experimental Results

This research has conducted some experiments with the way of combination of matrix splitting with gradient projection method, Incomplete Cholesky, and some options for the line search and the updating alpha methods. The data are from LIBSVM data collections. The algorithm is implemented in C programming language, complied using gcc and ran on a Fedora 13, 64 bit Red Hat Linux machine with 4 GB memory and Intel i7 QuadCore Bloomfield CPU running 2675 MHz. This research has compared four different methods for the line search and updating parameter method in the newly proposed main algorithm.

Table 3.1 SPGM method

Problem	Data	Feature	Kernel	Rank	SV	BSV	Training error	Time (second)
Splice	1,000	60	Polynomial	979	1,000	0	7.10%	3.72
	1,000	60	Gaussian	979	1,000	0	0%	3.88
Svmguide1	3,089	4	Polynomial	1,846	3,089	0	35.25%	58.29
Adult 4	4,781	123	Polynomial	1,193	4,781	0	24.26%	38.19
Mushrooms	8,124	112	Polynomial	701	8,124	0	10.91%	80.5

Table 3.1 shows the results for SPGM method described in section 3.3.2.1. The problems are medium sized. The algorithm uses polynomial or Gaussian kernel function. The rank denotes the rank of the Hessian matrix derived from the incomplete Cholesky decomposition. In this result, though the testing errors and the solution time are not bad, the number of support vectors is too large, which is not efficient because one should use all data point to test other problems.

Table 3.2 GVPM method

Problem	Data	Feature	Kernel	Rank	SV	BSV	Training error	Time (second)
Splice	1,000	60	Polynomial	979	1,000	0	7.10%	4.12
	1,000	60	Gaussian	979	1,000	0	0%	4.05
Svmguide1	3,089	4	Polynomial	28	2,887	663	6.53%	0.3
Adult 4	4,781	123	Polynomial	1,193	2,378	717	24.84%	38.75
Mushrooms	8,124	112	Polynomial	701	8,124	0	10.85%	85.2

Table 3.2 presents the results of the GVPM method. The training errors and solution times are similar to the SPGM method. The number of support vectors is much smaller than that of SPGM method in two problems such as svmguide1 and adult4.

Table 3.3 MSM method

Problem	Data	Feature	Kernel	Rank	SV	BSV	Training error	Time (second)
Splice	1,000	60	Polynomial	979	1,000	0	6.60%	4.08
	1,000	60	Gaussian	979	1,000	0	0%	4.13
Svmguide1	3,089	4	Polynomial	28	495	417	4.14%	2.56
Adult 4	4,781	123	Polynomial	1,193	2,533	267	24.84%	23.33
Mushrooms	8,124	112	Polynomial	701	5,969	190	1.32%	80.03

The MSM method has better results than the previous two methods. Though the solution time is similar to GVPM method, the training errors are much better than other methods. The number of support vectors is also smaller than others.

Table 3.4 AA method

Problem	Data	Feature	Kernel	Rank	SV	BSV	Training error	Time (second)
Splice	1,000	60	Polynomial	979	1,000	0	6.40%	4.17
	1,000	60	Gaussian	979	1,000	0	0%	3.97
Svmguide1	3,089	4	Polynomial	28	495	429	4.14%	3.14
Adult 4	4,781	123	Polynomial	1,193	2,133	1342	22.63%	41.03
Mushrooms	8,124	112	Polynomial	701	4,922	568	1.13%	84.83

Table 3.4 describes the results of the AA method. The results are very similar to the MSM method. The number of support vectors is also small and the training error has good results.

From these results, the MSM and AA method have better results than other two methods in terms of the training error and the number of support vectors.

Table 3.5 Large-scale problems (MSM method)

Problem	Data	Feature	Kernel	Rank	SV	BSV	Training error	Time (second)
Adult 6	11,220	123	Polynomial	510	5,713	549	23.99%	60.69
Adult 7	16,100	123	Polynomial	487	4,686	4,047	24.33%	75.63

Table 3.5 shows the results of two large scale problems. One usually says the problem of which size is larger than 10,000 is a large scale problem in SVM. This result has good training errors and especially the number of support vector is significantly smaller than the size of the problem.

Table 3.6 Large-scale problems (AA method)

Problem	Data	Feature	Kernel	Rank	SV	BSV	Training error	Time (second)
Adult 6	11,220	123	Polynomial	510	5,713	549	23.99%	60.69
Adult 7	16,100	123	Polynomial	472	3,140	2,466	24.33%	73.08
Adult 8	22,696	123	Polynomial	467	4,377	3,831	24.25%	81.01
ijcnn1	49,990	22	Polynomial	202	49,990	0	9.70%	61.87

Table 3.6 represents the results for large scale problems using AA method.

In the ν-SVM problem, the range of ν is $0 < \nu < 1$. The next experiments show the changes of the training error and the number of support vectors as the parameter ν changes.

Table 3.7 Sensitivity Analysis of ν for svmguide1 (GVPM method)

Problem	Data	Feature	nu	SV	BSV	training error	time (sec.)
Svmguide1	3089	4	**0.3**	2,101	144	7.02%	0.35
			0.4	2,887	663	6.53%	0.3
			0.5	2,121	328	8.02%	0.35
			0.6	1,676	1148	7.64%	0.33
			0.7	2,178	1420	10.65%	0.39

As the parameter ν increases, the training error increases and the number of support vectors is a little changed. The number of bound support vectors increases.

Table 3.8 Sensitivity Analysis of ν for mushrooms (GVPM method)

Problem	Data	Feature	nu (ν)	SV	BSV	training error	time (sec.)
Mushrooms	8124	112	**0.2**	4,922	568	1.13%	84.83
			0.3	8,115	219	1.64%	91.08
			0.4	6,741	0	2.90%	89.21
			0.5	7,735	152	6.40%	87.72
			0.6	6,570	947	9.90%	84.81
			0.7	6,571	4,266	10.36%	85.44
			0.8	6,980	5,658	10.42%	89.1

Table 3.8 shows the results of the sensitivity analysis of ν for mushrooms. The result is similar to the result of Table 3.7. As the parameter ν increase, the training error increases and the number of support vectors and bound support vectors increase.

Table 3.9 Comparisons of Bitran-Hax and Dual Bound algorithm (AA method)

Problem	Data	Feature	Kernel	Method	Time (Second)
Adult 1	1,605	123	Polynomial	Bitran-Hax	19.73
				Dual Bound	20.44
Adult 2	2,265	123	Gaussian	Bitran-Hax	77.68
				Dual Bound	78.1
Adult 3	3,185	123	Gaussian	Bitran-Hax	48.84
				Dual Bound	48.99
Adult 4	4,781	123	Polynomial	Bitran-Hax	31.99
				Dual Bound	31.54
Adult 5	6,414	123	Polynomial	Bitran-Hax	45.21
				Dual Bound	43.05
Adult 6	11,220	123	Polynomial	Bitran-Hax	60.62
				Dual Bound	59.8

The SVM problem has a subproblem as the quadratic knapsack problem. Table 3.9 compares the Dual Bound and the Bitran-Hax algorithm. Like the results in chapter 3, the Bitran-Hax algorithm works a little better than the Dual Bound when the size of the problem is small. As the size of the problem increases, the Dual Bound algorithm gets better solution time.

The experimental results showed that the four different methods have different results for the line search and updating the parameter method. The SPGM and GVPM have good results for the training error and the solution time, but the large number of support vectors is a problem. On the other hand, the MSM and AA method have better results for the training and the solution time than the other two methods. The number of support vectors is also much smaller than the other two ones. This research also has good results for large scale problem with the newly proposed decomposition approach. The results in the last two tables showed the properties of ν-SVM problem well.

3.5 Conclusions

In this chapter, this research proposed a new solving approach for the ν-SVM classification problem. The solution of the ν-SVM problem is obtained by solving a quadratic programming problem with linear constraints and box constraints. The quadratic programming problem is a singly linearly constrained quadratic convex problem with box constraints. The SVM problem has a huge and dense Hessian matrix that is hard to solve with a traditional method for the quadratic program.

The newly proposed method uses a matrix splitting and gradient projection method with the incomplete Cholesky decomposition method. The matrix splitting method decomposes the Hessian matrix into a sum of two simple matrices and makes a subproblem with the simple Hessian matrix that is a nonlinear knapsack problem. The algorithm solves the subproblem iteratively using the Bitran-Hax or Dual Bound algorithm until the optimal solution is obtained. During the procedure, the new algorithm uses a similar gradient projection method including the line search and using a parameter α. However, this method can be only used to solve medium sized problems. For the large scale problems, new algorithm uses the incomplete Cholesky decomposition method. The incomplete Cholesky decomposition method is well known as a simple, stable, and accurate. Moreover, the algorithm took advantage of solving a singly linearly constrained quadratic convex problem by using some other methods for the line search and updating parameter methods.

The algorithm proposed in this research has a solving procedure as a combination of the matrix splitting method, gradient projection method, and the incomplete Cholesky decomposition. With this frame of methodology, the methods of the line search and updating parameter α such as BB type rule are open to be used from the research of a singly linearly constrained quadratic convex problems. In addition to that, the subproblem resulted from the matrix splitting is a quadratic knapsack problem. The Dual Bound algorithm or Bitran-Hax algorithm introduced in chapter 2 can be used to solve the problem.

Some experimental results have shown that new algorithm solved medium sized problems and large scale problems as well. Although the results have not shown a significant improving or accuracy comparing with other well known software, the algorithm has performed well in the medium and large scale problems. That can be seen as a great potential and promising solving algorithm for the ν-SVM problem. Furthermore, many methods for the line search and updating parameter can be plugged in new algorithm to be extended or improved. This methodology can be an alternative and another direction of solving the SVM problem.

CHAPTER 4 - Supplier Selection

In this chapter, this research applies the SVM classification to the supplier selection problem. Supplier selection is one of important issues in supply chain companies that need to purchase raw materials or components from upstream suppliers in order to produce finished goods. The SVM classification is used to select qualified suppliers out from all potential suppliers. Among the qualified suppliers obtained by the SVM, the final suppliers are selected by solving a mathematical programming problem. The procedure of these two steps is called an integrated approach for supplier selection. This research proposes an efficient integrated method for the supplier selection problem using SVM classification and the mixed integer programming model.

The motivation for the research in this chapter is to handle high dimensional supplier selection problems and to develop an efficient and fast method for solving those problems. Industries continue to expend, to become more interactive and complicated. Supplier selection is also becoming a high dimensional problem. Qualitative methods successfully handle large amount of intangible information and find the best solution for each case. However, qualitative methods are limited by dimension and generalization. For this reason, a number of quantitative methods have been proposed for solving the supplier selection. The quantitative methods can solve problems of higher dimension, but the methods require the numerical information from qualitative methods. Therefore, the integrated approach of the qualitative and quantitative methods has become the focus of research in recent years. This chapter proposes an efficient approach, integrating the SVM and the mathematical programming to solve the supplier selection problem that may be a large scale problem.

4.1 Supplier selection

As product complexity has increased, manufacturing companies depend more on outsourcing or purchasing parts of a product or materials for the production from other companies. These upstream companies are suppliers. Procurement in a manufacturing company has become an important issue because it directly affects the quality of product, on-time delivery, inventory control, and production planning. Moreover, the procurement plays an important role in determining the cost structure of a product which largely relies on which suppliers the company has worked with.

The supplier selection is a problem that selects the best suppliers that meet the requirement of the buyer company. Aissaoui et al. (2007) described types of supplier selection problems, focusing especially on number of suppliers and criteria. This research considers the multiple suppliers and multiple criteria. The multiple suppliers, also known as multiple sourcing, can avert product shortages and keep competition among eligible suppliers. The multiple criteria can help in making better decisions. A company cannot change suppliers every day, so once a company selects suppliers to work with, formal contracts usually are signed to ensure the terms and conditions of the procurement requirements and to govern the supply chain coalition, therefore, the company would keep purchasing from the suppliers for a while because frequently changing suppliers is costly and time consuming task. This is a reason that supplier selection is important and should be prudent in many companies. Moreover, the criteria for selecting suppliers have conflicts with each other. For example, a supplier with good quality or service may not be the supplier has lowest prices. Therefore, the supplier selection requires compromise among the criteria. The criteria have both tangible and intangible factors such as prices and services. The decision on suppliers requires accommodating subjective standards even if a quantitative, systematic method is used for selecting suppliers. However, the qualitative factors cannot be directly factored into mechanical solution procedure, but they are instead transformed into numerical scores. The four steps of selecting suppliers are in the following subsections as described in De Boer et al. (2001) and Aissaoui et al. (2007).

4.1.1 Problem definition

The problem definition requires a company to establish a goal for purchasing materials or components from suppliers. The company may expect an improvement in the quality of the product, a better service for customers, a lower pricing strategy, and so on. Depending on how the decision maker defines the problem, the rest of the selection steps change. This step depends on the decision maker's opinion.

4.1.2 Deciding on Criteria

The company should decide which criteria are used for selecting suppliers. The criteria denote the measurement of the value of suppliers. There are many factors in establishing the supplier selection criteria describing the values of a supplier such as price, service, delivery accuracy, quality, and so on. Once a company defines the goal of procurement, the criteria are determined to achieve the goal. These criteria could be different from one company to another because their procurement goals are different from each other. Decisions on criteria are also subjective as the same as the previous step. A company may choose only one criterion. However, the multiple criteria are usually preferred because the company's decision would be driven by more than one objective.

The factors can be divided into two types such as quantitative and qualitative. For the convenience of measuring, the qualitative factors are expressed in numbers. Dickson (1966) surveyed many companies for supplier selection issues and introduced 23 factors for the criteria. Weber et al. (1991) presented 74 factors with the same way of Dickson's study. The majority factors for the supplier selection problem are covered by these factors until today. The factors often correlate with each other. For example, if the supplier's price is low, then the quality may be low as well. Since there are some interactions between factors, a compromised solution becomes necessary. The large number of factors is not always better than the small number of factors because using too many factors makes the goal of procurement meaningless and is difficult to collect data. The decision of the criteria is totally up to the environment of the company. For additional information, see Aissaoui et al. (2007).

4.1.3 Pre-selection

Once the criteria are determined, the company should collect data on suppliers for all criteria to help in evaluating supplies. The company can then select suppliers who meet these criteria. In this step, the company selects qualified suppliers from all potential suppliers. In fact, before the pre-selection step, the company should collect the data of suppliers for all factors because the data are necessary for evaluating suppliers. This research assumes the collection of the data has finished at the end of the decision of the criteria. The qualified supplier denotes the potential supplier that satisfies a certain level of the criteria and has possibility to be the final supplier to contract with the buyer. If the company selects the final suppliers with the criteria without the pre-selection, that seems to be simple. However, using two steps for the selection can get better selection than single step. It is the same reason the recruiting procedure usually uses the screening first and then does the interview. This strategy reduces the failure rate of the selection. The pre-selection step also called as the pre-qualification divides the suppliers into qualified suppliers and the others. The other suppliers would not be considered for the final suppliers. The selection of qualified suppliers is a binary classification problem that classifies suppliers into two groups. Any one of several quantitative methods can be used in this step. The literature reviews for the methods can be found in De Boer (2001), Aissaoui et al. (2007), and Ho et al. (2010).

Qualitative methods have been used for pre-selection. Wright (1975) suggested a lexicographic rule that evaluates all suppliers with the most important criterion first and then checks the other criteria sequentially. Crow et al. (1980) proposed a conjunctive rule that uses the minimum threshold for each criterion and selects only suppliers that satisfy all minimum thresholds. The categorical method was introduced by Timmerman (1986). Using the criteria, the method categorizes suppliers into three groups: good, neutral, and bad.

Data Envelopment Analysis (DEA) is a method to use the ratio of multiple inputs and outputs. The DEA can classify suppliers into efficient and inefficient groups in Weber and Ellram (1993), Weber and Desai (1996), Weber et al. (1998), Papagapiou et al. (1997), and Liu et al. (2000). The DEA is a method for analysis of the system that has multiple inputs and outputs. Thus, the supplier selection, with its multiple criteria, can be solved with DEA. However, the main drawback of DEA is the difficulty of model specification, because results are very sensitive to the selection of inputs and outputs.

Data mining is anther approach that could be used for the pre-selection problem. Cluster analysis (CA) minimizes the differences between values within a group while it maximizes the differences between values from different groups. Hinkle et al. (1969) and Holt (1998) applied the CA to classify the suppliers. However, this method has no mechanism for differentiating between relevant and irrelevant variables and is an unsupervised learning method which has lower accuracy than the supervised method.

Case based reasoning (CBR) is also used for this problem. The CBR uses the previous decision history or similar results. Ng et al. (1995) developed a CBR system for the pre-selection problem.

4.1.4 Final selection

The last step of the supplier selection is to determine the final suppliers to make contracts and assign the order quantities to them. The final suppliers are selected out of the list of qualified suppliers which were selected in the pre-selection step. There are two problems in this step. First, the company should select the final suppliers from the qualified suppliers. This problem is the same as the pre-selection problem. Therefore, the methods in the previous section can be also used for this problem. Next, the final suppliers are allocated the orders. This is an allocation problem with certain capacity constraints. The combination of two methods can be used for this step.

This research considers a multiple sourcing problem that has one more suppliers. In addition, this research considers the number of items or materials can be supplied by a supplier. If the material is a single unit, each supplier has only one material, which is a simple case. On the other hand, if the material is a multiple type, then each supplier can produce or provide two or more materials. In this case, the procurement decision can consolidate the purchase to obtain the quantity discount from suppliers. The supplier and the buyer company may also reduce the ordering and logistic cost. For discount, the ordering may change, which affects the inventory management factors for the buyer company. This complicates the problem, making it harder to solve as described in Aissaoui et al. (2007). In addition to considering the multiple material types, the time period can be considered in the supplier selection. Most studies and methods for the supplier selection are dealing with the single period. On the other hand, some studies have considered the supplier selection problem with multiple time periods such as Aissaoui et al. (2007). Such multiple period models must consider the inventory management and the lot sizing rule. Furthermore, other studies like Rosenblatt et al. (1998), Ghodsypour and O'Brien (2001), Liao and Kuhn (2004) have identified the economic order quantity (EOQ) concept as useful in selecting suppliers under these conditions. The lot sizing rule is also considered in Buffa and Jackson (1983), Bender et al. (1985), Tempelmeier (2002), Hong et al. (2005).

The mathematical programming approach has been widely used for solving this step such as linear programming, integer programming, nonlinear programming, stochastic programming, goal programming, multi-objective programming, and so on are discussed in De Boer (2001), Aissaoui et al. (2007), and Ho et al. (2010). The objective function of the mathematical model is formulated to minimize costs or maximize profits. The constraints are the capacity of the supplier and tolerances of the factors the company considers. The most advantage of the mathematical programming method is the ability to assign the order quantities to the final suppliers if decision variables are defined as the amount of order quantity for each final supplier. This ability means that the mathematical programming method can be used in combination with other methods. For example, a classification method selects the final suppliers and then the mathematical programming method assigns the order quantity. This approach is called as an integrated approach.

The total cost of ownership (TCO) model uses the total cost, including all relevant costs of purchasing, to select suppliers. Ellram (1995) introduced this method, but TCO has been combined with rating system in Monczka and Trecha (1988) and Smytka and Clemens (1993) and with mathematical programming in Degraeve et al. (2004). Although finding all relevant costs can be difficult, the method uses more detailed information to select suppliers.

The integrated approach is useful in this step because the problem has two issues such as selecting and assigning. Mathematical programming is a common assigning method. Among many selecting methods are the methods mentioned in the pre-selection step and the analytical hierarchy process (AHP), a popular analytical tool that provides the scores for all suppliers derived by pair-wise comparisons between suppliers. AHP can be used for the multiple criteria problems and directly handle both quantitative and qualitative factors. One of integrated approaches has the AHP selects the final suppliers and the mathematical programming assign order quantities to them. The fuzzy set theory and DEA are also used for the final selection in combination with mathematical programming.

4.2 Literature Review

Supplier selection is an important issue in companies because it directly affects the profits and combines with other functions such as production, sales, finance, and so on. In this respect, extensive studies have been proposed to solve this problem. Depending on what the company prefers, an individual or integrated approach may be used to select suppliers. For example, a company might use only the AHP method, but another company may use an integrated approach with both the AHP and the mathematical programming method. This research focuses on the integrated method because the supplier selection occurs in two major steps such as selecting and assigning, and these two steps each need a solution. The idea in this research is to use the support vector machine (SVM) for the pre-selection step. This section reviews the literature of the integrated approaches for the supplier selection and using SVM in the supply chain management.

4.2.1 Integrated approaches

The integrated approach to select suppliers uses more than two methods. Most integrated approaches use two methods for the two different steps. In this case, the first step selects qualified suppliers and the second step selects the final suppliers and allocates orders to the final suppliers. The mathematical programming method is usually used for the second step while one of many classification methods is used for the first step.

Ghodsypour and O'brien (1998) proposed an integrated method of the supplier selection using the AHP and the linear programming method. AHP uses both tangible and intangible factors and find the final scores of suppliers. The final scores is the coefficient of the objective function of linear programming in the next step. The linear programming model maximizes the total value of purchasing (TVP) and the solutions provide the order quantities for the final suppliers.

Weber et al. (1998) suggested a little different approach, using the mathematical programming first. The multi-objective programming (MOP) method selects suppliers first. With the optimal solution from MOP, the method then finds a selection path which improves the MOP criteria performance so that some non-selected suppliers can be included in a specific MOP solution. One of three methods they proposed to find the selection path is the data envelopment analysis (DEA).

Cebi and Bayraktar (2003) used a combination of AHP and the lexicographic goal programming (LGP) to solve the supplier selection problem. AHP finds the scores for all objective functions and the LGP assigns orders to the suppliers. Wang et al. (2004) proposed an integrated method using AHP and preemptive goal programming (PGP). The AHP method selects the suppliers and PGP assigns the order quantities to the suppliers.

Hong et al. (2005) introduced a method using the meaning period unit (MPU) and mixed integer programming. They considered multiple period problems. The method divides the total period into several MPUs and identifies the procurement condition by MPU in the pre-qualification step. Mixed integer programming is used in the final selection step. Sarfaraz and

Balu (2006) suggested a method combining the quality function deployment (QFD), AHP, and PGP. QFD helps make criteria measurable. AHP then selects suppliers meeting the criteria, and PGP then assigns orders. Tseng et al. (2006) used the rough set theory (RST) to manipulate the data, make weight and decision rules for factors, and identify significant features. The method uses the RST iteratively until the data are appropriate for the selection. Then, the support vector machine (SVM) selects suppliers. This method integrates an individual approach (RST) and a population based approach (SVM). Ting and Cho (2008) proposed an integrated approach using AHP and multi-objective linear programming (MOLP). AHP selects qualified suppliers and the MOLP assigns orders to the suppliers.

Demirtas and Ü stün (2008) used the analytic network process (ANP) for selecting qualified suppliers and the multi-objective mixed integer linear programming (MOMILP) for the final selection. Kumar et al. (2004) and Azadeh et al. (2010) suggested an integrated approach of the AHP and fuzzy linear programming. Kokangul and Susuz (2009) proposed a method of combination of the AHP and nonlinear integer multi-objective programming. Che and Wang (2008) used the genetic algorithm (GA) and mathematical model for supplier selection and production planning. Kuo et al. (2010) introduced a method that integrated particle swarm optimization (PSO), fuzzy neural network (FNN), and artificial neural network (ANN). FNN collects the qualitative data, and PSO provides the initial weights for the FNN model. The ANN selects suppliers using the qualitative data from FNN and quantitative data from the ERP or database system. Their method takes the advantages of machine learning like the artificial neural network. As Kuo et al. (2010) mentioned, the ANN type method has strengths that do not need complex formulation, but can handle large scale data and uncertainty. However, the neural network has drawbacks: slow convergence, lack of generalization, and lack of theoretical basis.

On the other hand, the support vector machine (SVM), another machine learning method, overcomes these problems. This research uses the support vector machine for selecting potential suppliers using past data.

4.2.2 Support vector machine in supply chain management

Supply chain management (SCM) becomes an important issue in both industries and academia. The SCM aims to achieve the optimal condition for all related sectors such as the buyer, supplier, transportation, customer, and so on. Support vector machine (SVM) is a new supervised learning method for regression and classification which overcomes drawbacks of neural network method theoretically and practically. There does not seem to be any relationship between the SCM and the SVM.

However, in recent years, some researchers have tried to connect SCM with SVM. SCM can be one of applications of SVM. On the one hand, SVM can be one of methodologies of SCM. SVM is used in SCM in such a way of supplier selection, demand forecasting, and the others. For the literature of supplier selection, Sun et al. (2005) proposes a supplier selection model based on support vector machine for classification problem and presents the supplier selection criteria and quantitative methods using fuzzy and pairwise comparison. Wen and Li (2006) establish a set of index system which uses multi-layer SVM classifier to assess the credit grade of suppliers. Hsu et al. (2007) applies the SVM to build the supplier evaluation classifier and uses the Likert and Fuzzy for scaling data. Guosheng and Guohong (2008) use the SVM for regression problem to predict the credit index of suppliers. Cai et al. (2008) divides the supplier selection stage into two stages; primary election and well-chosen. SVM for classification problem is applied in supplier primary election stage. Next, for the demand forecasting, Carbonneau et al. (2008) uses the SVM for regression problem to forecast distorted demand signal with high noise in the context of supply chain. Shouquan and Zhiwen (2007) forecast the demand of multi-echelon of the supply chain based on SVM for regression problem which aims to alleviate the bullwhip effect and to improve the supply chain performance. Yue et al. (2007) employs the technique of SVM for regression problem to forecast the demand of beers for retailers. Carbonneau et al. (2008) compares several machine learning techniques including SVM for regression problem to forecast the distorted demand at the end of a supply chain. At last, for the other trials, Li et al. (2005) considers SVM as a reasoning method to find an effective solution of collaborative identification of coordination questions in supply chain. Wan et al. (2005) applies the simulation optimization with surrogate model to SCM. The Least Square Support Vector Machine (LSSVM) captures the casual relations embedded in simulation results. Xiaohui et al. (2007) uses the recognition and regression forecasting function of the support vector machine to put forward the order forecasting model.

Table 4.1 Supply Chain Management with SVM

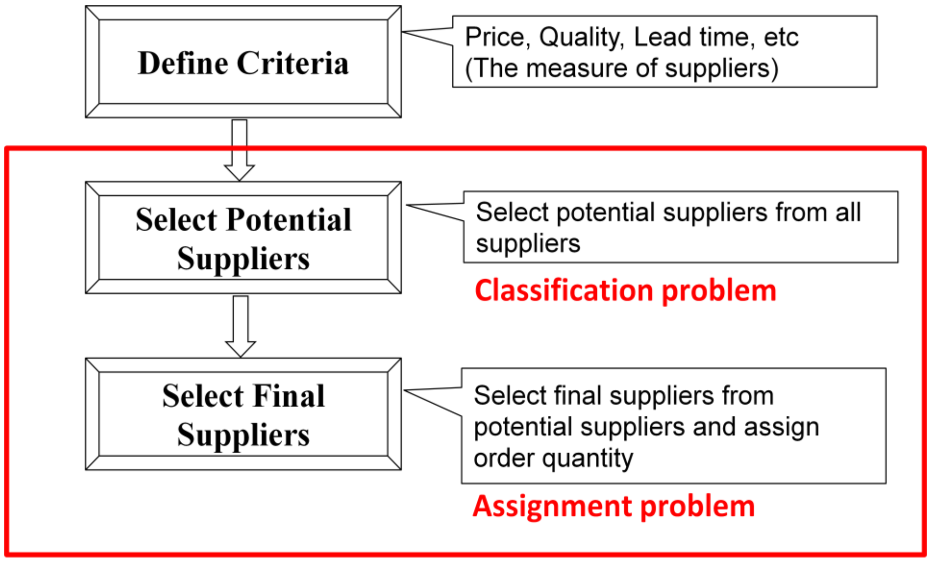
Application	Type	Authors	Year	Software	Data	Input	Output	Comparisons
Supplier Selection	Classification	Sun et al.	2005	N/A	Simple ex.	7	3	FS
	Classification	Wen & Li	2006	N/A	Simple ex.	13	4	BPNN
	Classification	Hsu et al.	2007	LIBSVM	Questionnaire	5	3	SLF
	Regression	Guosheng & Guohong	2008	N/A	Simple ex.	5	index	BPNN
	Classification	Cai et al.	2008	WINSVM	Simple ex.	7	2	3 Kernels
	Classification	Guo et al.	2009	N/A	Real data	30	7	SVM
Demand Forecast	Regression	Carbonneau et al.	2007	mySVM	ERP(real)	D	D	ANN/RNN
	Regression	Shouquan & Zhiwen	2007	N/A	Logistics(real)	6	D	ANN
	Regression	Yue et al.	2007	LIBSVM	Retailer's(real)	7	D	Statistical method, Winder model, RBFNN
	Regression	Carbonneau et al.	2008	LS-SVM	Real data	D	D	NN, RNN
	Regression	Wu	2010	N/A	Real data	6	D	*v*-SVM
Lead time Forecast	Regression	de Cos Juez et al.	2010	N/A	Aerospace	12	Lead time	Cox model
Collaborative Identification	Classification	Li et al.	2005	LIBSVM	Simple ex.	7	2	N/A
Simulation Optimization	Regression	Wan et al.	2005	N/A	Simple ex.	N/A	N/A	N/A
Order Prediction	Regression	Xiaohui et al.	2007	LIBSVM	Simple ex.	6	7	BPNN

Table 4.1 shows the literature of using the SVM in the supply chain management. The two major applications are the supplier selection and the demand forecast. The SVM classification is applied to the supplier selection and the regression is for the demand forecast. The software column describes the well-known software used in that research and 'N/A' means that the authors implemented the algorithm. In the input and output columns, the D denotes the demand. The number in the input column denotes the index or factor. For example, 7 represents the 7 factors. The number in the output column represents the class or grade. The 'Simple ex' in the data column denotes a simple example. The last column for the comparisons is the method compared with the SVM in the literature. The methods in the last column are abbreviated as follows. FS is for the fuzzy synthetical evaluation. BPNN is for the back propagation neural network. LSF is for the scaling by Likert and fuzzy. ANN is for the artificial neural network. RNN is for the recurrent neural network. RBFNN is for the radius basis function neural network.

4.3 Methodology

As it can be seen in the section 4.1, a variety of methods can be used for each step of the supplier selection solving procedure. Although the first two steps such as the problem definition and the criteria decision are important, they are heavily dependent on the experience and knowledge of experts. Therefore, the qualitative methods are used to solve those problems. On the other hand, the last two steps such as the pre-selection and the final selection are complicated to solve with the qualitative methods. If the number of suppliers and factors are large, the problem gets more complicated. For the high complexity problems, the quantitative methods can be more attractive than the qualitative methods. This research focuses on the pre-selection step using the SVM and the final supplier selection using the mathematical programming. This research assumes the problem with multiple sourcing and a single period. If one assumes the first two steps are already determined, the selection procedure of simple version is as follows (Aissaoui et al., 2007).

Figure 4.1 Supplier Selection Procedure


Define Criteria
Price, Quality, Lead time, etc (The measure of suppliers)
Select Potential Suppliers
Select potential suppliers from all suppliers
Classification problem
Select Final Suppliers
Select final suppliers from potential suppliers and assign order quantity
Assignment problem

The interest in this research is the last two steps as can be seen in Figure 4.1. The pre-selection step is a classification problem and the final selection is an assignment problem. This research proposes an integrated method for solving these two steps. If the company has the history data for suppliers, the SVM can be used to select the potential suppliers because the SVM needs to use the history data for finding the pattern of the data. After the potential suppliers are selected with the SVM, the mathematical programming finds the final suppliers and assigns the order quantities. The procedure of solving method is as follows.

Figure 4.2 Solution Procedure for Supplier Selection

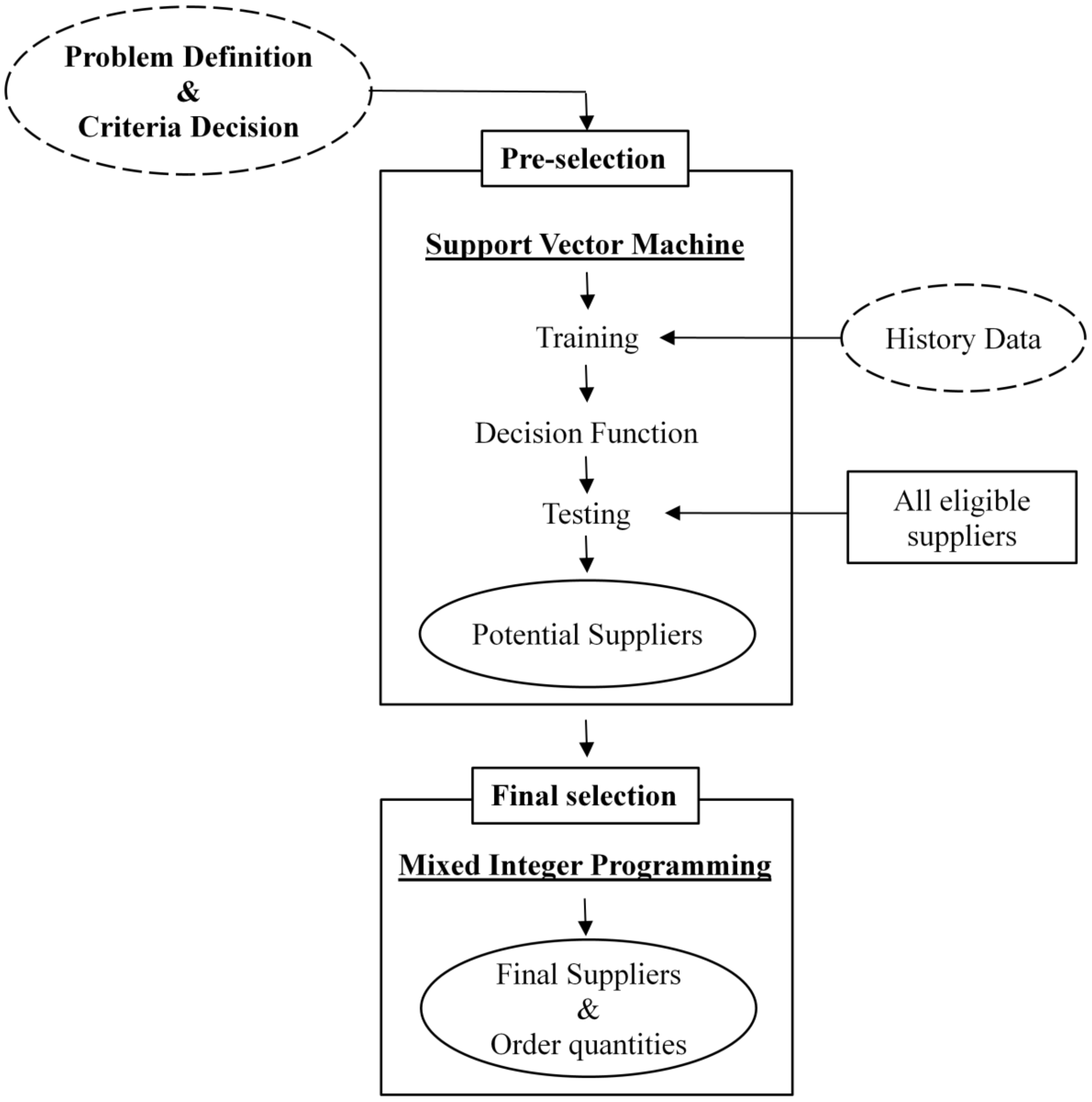


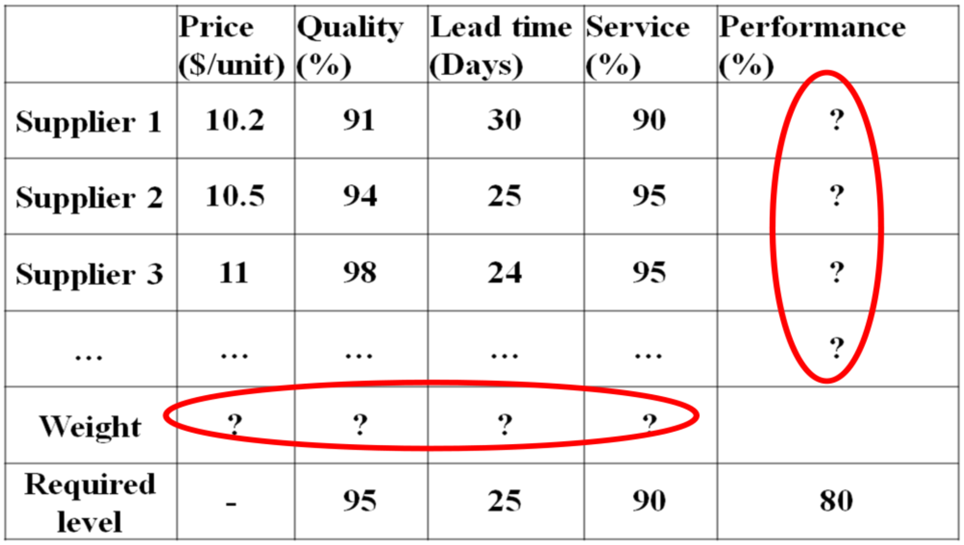
Figure 4.2 presents the solving procedure proposed in this research. The SVM is used to select the qualified suppliers. The dotted circles described the criteria and the history data given. After training with the history data, the SVM finds the pattern of the data and makes the decision function. The data for all potential suppliers are examined by the decision function. The decision function can determine each supplier is appropriate for the qualified supplier or not. Depending

on the needs or the status of the qualified suppliers, the final suppliers are selected with the order quantities in the final step using the mathematical programming model. The next two subsections describe the two steps respectively.

4.3.1 Pre-selection

The pre-selection is to select the potential suppliers out of all eligible suppliers. The company should predict two things in this step. The simple example of the supplier selection is as follows.

Figure 4.3 Example of a Supplier Selection Problem

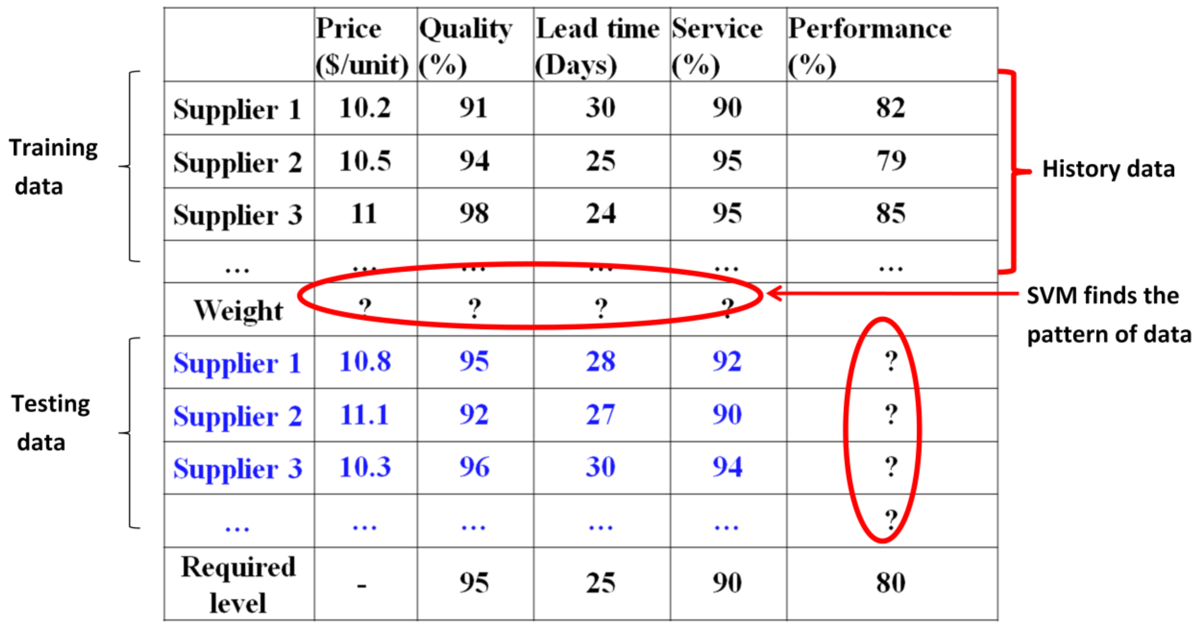


	Price ($/unit)	Quality (%)	Lead time (Days)	Service (%)	Performance (%)
Supplier 1	10.2	91	30	90	?
Supplier 2	10.5	94	25	95	?
Supplier 3	11	98	24	95	?
…	…	…	…	…	?
Weight	?	?	?	?	
Required level	-	95	25	90	80

Figure 4.3 shows an example of the pre-selection. The factors in the first row are the criteria defined at the second step. In this example, the factors are price, quality, lead time, and service. The company has the information on suppliers for these criteria, but the company doesn’t know how the suppliers will perform after selecting. And the company also wants to know which criterion is the most important, the second most, and so on, which are the weights for the factors. Therefore, the company wants to predict these performances of suppliers after finding the weights for the criteria. The pre-selection problem is to find the weights and to predict the performance of the suppliers to contract with.

This research uses the SVM for this problem. The goal for finding the weights and predicting the performance is to classify the supplier group into the qualified suppliers and the others. The machine learning method has two types. The one is a supervised method and the other is an unsupervised method. These two types are distinguished by what data used in the training phases. The supervised learning method such as the neural network and the SVM uses the data with the input and the output. In Figure 4.3, the factors are the input and the performance is the output. On the other hand, the unsupervised method such as clustering analysis using only input data. In Figure 4.3, the unsupervised method can be directly used because there are the input data and the performance is unknown. However, this research uses the SVM in the pre-selection processes. How can one use the SVM? As mentioned earlier, this research assumes that the history data can be given with the performance. Even if the company does not have the history data, the company can get the data after one more experience of the supplier selection.

Figure 4.4 Applying SVM to Pre-selection



	Price ($/unit)	Quality (%)	Lead time (Days)	Service (%)	Performance (%)
Supplier 1	10.2	91	30	90	82
Supplier 2	10.5	94	25	95	79
Supplier 3	11	98	24	95	85
…	…	…	…	…	…
Weight	?	?	?	?	
Supplier 1	10.8	95	28	92	?
Supplier 2	11.1	92	27	90	?
Supplier 3	10.3	96	30	94	?
…	…	…	…	…	?
Required level	-	95	25	90	80

Training data
Testing data
History data
SVM finds the pattern of data

Figure 4.4 shows how to apply the SVM for the pre-selection problem. From Figure 4.4, the training data have the input and the output information. The training data is the history data given. The procedure that the SVM finds the pattern of the data denotes the training phase. The pattern of the data is expressed by the support vectors which can be considered as the weights for the factors in Figure 4.4. With the pattern found from the training phase, the SVM tests the suppliers and classifies them into two groups such as qualified suppliers and the others. The details for the SVM are in the Chapter 3.

The advantages of the SVM for the pre-selection are as follows.

(1) The method has a high accuracy. The SVM as a supervised method is more accurate than the clustering analysis as an unsupervised method.

(2) The method can handle large-scale problems. If the number of factors and the data size are large scale, the AHP or statistical methods would have difficulties to arrive to a meaningful solution within the reasonable amount of time.

(3) If the company has the history data, the solution time is minimal. The training and testing tasks take several seconds or minutes even if the size of data is more than 10,000. On the contrary, using the AHP approaches for solving the pre-selection problem could be a tedious and prolonged task. For example, the purchasing team members and the experts would get the questionnaire for asking the pair-wise comparisons by all factors and then another questionnaire for the comparisons by all suppliers for each factor. After the collection of the results of the two surveys, the summary matrix is made with those results and the final weights are calculated. Then the method checks the consistency of the result. If the consistency is less than a certain tolerance, the survey is repeated until the consistency is higher than the tolerance. Though the AHP method has a lot of advantages, there are some disadvantages. The method may take too long if there is no consensus and it could be subjective based on the members surveyed.

(4) The current testing data would be the training data in the next period. After testing the data of suppliers, the results would be the history data in the next time. With larger training data, the training can be more accurate.

(5) The method does not need a complicated formulation. Regardless of the type of problem, the SVM only needs to solve a quadratic programming problem.

(6) The method allows some missing data. The SVM can solve the problem even if some data are missing.

(7) The method is objective and non-parametric. Since the SVM finds a pattern of the data, the method is more objective than other methods. Unlike statistical methods, the SVM as a machine learning method does not have parameters.

4.3.2 Final selection

The final selection step is to find the final suppliers among the potential suppliers and to allocate the order quantities to the final suppliers. In fact, if one solves the assignment problem, the final suppliers are automatically selected because the suppliers which have positive order quantities at the optimal solution are the final suppliers. The mathematical programming is a useful tool for the allocation problem. The decision variable in the mathematical programming can be defined the amount or fraction of each assignment so that the optimal solution is the final allocation. This research uses a mixed integer linear programming model for the final selection.

The objective function and constraints for the final selection model are dependent on the procurement strategies used by the company. For instance, if the company focuses on the quality of the product, the objective function may maximize the quality of the product and one of constraints is to restrict the minimum level of the quality. If another company may be interested in other factors, then the objective function and constraints would be changed according to the strategy and factors. Therefore, the objective function and constraints in the mathematical programming model for the final selection may be different by each company.

Ho et al. (2010) showed the most popular factors in the literature are quality, delivery, and price. This research considers price, quality, delivery, and service. The mathematical model in this research is based on the model proposed by Pan (1989). Pan (1989) proposed a simple linear programming model for the supplier selection problem. The decision variables are fractions of order quantities for the final suppliers. He et al. (2009) suggested an integrated method of the chance constrained programming model and the genetic algorithm. He et al. (2009) used Pan's linear programming model and the experimental results showed the solution tended to be extreme values. For instance, only a few suppliers are assigned to orders. To avoid these fragmented solutions, He et al. (2009) introduced a tight constraint in the model, but the results have not been different. In this respect, this research modified the Pan's model into a mixed integer linear programming. The details are as follows.

Definitions

Set

$N = \{1,2,\dots,n\}$: number of the potential suppliers

Variables

x_i : order quantity for the supplier i, $i \in N$

y_i : 1 if the supplier i is given any order and 0 otherwise, $i \in N$

Define parameters

Q: level of quality to achieve (%)

L: level of lead time to achieve (days)

S: level of service to achieve (%)

D: demand from production

q_i : value of quality of the supplier i, $i \in N$

l_i : value of lead time of the supplier i, $i \in N$

s_i : value of service of the supplier i, $i \in N$

p_i : value of price of the supplier i, $i \in N$

c_i : value of capacity of the supplier i, $i \in N$

o_i : ordering cost of the supplier i, $i \in N$

Mathematical Model

(FSP)
$$\min_x \sum_{i\in N} p_i x_i + \sum_{i\in N} o_i y_i \tag{4.1}$$
$$s.t. \quad \sum_{i\in N} q_i x_i \geq Q \tag{4.2}$$
$$\sum_{i\in N} l_i x_i \leq L \tag{4.3}$$
$$\sum_{i\in N} s_i x_i \geq S \tag{4.4}$$
$$\sum_{i\in N} x_i \geq D \tag{4.5}$$
$$c_i y_i - x_i \geq 0, \text{ for all } i \in N \tag{4.6}$$
$$x_i \geq 0, \text{ for all } i \in N \tag{4.7}$$
$$y_i \in \{0,1\}, \text{ for all } i \in N \tag{4.8}$$

The final selection model is a mixed integer linear programming problem. The objective function is to minimize the aggregated prices and the ordering costs. The constraints (4.2~4) denote the company should get higher levels of quality, lead time, and service from the final suppliers. The constraint (4.5) shows the amount of orders should satisfy the demand from production site. The constraint (4.6) is the capacity constraint that the maximum unit from the supplier i is c_i. The amount of order cannot be negative value in (4.7) and the variable y_i is a binary in (4.8).

4.4 Experimental Results

This section shows the experimental results for the supplier selection. This research randomly generated some data for the pre-selection. The sample problems are based on the examples in Pan (1989) and He et al. (2009). The data are randomly generated from the uniform distribution with the ranges as follows.

Table 4.2 Property of Data Set

Factors	**Distributions**	**Required level**
Price	U[10.0. 11.5]	-
Quality (%)	U[91, 99]	>= 95
Lead time (days)	U[22, 30]	<= 25
Service (%)	U[87, 97]	>= 90

These values in Table 4.2 are generated for each supplier and the response value so called the label is generated as well. The response value denotes the result of the supplier. If the supplier is successful, the response value is 1. If the supplier is not successful, the response value is -1. First, the values of the four factors are randomly generated. These values are checked if each supplier satisfies the required level. Among the suppliers that satisfy the requirement, it is randomly assigned 20% of the suppliers as the successful suppliers. The size of instances is from 30 to $5{,}000$.

The algorithm was implemented in C programming language, complied using gcc and run on a Fedora 13, 64 bit Red Hat Linux machine with 4 GB memory and Intel i7 QuadCore Bloomfield CPU running 2675 MHz. Four different training data were trained with the SVM and six different testing data were tested with the decision function of the SVM.

Table 4.3 Experimental Results of Pre-selection using SVM

Training data size	Testing data size	Accuracy (%)	Training time (second)
500	30	86.67	0.04
	100	84.00	
	500	67.00	
	1000	85.80	
	3000	85.30	
	5000	85.56	
1000	30	86.67	2.46
	100	88.00	
	500	66.60	
	1000	86.20	
	3000	86.00	
	5000	85.86	
3000	30	86.67	18.54
	100	88.00	
	500	67.00	
	1000	86.50	
	3000	86.44	
	5000	86.28	
5000	30	86.67	37.18
	100	88.00	
	500	66.60	
	1000	86.70	
	3000	86.27	
	5000	85.92	

Table 4.3 shows the results of experiments for the pre-selection. When the size of the training data is 3000 or 5000, the accuracies are a little better than smaller ones. The method of the experiments is AA method described in Chapter 3. The kernel function is the Gaussian function and the value of ν is 0.3. The data generated by the rules of Table 4.2 need to be normalized. The normalization in the SVM improves the performance significantly described in Ali and Smith-Miles (2006). This research uses the Min-Max normalization method as follows.

$$d_i^{norm} = \frac{d_i - d_{min}}{d_{max} - d_{min}} \times (U - L) + L \qquad (4.9)$$

(4.9) is the formulation of the Min-Max normalization. d_i denotes the value of the supplier i for a certain factor. d_i^{norm} is the normalized value. d_{min} and d_{max} are the minimum and the maximum value of the column. U and L are the upper and lower bound. This research uses 0 and 1 for the lower and upper bound.

The results show there is only a little difference for the accuracy among different sizes of the training data. The accuracy in the results is appropriate because the response values of the suppliers are randomly generated out of the suppliers which satisfy the minimum levels of the requirements. Therefore, the data may have an exact pattern or not. The training time is from 0.04 to 37.18 seconds. The testing time takes only a few seconds. If the AHP was used to the problem, it takes more time than the SVM because the AHP needs to do several surveys and make a consensus. Though the SVM can be applied to the only case the data of all suppliers are available, the SVM can be an attractive alternative for solving the pre-selection problem if the company has the history data for suppliers.

4.5 Conclusions

The supplier selection is an important issue in many companies that should purchase materials or components from supplier companies. The problem is directly related with the quality of product and the profits. This chapter focuses on the two steps: the pre-selection and the final selection among the procedure of the supplier selection. The pre-selection step is a classification problem while the final selection step is an assignment problem. This chapter proposed an integrated solving approach that is the combination of the SVM and the mixed integer programming. Once a company selects suppliers to make contracts, the company wants to work with the suppliers for a while because the company could spend additional time and money for re-evaluating other suppliers. Therefore, the company should select suppliers carefully. In this respect, the integrated solving approach can be more attractive than single approach.

The SVM is applied for the pre-selection step. The SVM classifies all potential suppliers into the qualified suppliers and the others. Using the SVM, the history data including the response values are needed. Companies may have the history data or can have their history data from their previous transactions and selections. The SVM as a supervised learning method has some advantages such as good accuracy, handling large scale and missing data, and so on. With the supplier's data, the SVM finds the pattern of the data which can be considered as the weights for the factors. The pattern of the data can be expressed as a model that consists of some vectors called as support vectors. The decision function uses the model of the data pattern and tests other suppliers to be classified. After the classification, the results can be added to the history data for the suppliers and used to the next pre-selection problem.

In the final selection step, this research formulated a mixed integer programming model. The decision variable is the amount of materials or components to purchase from each supplier. The constraints are the requirements of quality, lead time, demand, and service that are the levels satisfaction of the company. In the optimal solution, the suppliers corresponding to the variables that have non-zero values are the final suppliers and the other suppliers are not selected for the final suppliers. The solution provides both the final suppliers and their order quantities.

The experimental results showed the SVM for the pre-selection step can handle large scale problems and have an appropriate accuracy if there is the history data for suppliers. The accuracies for most instances are more than 85%. Once the history data was trained to find the pattern, the decision function from the training can be applied to any size of the test problems. The solution time is very reasonable to practical applications.

The chapter showed the SVM has a great potential for solving the pre-selection problem as a classification problem. The supplier selection is one of essential issues in many companies. The history data are necessary for using the SVM in the supplier selection problem. The data for some new suppliers may not exist and cannot be applied to the SVM. However, the selections with other methods would be the history data for those suppliers and could be applied to the SVM in the future.

CHAPTER 5 - Financial Problem using SVM

5.1 Company Credit Rating Classification

In this chapter, SVM solution method is applied to a real world problem. The credit prediction of the company is an essential part of the decision procedure for a loan or an investment in banks or financial companies. This research uses the data of Korean companies. The main factors have been chosen by a statistical method to be used for the SVM training. The SVM finds the pattern of the data and classify the companies into financially healthy ones and the others. The experimental results show that the newly proposed method has a great potential to become a good alternative solving approach for large-scale SVM problems.

The credit rating of a company is important for investors as well as the company itself because it is a crucial measure of the decision of the investment or loan to the company. If the evaluation of the credit rating is wrong, investors and the company may have a big loss. The accurate evaluation or prediction of the credit rating can provide the information of the proper companies to invest and the indication of bankruptcy as well. Investors may adopt the credit ratings from external credit rating agencies or try to estimate their internal ratings. In both cases, the method of rating is important. The financial data of the company are closely related to the credit rating. Many analytical methods from statistics, mathematical model, and data mining have been applied to find the main factors to affect the credit rating and the interactions between the factors. The SVM as a supervised machine learning method has been a popular classification tool which recognizes the pattern from the data itself. This research uses the SVM to predict the credit ratings of companies to classify them into two groups.

The data is provided by Korea Small Business Institute in Korea and has the financial ratios of small and medium sized companies in Korea. Min et al. (2006) used four feature subsets out of 32 financial ratios categorized by stability, profitability, growth, activity, and cash flow. On the other hand, this research considered the two types of features such as profitability and stability as following table.

Table 5.1 Financial Ratios

Type	**Ratios**	**Calculation**
Profitability	Return on equity ordinary income	Ordinary income / total capitals
	Return on Sales	Net income / sales
	Return on equity	Net income / total capitals
	Equity turnover ratio	Sales / total capitals
	Fixed asset turnover ratio	Sales / fixed assets
Stability	Equity ratio	Total capitals / total assets
	Fixed asset ratio	Fixed assets / total capitals
	Debt ratio	Total liabilities / total capitals
	Current ratio	Fixed assets / current liabilities

The data consists of nine financial ratios of companies and their credit ratings in 2008 fiscal year. The credit rating starts from the best 'AA+' to the worst 'D'.

5.2 Experimental Results

The application to the real world problem has been conducted to classify the credit ratings of companies in Korea. Three sizes of data are considered such as 6324, 3302, and 1057. In each size of data, 60% of companies arbitrarily selected are used for training and 40% of companies are used for testing. Two classes are defined as good and bad groups by credit ratings to be classified. Features are nine financial ratios and the label is one of two classes. Each financial ratio is scaled from zero to one. The experimental results are as follows.

Table 5.2 Classification of credit ratings into good (A~B) and bad (C~D) groups

Size		**Feature**	**nu**	**Kernel**	**Accuracy (%)**	
Training	**Testing**				**Training**	**Testing**
3795	2529	9	0.5	Polynomial	89.54	89.57
1982	1320	9	0.5	Gaussian	90.47	90.46
635	422	9	0.3	Polynomial	90.71	90.76

Table 5.2 shows the results of classification when the credit ratings are classified into two groups with good (A, B) and bad (C, D). Polynomial and Gaussian kernel functions are used. The results have good performances about 90% of accuracy.

Table 5.3 Classification of credit ratings into good (A) and bad (B~D) groups

Size		Feature	nu	Kernel	Accuracy (%)	
Training	Testing				Training	Testing
3795	2529	9	0.4	Gaussian	94.89	94.90
1982	1320	9	0.4	Polynomial	92.79	92.81
635	422	9	0.4	Polynomial	84.10	85.55

Table 5.3 presents the results of different grouping, which classifies the companies into good (A) and bad (B, C, D) groups. If the kernel and the value of ν are different, the accuracies are changed. The experiments have done with some combinations of the value of ν and the kernel function and found the best values. The results show that the combinations of the kernel function and the value of ν are different by the size of data and the type of classification. The experimental results showed that the new algorithm has performed very well in the real world problem. The experiments have considered nine financial ratios to classify the companies into two groups such as good and bad.

5.3 Conclusions

This chapter proposed an alternative method for assessing companies with company credit ratings using SVM. Assessing companies is critical in many cases. Banks or investment companies require the measurement as accurate as possible. Individual investors also want to get more convinced information about the company to be invested. The company itself who is assessed also wants to know its current status. Thus, systematic method has become a preferable method for the measurement of company credit ratings because the assessment directly affects profits or losses.

This research used real data of small size companies in Korea. The factors considered in this research are nine financial ratios represented companies' profitability and stability status. The data include company credit ratings, so it is used 40% of data for training and 60% for testing using SVM. This research compared credit ratings in data set with predicted credit ratings from out testing. Experimental results showed that the accuracy in most cases is more than 90%, which proves the new method is viable.

Some may argue why the credit ratings need to be predicted again because the data already have all the results from an institution like Moody's that determines and announces the credit ratings for most companies. There are three reasons. First, credit ratings do not come out every month or week from the institution. Banks or investors may want to get credit ratings of companies whenever they need. Second, banks or investors can have different factors from Moody's to assess companies. Third, Moody's cannot assess all companies, but investors want to the other companies or small sized companies that are not assessed by Moody's.

Future work of this research is to extend the proposed model using additional factors including other financial factors and non-financial factors. The study for finding appropriate factors is also another interesting issue for this problem. While this research used the SVM classification technique, the SVM regression model can be used for this problem in the future. SVM is a statistical supervised machine learning method and a quantitative method. One may consider a solution approach combining SVM and other qualitative or techniques.

CHAPTER 6 - Conclusions and Future Research

6.1 Conclusions

The nonlinear knapsack problem, the support vector machine (SVM), and supplier selection are all interesting and important topics for researchers and industries. The SVM has long been a popular tool with many applications. Started from research on the SVM algorithm, this research has suggested a new solution algorithm of the subproblem of the SVM as well as an application for supply chain management (SCM). In addition to an efficient approach for solving the ν-SVM classification problem, this dissertation has proposed a new algorithm for solving the continuous separable nonlinear knapsack problem and an integrated approach for the supplier selection problem using the SVM in discussion ranging from theory to application with the nonlinear knapsack problem in Chapter 2 and the SVM and the supplier selection described in Chapter 3 and 4.

An efficient pegging algorithm was proposed for the nonlinear knapsack problem. The problem is separable, continuous, and convex with box constraints on variables. The pegging algorithm is an iterative method that fixes some variables into their optimal values at each iteration until the solution reaches the optimal. The newly developed method, the Dual Bound algorithm, is based on the Bitran-Hax algorithm, a popular pegging method. The motivation for developing this algorithm was the two time consuming calculations in the Bitran-Hax algorithm. The Dual Bound algorithm introduced the new concept of the dual bound. The dual bound can check the feasibility of each variable instead of the bound of the primal variable because all primal variables have dual bounds. Using the dual bound instead of the bounds of primal variables provides two advantages. One, the algorithm must consider only one dual variable because the primal problem has only one constraint. Two, the calculation of the dual bound does not depend on the iteration; thus it need not be calculated at each iteration. The experimental results showed the Dual Bound algorithm performed better than the Bitran-Hax algorithm as the size of the problem increased.

In Chapter 3, this research focused on the ν-SVM classification problem. The quadratic programming problem should be solved to find the decision function of the SVM. The quadratic programming problem referred to the singly linearly constrained quadratic convex problem with box constraints, which has a dense Hessian matrix, one linear equality constraint, and box constraints. It has a large and dense Hessian matrix, so the SVM problem requires a special solution approach. The sequential minimal optimization (SMO) has been a popular method for solving the SVM problems. The SMO solves smaller problems instead of solving the original problem sequentially. Though the SMO can handle large scale problems, reducing the number of iterations and developing a good method to find the smaller problems are still challenging issues. On the other hand, because the SVM problem is a convex quadratic programming problem, some studies of the SVM use the gradient or projection type methods. However, those methods can handle only small or medium scale problems even if the methods converge quickly. Therefore, the decomposition is necessary for the SVM. This research proposed using the matrix splitting method and the incomplete Cholesky composition for the ν-SVM classification problem. The original problem has a quadratic objective function, two linear constraints, and box constraints. Applying the augmented Lagrangian method, the original problem becomes the singly linearly constrained quadratic convex problem with box constraints. The overall solving procedure of the new method is based on the matrix splitting method. Several different options have been used for the line search and for updating α. The direction is found by solving a subproblem. The Dual Bound algorithm described in Chapter 2 is used to solve the subproblem. The Hessian matrix is decomposed by the incomplete Cholesky decomposition method, which is suitable because the Hessian matrix of the SVM is dense and has low rank. The experimental results showed the method performed well and solved large scale problems. The method in this research has great potential for extension because other methods for line search and updating α can be used.

An integrated approach for the supplier selection was proposed in Chapter 4. This research focused on the pre-selection and the final selection steps among the supplier selection procedure. The ν-SVM classification solves the pre-selection step and the mixed integer programming model is used for the final selection. The biggest contribution of the method in this research is in using the SVM for pre-selection. Pre-selection chooses the potential suppliers out of all eligible suppliers. Only potential suppliers can become final suppliers. In this step, the company needs only to classify the suppliers into two groups: potential suppliers and others, which is a classification problem. In previous methods, such as the analytic hierarchy process (AHP) or cluster analysis (CA), information about the performance of suppliers has not been used because these methods are unsupervised methods and are thus subjective and require lengthy calculations to get the results. On the other hand, the SVM is objective and fast. As a supervised method, the SVM requires performance information for suppliers. The SVM cannot apply to the problem without historical data. However, the SVM is more attractive than other methods if the history data are available.

This dissertation has examined three challenging issues. The nonlinear knapsack problem has many applications and can occur as a subproblem in many optimization algorithms. Thus, the new pegging algorithm, which is fast and efficient, is a significant contribution. The SVM classification, which has been a popular tool in many different areas, can be modified using matrix splitting method, several options of line search and updating α, and the incomplete Cholesky decomposition to solve SVM problems with large and dense Hessian matrices. The importance of the supplier selection has increased in the company because products have become more complicated and change so rapidly that outsourcing has become a necessary part of production. This research uses the SVM for pre-selecting such suppliers, which is a significant contribution to the field. In summary, this dissertation combines three different areas: mathematical programming, data mining, and supply chain management.

6.2 Future Research

The Dual Bound algorithm for the nonlinear knapsack problem uses the dual bound instead of the primal variables to check feasibility. In other words, checking feasibility requires only one dual variable instead of all primal variables. Moreover, calculating the dual bound is not in the loop of the iteration. Therefore, if one could calculate the dual variable that can peg the largest number of variables first, then the computations could be still more reduced because the algorithm would only check the remaining variables from the next iterations. If one can also calculate the range of the dual variable or the direction of the dual variable, then the algorithm can peg still more variables and significantly reduce the number of iterations. The concept of the dual bound can be applied to other optimization algorithms when they need to check feasibility. The Dual Bound algorithm can be combined with the fuzzy theory or stochastic programming algorithm when the problem contains uncertainties. The Dual Bound algorithm also can be applied to other problems and applications.

The new method for the ν-SVM classification problem is a flexible algorithm because a variety of different line searches and ways of updating α can be used in this algorithm. Finding the best combination of methods is the challenge. The performance of the method depends on the combination of the line search and updating methods. This research used four different combinations of the following methods: SPGM, GVPM, MSM, and AA. Other combinations could be tested. In the final solution, the number of support vectors is a little large. It needs to find a way to reduce the number of support vectors in the future. For implementation, the proposed algorithm must have an efficient data structure to store the large incomplete lower triangular matrix for the incomplete Cholesky decomposition. If the rank of the Hessian matrix can be calculated before the incomplete Cholesky decomposition, the exact amount of memory needed for the triangular matrix can be allocated in advance.

The data for pre-selecting suppliers were randomly generated. The new integrated approach should be applied to real problems. In reality, the suppliers may have new proposals that would differ from historical data. The company can also negotiate with suppliers. The negotiation or relationship with suppliers can be included in the new solution approach. The environmental or social factors and effects can also be considered in new model. The problem considered in this research focuses on a single time period. The newly proposed integrated approach can, however, be extended to problems covering multiple time periods and some uncertainties. Because the AHP works well for selecting suppliers without historical data and the SVM performs well to select suppliers while using historical data, future research could consider integrating the AHP and the SVM for pre-selection.

References

Aissaoui, N., Haouari, M., & Hassini, E. (2007). Supplier selection and order lot sizing modeling: A review. *Computers and Operations Research, 34*(12), 3516-3540.

Ali, A., Helgason, R., Kennington, J., & H., L. (1980). Computational comparison among three multicommodity network flow algorithms. *Operations Research, 28*(4), 995-1000.

Ali, S., & Smith-Miles, K. A. (2006). Improved support vector machine generalization using normalized input space. *AI 2006: Advances in Artificial Intelligence, Lecture Notes in Artificial Intelligence 4304*, 362-371.

Alzate, C., & Suykens, J. (2008). *Sparse Kernel Models for Spectral Clustering Using the Incomplete Cholesky Decomposition.* Paper presented at the 2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence), Hong Kong.

Alzate, C., & Suykens, J. A. K. (2008). A regularized kernel CCA contrast function for ICA. *Neural Networks, 21*(2-3), 170-181.

An, S. J., Liu, W. Q., & Venkatesh, S. (2007). Fast cross-validation algorithms for least squares support vector machine and kernel ridge regression. *Pattern Recognition, 40*(8), 2154-2162.

Antoniou, A., & Lu, W. S. (2007). *Practical Optimization: Algorithms and Engineering Applications* (1 ed.): Springer.

Arasu, U. (2000). *Solving Large-Scale Nonlinear Stochastic Network Problems.* Kansas State University.

Azadeh, A., Saberi, M., & Anvari, M. (2010). An integrated artificial neural network algorithm for performance assessment and optimization of decision making units. *Expert Systems with Applications, 37*(8), 5688-5697.

Bach, F. R., & Jordan, M. I. (2005). *Predictive low-rank decomposition for kernel methods.* Paper presented at the the 22nd International Conference on Machine Learning.

Barzilai, J., & Borwein, J. M. (1988). Two-point step size gradient methods. *IMA Journal of Numerical Analysis, 8*(1), 141-148.

Bazaraa, M. S., Sherali, H. D., & Shetty, C. M. (1993). *Nonlinear Programming : Theory and Algorithms* (2 ed.). New York: John Wiley & Sons.

Bender, P. S., Brown, R. W., Isaac, M. H., & Shapiro, J. F. (1985). Improving Purchasing Productivity at Ibm with a Normative Decision Support System. *Interfaces, 15*(3), 106-115.

Bennett, K. P., & Bredensteiner, E. J. (1997). A Parametric Optimization Method for Machine Learning. *INFOMS Journal on Computing, 9*(3), 311-318.

Bennett, K. P., & Bredensteiner, E. J. (2000). *Duality and Geometry in SVM Classifiers.* Paper presented at the the Seventeenth International Conference on Machine Learning, San Francisco.

Birgin, E. G., Martinez, J. M., & Raydan, M. (2000). Nonmonotone spectral projected gradient methods on convex sets. *Siam Journal on Optimization, 10*(4), 1196-1211.

Bitran, G. R., & Hax, A. C. (1981). Disaggregation and Resource-Allocation Using Convex Knapsack-Problems with Bounded Variables. *Management Science, 27*(4), 431-441.

Bretthauer, K. M., Ross, A., & Shetty, B. (1999). Nonlinear integer programming for optimal allocation in stratified sampling. *European Journal of Operational Research, 116*(3), 667-680.

Bretthauer, K. M., & Shetty, B. (2002). The nonlinear knapsack problem - algorithms and applications. *European Journal of Operational Research, 138*(3), 459-472.

Bretthauer, K. M., Shetty, B., & Syam, S. (2003). A specially structured nonlinear integer resource allocation problem. *Naval Research Logistics, 50*(7), 770-792.

Buffa, F. P., & Jackson, W. M. (1983). A Goal Programming Model for Purchase Planning. *Journal of Purchasing and Materials Management, 19*(3), 27-34.

Cai, L., Song, F., & Yuan, D. (2008). *Study on the application of SVM in supplier primary election.* Paper presented at the 1st International Workshop on Knowledge Discovery and Data Mining, WKDD, Adelaide, SA, Australia.

Camps-Valls, G., Munoz-Mari, J., Gomez-Chova, L., Richter, K., & Calpe-Maravilla, J. (2009). Biophysical Parameter Estimation With a Semisupervised Support Vector Machine. *Ieee Geoscience and Remote Sensing Letters, 6*(2), 248-252.

Carbonneau, R., Laframboise, K., & Vahidov, R. (2008). Application of machine learning techniques for supply chain demand forecasting. *European Journal of Operational Research, 184*(3), 1140-1154.

Ç ebi, F., & Bayraktar, D. (2003). An integrated approach for supplier selection. *Logistics Information Management, 16*(6), 395 - 400.

Chang, C. C., & Lin, C. J. (2001). Training nu-support vector classifiers: Theory and algorithms. *Neural Computation, 13*(9), 2119-2147.

Che, Z. H., & Wang, H. S. (2008). Supplier selection and supply quantity allocation of common and non-common parts with multiple criteria under multiple products. *Computers & Industrial Engineering, 55*(1), 110-133.

Conn, A. R., Gould, N. I. M., & Toint, P. L. (1988). Global Convergence of a Class of Trust Region Algorithms for Optimization with Simple Bounds. *Siam Journal on Numerical Analysis, 25*(2), 433-460.

Cottle, R. W., Duvall, S. G., & Zikan, K. (1986). A Lagrangean Relaxation Algorithm for the Constrained Matrix Problem. *Naval Research Logistics, 33*(1), 55-76.

Crisp, D. J., & Burges, C. J. C. (2000). *A Geometric Interpretation of v-SVM Classifiers.* Paper presented at the Neural Information Processing Systems (NIPS).

Crow, L. E., Olshavsky, R. W., & Summers, J. O. (1980). Industrial Buyers Choice Strategies - a Protocol Analysis. *Journal of Marketing Research, 17*(1), 34-44.

Dai, Y. H., & Fletcher, R. (2006). New algorithms for singly linearly constrained quadratic programs subject to lower and upper bounds. *Mathematical Programming, 106*(3), 403-421.

Dai, Y. H., & Zhang, H. C. (2001). Adaptive two-point stepsize gradient algorithm. *Numerical Algorithms, 27*(4), 377-385.

de Boer, L., Labro, E., & Morlacchi, P. (2001). A review of methods supporting supplier selection. *European Journal of Purchasing & Supply Management, 7*(2), 75-89.

Debnath, R., & Takahashi, H. (2006). *SVM training: Second-order cone programming versus quadratic programming.* Paper presented at the IEEE International Conference on Neural Networks.

Degraeve, Z., Labro, E., & Roodhooft, H. (2004). Total cost of ownership purchasing of a service: The case of airline selection at Alcatel Bell. *European Journal of Operational Research, 156*(1), 23-40.

Demirtas, E. A., & Ü stün, Ö . (2008). An integrated multiobjective decision making process for supplier selection and order allocation. *Omega, 36*(1), 76-90.

Dickson, G. W. (1966). An analysis of vendor selection: systems and decisions. *Journal of Purchasing, 2*(1), 5-17.

Dussault, J. P., Ferland, J. A., & Lemaire, B. (1986). Convex Quadratic-Programming with One Constraint and Bounded Variables. *Mathematical Programming, 36*(1), 90-104.

Ellram, L. M. (1995). Total cost of ownership: an analysis approach for purchasing. *International Journal of Physical Distribution & Logistics Management, 25*(8), 4-23.

Eu, J. H. (1991). The Sampling Resource-Allocation Problem. *Ieee Transactions on Communications, 39*(9), 1277-1279.

Evtushenko, Y. G., & Zhadan, V. G. (1994). Barrier-Projective Methods for Nonlinear Programming. *Computational Mathematics and Mathematical Physics, 34*(5), 579-590.

Ferris, M. C., & Munson, T. S. (2003). Interior-point methods or massive support vector machines. *Siam Journal on Optimization, 13*(3), 783-804.

Fine, S., & Scheinberg, K. (2001). Efficient SVM training using lower rank kernal representations. *Journal of Machine Learning Research, 2*, 243-264.

Friedlander, A., Martinez, J. M., Molina, B., & Raydan, M. (1998). Gradient method with retards and generalizations. *Siam Journal on Numerical Analysis, 36*(1), 275-289.

Fu, Y. S., & Dai, Y. H. (2010). Improved Projected Gradient Algorithms for Singly Linearly Constrained Quadratic Programs Subject to Lower and Upper Bounds. *Asia-Pacific Journal of Operational Research, 27*(1), 71-84.

Ghodsypour, S. H., & O'Brien, C. (1998). A decision support system for supplier selection using an integrated analytic hierarchy process and linear programming. *International Journal of Production Economics, 56-7*, 199-212.

Ghodsypour, S. H., & O'Brien, C. (2001). The total cost of logistics in supplier selection, under conditions of multiple sourcing, multiple criteria and capacity constraint. *International Journal of Production Economics, 73*(1), 15-27.

Goldfarb, D., & Scheinberg, K. (2008). Numerically stable LDLT factorizations in interior point methods for convex quadratic programming. *Ima Journal of Numerical Analysis, 28*(4), 806-826.

Golub, G. H., & Van Loan, C. F. (1996). *Matrix Computations* (3rd ed.): The Johns Hopkins University Press.

Guosheng, H., & Guohong, Z. (2008). Comparison on neural networks and support vector machines in suppliers' selection. *Journal of Systems Engineering and Electronics, 19*(2), 316-320.

He, S. W., Chaudhry, S. S., Lei, Z. L., & Wang, B. H. (2009). Stochastic vendor selection problem: chance-constrained model and genetic algorithms. *Annals of Operations Research, 168*(1), 169-179.

Higham, N. J. (1990). Analysis of the Cholesky decomposition of a semi-definite matrix *Reliable Numerical Computation* (pp. 161-185): Oxford University Press.

Hinkle, C. L., Robinson, P. J., & Green, P. E. (1969). Vendor Evaluation Using Cluster Analysis. *Journal of Purchasing, 5*, 49-58.

Ho, W., Xu, X. W., & Dey, P. K. (2010). Multi-criteria decision making approaches for supplier evaluation and selection: A literature review. *European Journal of Operational Research, 202*(1), 16-24.

Holt, G. D. (1998). Which contractor selection methodology? *International Journal of Project Management, 16*(3), 153-164.

Hong, G. H., Park, S. C., Jang, D. S., & Rho, H. M. (2005). An effective supplier selection method for constructing a competitive supply-relationship. *Expert Systems with Applications, 28*(4), 629-639.

Hsu, C. F., Chang, B., & Hung, H. F. (2007). *Applying SVM to build supplier evaluation model - comparing likert scale and fuzzy scale.* Paper presented at the IEEE International Conference on Industrial Engineering and Engineering Management.

Ibaraki, T., & Katoh, N. (1988). *Resource Allocation Problems : Algorithmic Approaches*. Cambridge: MIT Press.

Joachims, T. (1999). Making large-scale support vector machine learning practical *Advances in kernel methods: support vector learning* (pp. 169-184). Cambridge: MIT Press.

Kashima, H., Ide, T., Kato, T., & Sugiyama, M. (2009). Recent Advances and Trends in Large-Scale Kernel Methods. *Ieice Transactions on Information and Systems, E92d*(7), 1338-1353.

Keller, H. B. (1965). On the Solution of Singular and Semidefinite Linear Systems by Iteration. *SIAM Journal of Numerical Analysis, 2*(2), 281-290.

Kianmehr, K., & Alhajj, R. (2006). *Support Vector Machine Approach for Fast Classification.* Paper presented at the the International Conference on Data Warehouse and Knowledge Discovery, Poland.

Klingman, D., Napier, A., & Stutz, J. (1974). Netgen - Program for Generating Large-Scale Capacitated Assignment, Transportation, and Minimum Cost Flow Network Problems. *Management Science Series a-Theory, 20*(5), 814-821.

Kodialam, M. S., & Luss, H. (1998). Algorithms for separable nonlinear resource allocation problems. *Operations Research, 46*(2), 272-284.

Kokangul, A., & Susuz, Z. (2009). Integrated analytical hierarch process and mathematical programming to supplier selection problem with quantity discount. *Applied Mathematical Modelling, 33*(3), 1417-1429.

Kumar, M., Vrat, P., & Shankar, R. (2004). A fuzzy goal programming approach for vendor selection problem in a supply chain. *Computers & Industrial Engineering, 46*(1), 69-85.

Kuo, R. J., Hong, S. Y., & Huang, Y. C. (2010). Integration of particle swarm optimization-based fuzzy neural network and artificial neural network for supplier selection. *Applied Mathematical Modelling, 34*(12), 3976-3990.

Lee, Y. J., & Mangasarian, O. L. (2001). *RSVM: Reduced support vector machines.* Paper presented at the First SIAM International Conference on Data Mining, Chicago.

Li, X. Y., Li, H., & Zhou, Y. C. (2005). *Collaborative identification of coordination questions in supply chain based on support vector machines.* Paper presented at the 2005 International Conference on Machine Learning and Cybernetics, ICMLC 2005, Guangzhou, China.

Liao, Z., & Kuhn, A. (2004). *Operational integration of supplier selection and procurement lot sizing in supply chain.* Paper presented at the Global project and manufacturing management symposium, Siegen, Germany.

Lin, C. J., Lucidi, S., Palagi, L., Risi, A., & Sciandrone, M. (2009). Decomposition Algorithm Model for Singly Linearly-Constrained Problems Subject to Lower and Upper Bounds. *Journal of Optimization Theory and Applications*, *141*(1), 107-126.

Lin, C. J., & Saigal, R. (2000). An incomplete Cholesky factorization for dense symmetric positive definite matrices. *Bit, 40*(3), 536-558.

Lin, Y. Y., & Pang, J. S. (1987). Iterative Methods for Large Convex Quadratic Programs - a Survey. *Siam Journal on Control and Optimization*, *25*(2), 383-411.

Liu, J., Ding, F. Y., & Lall, V. (2000). Using data envelopment analysis to compare suppliers for supplier selection and performance improvement. *Supply Chain Management: An International Journal, 5*(3), 143-150.

Louradour, J., Daoudi, K., & Bach, F. (2006). *SVM speaker verification using an incomplete cholesky decomposition seqeunce kernel.* Paper presented at the IEEE Odyssey.

Luo, Z., & Tseng, P. (1992). Error bound and convergence analysis of matrix splitting algorithms for the affine variational inequality problem. *SIAM Journal of Optimization, 2*(1), 43-54.

Mangasarian, O., L. (1969). *Nonlinear Programming*. New York: McGraw-Hill.

Markowitz, H. (1952). Portfolio Selection. *The Journal of Finance, 7*(1), 77-91.

Martello, S., & Toth, P. (1990). *Knapsack Problems : Algorithms and Computer Implementations*. New York: John Wiley & Sons.

Mavroforakis, M. E., & Theodoridis, S. (2006). A geometric approach to support vector machine (SVM) classification. *Ieee Transactions on Neural Networks, 17*(3), 671-682.

Min, S. H., Lee, J., & Han, I. (2006). Hybrid genetic algorithms and support vector machines for bankruptcy prediction. *Expert Systems with Applications*, *31*(3), 652-660.

Monczka, R. M., & Trecha, S. J. (1988). Cost-Based Supplier Performance Evaluation. *Journal of Purchasing & Materials Management, 24*(1), 2-7.

Nehate, G. (2006). *Solving Large Scale Support Vector Machine Problems using Matrix Splitting and Decomposition methods.* Kansas State University.

Ng, S. T., Smith, N. J., & Skidmore, R. M. (1995). Case-based reasoning for contractor prequalification a feasibility study. *Developments in Artificial Intelligence for Civil and Structural Engineering,* 61-66.

Ohuchi, A., & Kaji, I. (1984). Lagrangian Dual Coordinatewise Maximization Algorithm for Network Transportation Problems with Quadratic Costs. *Networks*, *14*(4), 515-530.

Ortega, J. M. (1972). *Numerical analysis; a second course*. New York: Academic Press.

Osuna, E., Freud, R., & Girosi, F. (1997). *An improved training algorithm for support vector machines.* Paper presented at the IEEE Workshop, New York.

Pan, A. C. (1989). Allocation of order quantity among suppliers. *Journal of Purchasing and Materials Management, 25*, 36-39.

Pang, J. S. (1982). On the Convergence of a Basic Iterative Method for the Implicit Complementarity-Problem. *Journal of Optimization Theory and Applications, 37*(2), 149-162.

Papagapiou, A., Mingers, J., & Thanassoulis, E. (1997). Would you buy a used car with DEA? *OR Insight*, *10*(1), 13-19.

Pardalos, P. M., & Kovoor, N. (1990). An Algorithm for a Singly Constrained Class of Quadratic Programs Subject to Upper and Lower Bounds. *Mathematical Programming, 46*(3), 321-328.

Patriksson, M. (2008). A survey on the continuous nonlinear resource allocation problem. *European Journal of Operational Research, 185*(1), 1-46.

Platt, J. C. (1999). Fast training of support vector machines using sequential minimal optimization *Advances in kernel methods: support vector learning* (pp. 185-208). Cambridge: MIT Press.

Raydan, M. (1997). The Barzilai and Borwein gradient method for the large scale unconstrained minimization problem. *Siam Journal on Optimization, 7*(1), 26-33.

Robinson, A. G., Jiang, N., & Lerme, C. S. (1992). On the Continuous Quadratic Knapsack-Problem. *Mathematical Programming, 55*(1), 99-108.

Rosenblatt, F. (1962). *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms*: Spartan Books.

Rosenblatt, M. J., Herer, Y. T., & Hefter, I. (1998). Note. An acquisition policy for a single item multi-supplier system. *Management Science, 44*(11), S96-S100.

Sarfaraz, A. R., & Balu, R. (2006). *An Integrated Approach for Supplier Selection.* Paper presented at the Industrial Informatics, 2006 IEEE International Conference, Singapore.

Schölkopf, B., Smola, A. J., Williamson, R. C., & Bartlett, P. L. (2000). New support vector algorithms. *Neural Computation, 12*(5), 1207-1245.

Serafini, T., Zanghirati, G., & L., Z. (2005). Gradient projection methods for quadratic programs and applications in training support vector machines. *Optimization Methods and Software, 20*(2-3), 353-378.

Shouquan, L., & Zhiwen, Z. (2007). *Multi-echelon supply chain demand forecast based on support vector machines.* Paper presented at the International Conference on Transportation Engineering 2007, ICTE 2007.

Smytka, D. L., & Clemens, M. W. (1993). Total Cost Supplier Selection Model: A Case Study. *Journal of Supply Chain Management, 29*(1), 42-49.

Stefanov, S. M. (2004). Polynomial algorithms for projecting a point onto a region defined by a linear constraint and box constraints. *Journal of Applied Mathematics, 2004*(5), 409-431.

Sun, H. L., Xie, J. Y., & Xue, Y. F. (2005). *An SVM-based model for supplier selection using fuzzy and pairwise comparison.* Paper presented at the 2005 International Conference on Machine Learning and Cybernetics, Guangzhou, China.

Suykens, J. A. K., & Vandewalle, J. (1999). Least squares support vector machine classifiers. *Neural Processing Letters, 9*(3), 293-300.

Tamir, A. (1980). Efficient Algorithms for a Selection Problem with Nested Constraints and Its Application to a Production-Sales Planning-Model. *Siam Journal on Control and Optimization, 18*(3), 282-287.

Tempelmeier, H. (2002). A simple heuristic for dynamic order sizing and supplier selection with time-varying data. *Production and Operations Management, 11*(4), 499-515.

Timmerman, E. (1986). An Approach to Vendor Performance Evaluation. *Purchasing and Materials Management, 22*(4), 1-7.

Ting, S. G., & Cho, D. I. (2008). An integrated approach for supplier selection and purchasing decisions. *Supply Chain Management-an International Journal, 13*(2), 116-127.

To, K. N., Lim, C. C., Teo, K. L., & Liebelt, M. J. (2001). Support vector learning with quadratic programming and adaptive step size barrier-projection. *Nonlinear Analysis-Theory Methods and Applications, 47*(8), 5623-5633.

Tseng, T. L., Huang, C. C., Jiang, F., & Ho, J. C. (2006). Applying a hybrid data-mining approach to prediction problems: a case of preferred suppliers prediction. *International Journal of Production Research, 44*(14), 2935-2954.

Vapnik, V. (1979). *Estimation of Dependences Based on Empirical Data*. Moscow: Nauka.

Vapnik, V., & Chervonenkis, A. (1974). *Theory of Pattern Recognition*. Moscow: Nauka.

Ventura, J. A. (1991). Computational Development of a Lagrangian Dual Approach for Quadratic Networks. *Networks, 21*(4), 469-485.

Wan, X. T., Pekny, J. F., & Reklaitis, G. V. (2005). Simulation-based optimization with surrogate models - Application to supply chain management. *Computers & Chemical Engineering, 29*(6), 1317-1328.

Wang, G., Huang, S. H., & Dismukes, J. P. (2004). Product-driven supply chain selection using integrated multi-criteria decision-making methodology. *International Journal of Production Economics, 91*(1), 1-15.

Weber, C. A., Current, J. R., & Benton, W. C. (1991). Vendor Selection Criteria and Methods. *European Journal of Operational Research, 50*(1), 2-18.

Weber, C. A., Current, J. R., & Desai, A. (1998). Non-cooperative negotiation strategies for vendor selection. *European Journal of Operational Research, 108*(1), 208-223.

Weber, C. A., & Desai, A. (1996). Determination of paths to vendor market efficiency using parallel coordinates representation: A negotiation tool for buyers. *European Journal of Operational Research, 90*(1), 142-155.

Weber, C. A., & Ellram, L. M. (1993). Supplier selection using multi-objective programming: a decision support system approach. *International Journal of Physical Distribution & Logistics Management, 23*(2), 3-14.

Wen, L., & Li, J. (2006). *Research of Credit Grade Assessment for Suppliers Based on Multi-Layer SVM Classifier.* Paper presented at the Sixth International Conference on Intelligent Systems Design and Applications (ISDA'06), Jinan, China.

Wright, P. (1975). Consumer Choice Strategies - Simplifying Vs Optimizing. *Journal of Marketing Research, 12*(1), 60-67.

Wu, C. H. (1993). *Solving large-scale nonlinear network problems with relaxation and decomposition algorithms.* Pennsylvania State University.

Xiaohui, H., Xiuxia, Y., & Hu, Y. (2007). *The model of order prediction based on SVM.* Paper presented at the The 7th International Conference on Intelligent Systems Design and Applications, ISDA 2007, Rio de Janeiro, Brazil.

Yue, L., Yafeng, Y., Junjun, G., & Chongli, T. (2007). *Demand forecasting by using support vector machine.* Paper presented at the Third International Conference on Natural Computation, ICNC 2007, Haikou, China.

Zhan, Y., & Shen, D. (2005). Design efficient support vector machine for fast classification. *Pattern Recognition, 38*(1), 157-161.

Zhou, W. D., Zhang, L., & Jiao, L. C. (2002). Linear programming support vector machines. *Pattern Recognition, 35*(12), 2927-2936.