USING PREFERENCE LEARNING FOR MULTI-OBJECTIVE OPTIMIZATION WITH APPLICATIONS IN SUPPLY CHAIN

by
ZEREN ALPOĞUZ

Submitted to the Graduate School of Engineering and Natural Sciences
in partial fulfillment of
the requirements for the degree of Master of Science

Sabancı University
December 2023

ABSTRACT

USING PREFERENCE LEARNING FOR MULTI-OBJECTIVE OPTIMIZATION WITH APPLICATIONS IN SUPPLY CHAIN

ZEREN ALPOĞUZ

Industrial Engineering, MSc. Thesis, December 2023

Thesis Supervisor: Asst. Prof. Ezgi KARABULUT TÜRKSEVEN

Keywords: Multi-criteria Decision Making, Multi-objective Optimization, Preference Learning, Weighted-Sum Method, Rank-SVM, Supply Chain Network

Choosing the right weight is a challenging task in solving a multi-criteria decision making (MCDM) problems. We utilize the learning-to-rank machine learning approach, Rank SVM, to learn the criteria weights in MCDM. As the training data, Rank SVM needs the pairwise preferences of the alternatives, as revealed by the decision maker (DM). We develop three strategies in offering alternative pairs to DMs. The first strategy is offering pairs from the Pareto frontier which represents a set of optimal solutions, the second strategy is offering pairs from the feasible region meaning dominated and non-dominated solutions that are possible given the constraints and the third one is offering pairs from the utopian space that covers both feasible and infeasible solutions. The main objective of this study is to evaluate the impact of offering pairs from different regions on the learning process of Rank SVM and utilizing information learned in data generation strategies. To evaluate the performance and effectiveness of our strategies, we chose a three-echelon supply chain network problem as our test case. Experimental results obtained from three different settings provide a practical evaluation. We observe distinct impacts between strategies in offering alternative pairs; some strategies yield more accurate or consistent results than others. This highlights the importance of the source of alternative pairs in the effectiveness of preference learning algorithms. In addition, the use of learning information in the generation of training data provided a significant improvement except the Utopian region strategy.

ÖZET

ÇOK AMAÇLI OPTİMİZASYONDA TERCİHLİ ÖĞRENMEYİ KULLANMA VE TEDARİK ZİNCİRİNDEKİ UYGULAMASI

ZEREN ALPOĞUZ

Endüstri Mühendisliği, Yüksek Lisans Tezi, Aralık 2023

Tez Danışmanı: Dr. Öğr. Üyesi Ezgi KARABULUT TÜRKSEVEN

Anahtar Kelimeler: Çok Kriterli Karar Verme, Çok Amaçlı Optimizasyon, Tercihli Öğrenme, Ağırlıklı Toplam Yöntemi, Rank-SVM, Tedarik Zinciri Ağı

Çok kriterli karar verme (ÇKKV) problemlerini çözmede doğru kriter ağırlıkların belirlenmesi zorlu bir iştir. ÇKKV'deki kriter ağırlıklarını öğrenmek için, sıralamaya dayalı makine öğrenme yaklaşımlarından biri olan olan Sıralama Destek Vektör Makinesi kullanılmaktadır. Sıralama Destek Vektör Makinesi için eğitim verisi, karar vericinin belirlediği alternatiflerin ikili tercihlerinden oluşmaktadır. Bu çalışmada karar vericilere ikililer önermek için üç farklı strateji geliştirilmektedir. İlk strateji, en iyi çözümler setini temsil eden Pareto sınırından ikililer sunmaktır; ikinci strateji, kısıtlamalar göz önünde bulundurularak domine edilemeyen ve domine edilebilen çözümleri içeren bölgeden ikililer sunmaktır; üçüncüsü ise hem olurlu (feasible) hem de uygun olmayan (infeasible) çözümleri kapsayan ütopik alandan ikililer sunmaktır. Bu çalışmanın temel amacı, karar vericiye farklı bölgelerden ikililer sunmanın Sıralama Destek Vektör Makinesinin öğrenme süreci üzerindeki etkisini değerlendirmek ve öğrenilen bilgilerin veri oluşturma stratejilerinde kullanılmasıdır. Stratejilerin performansını ve etkinliğini değerlendirmek için, test vakası olarak üç kademeli bir tedarik zinciri dağıtım ağı problemi seçilmiştir. Üç farklı tedarik zinciri dağıtım ağı probleminden elde edilen sonuçlar pratik bir değerlendirme sağlamaktadır. Alternatif ikililer sunma stratejileri arasında belirgin farklılıklar gözlemlenmektedir; bazı stratejiler diğerlerinden daha doğru veya tutarlı sonuçlar vermektedir. Bu, tercih öğrenme algoritmalarının etkinliğinde alternatif ikililerin kaynağının önemini vurgulamaktadır. Ayrıca eğitim verilerinin oluşturulmasında öğrenme bilgilerinin kullanılması Ütopya bölgesi stratejisi dışında önemli bir gelişme sağlamıştır.

ACKNOWLEDGEMENTS

First and foremost, I extend my deepest gratitude to my thesis advisor, Asst. Prof. Ezgi Karabulut Türkseven, whose expertise, understanding, and patience, added considerably to my graduate experience. Your guidance helped me in all the time of research and writing of this thesis.

I cannot express enough thanks to my family, particularly my mother, my father and my sister. Your sacrifices, unconditional love, and unwavering belief in my abilities have been the cornerstone of my achievements.

I also owe a special debt of gratitude to my friends whose immense support and insightful critiques were invaluable.

Finally, I would like to thank all those who have directly or indirectly helped me during the development of this thesis.

to my family

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

MCDM	Multi-criteria Decision Making
MOO	Multi-objective Optimization
MOP	Multi-objective Optimization Problems
SVM	Support Vector Machine
SCM	Supply Chain Management
MCDM	Multi-criteria Decision Making
MOO	Multi-objective Optimization
MOP	Multi-objective Optimization Problems
SVM	Support Vector Machine
SCM	Supply Chain Management
LP	Linear Programming
MILP	Mixed Integer Linear Programming
MINLP	Mixed Integer Non-linear Programming
MCDA	Multi-criteria Decision Analysis

LIST OF SYMBOLS

$\succ$	Succeeds
$\sum$	Summation
$\geq$	Greater than or Equal to
$>$	Greater than
$\leq$	Less than or Equal to
$<$	Less than

1. INTRODUCTION

Multi-criteria decision making (MCDM) is about making decisions or selecting the best alternative from a set of available options when multiple criteria need to be considered (Sahoo & Goswami, 2023). Over the years, the application of MCDM has steadily grown. This rise is attributed to the growing understanding of the importance of considering various perspectives and managing numerous trade-offs in complex situations, necessitating the evaluation based on multiple criteria (Cinelli, Kadziński, Gonzalez & Słowiński, 2020). This trend has been especially pronounced in complex decision-making areas such as supply chain management (Govindan, Kadziński & Sivakumar, 2017), energy (Tseng, Ardaniah, Sujanto, Fujii & Lim, 2021), healthcare (Dias, Dias, Ventura, Rocha, Ferreira, Khouri & Lopes, 2022) and so forth.

Multi-objective optimization (MOO) or Pareto optimization is an area of MCDM that is concerned with mathematical optimization problems involving more than one objective function to be optimized simultaneously. Note that MOO and MCDM are distinct concepts. MOO aims to identify a collection of non-dominated solutions. In contrast, MCDM's goal is to order these non-dominated solutions and suggest one of them to decision-makers for potential implementation (Wang, Li, Rangaiah & Wu, 2022).

Recently, there are many optimization techniques for MOO that have been developed by researchers (Trisnaa, Mariminb, Arkemanb & Sunartib, 2016). No-preference, a-priori, a-posteriori and interactive are the four main classifications that utilize optimization algorithms for solving multi-objective optimization problems (MOPs) (Hwang & Masud, 1979). When a decision maker doesn't provide specific preference information, the MOO strategy used is known as a no-preference method (Hwang & Masud, 1979). In the a-priori method, various objectives are aggregated into a single objective, emphasizing the importance of each objective from the decision-makers' perspective. After this merging, single-objective algorithms can be utilized to find the optimal solution without any need for modification (Marler & Arora, 2010). Methods like the weighted sum method, the epsilon-constraint method, and the goal programming approach are examples of a-priori methods. In the a-posteriori category, the structure MOPs is preserved and optimized simultaneously, requiring modifications in the algorithms to solve multiple objectives. Techniques such as mathematical programming, evolutionary algorithms, and deep learning approaches fall under a-posteriori methods (Huy, Nallagownden, Truong, Kannan, Vo & Ho, 2022). For the interactive category, the preferences of decision-makers are assessed and integrated throughout the MOO process. This method involves continual interaction with decision-makers to align the optimization process with their changing preferences (Meignan, Knust, Frayret & Gaud, 2015). In this article, we focused on one of the a-priori method which is weighted sum method that consolidates various objectives into one by attributing weights to each. This leads to a singular objective function derived from the aggregated weighted objectives, reflecting their importance according to the decision maker's preferences (Zelany, 1974).

Selecting suitable weights for the weighting method can pose a significant challenge, even for those knowledgeable in the relevant problem area (Billal & Hossain, 2020). Ripon, Khan, Glette, Hovin & Torresen (2011) pointed out various shortcomings of the weighting method such as the difficulty in pre-determining the right weight for each objective function, the limitation of generating only one Pareto optimal solution in each run and the possibility of different combinations of weights leading to the same Pareto optimal solution. Even for an individual decision-maker, providing numerical values to represent the relative importance of various decision criteria proves to be a difficult task. This complexity is naturally increased when attempting to gather these criteria weights from multiple decision-makers (Odu, 2019). Despite the fact that the appropriate weights to accurately represent the decision maker's preferences is not straightforward, weights can significantly affect the results (Keshavarz-Ghorabaee, Amiri, Zavadskas & Turskis, 2021).

In practical applications, criteria weights are frequently determined subjectively by experts (Zavadskas & Podvezko, 2016). There are numerous methods developed for determining these weights based on expert evaluations of their significance. Among the well-known approaches are the Analytic Hierarchy Process (AHP) (Saaty, 1980), the Delphi method (Hwang & Lin, 1987), the Stepwise Weight Assessment Ratio Analysis (SWARA) (Kersuliene, Zavadskas & Turskis, 2010) and the Factor Relationship (FARE) method.

New approaches have developed, fuzzy-based methods for managing uncertainty, data-centric models that utilize machine learning and big data analytics, as well as integrated hybrid methodologies combining various techniques (Sahoo & Goswami, 2023). Preference learning, a subset of machine learning, concentrates on ranking a set of alternatives into ordered classes, labels, or levels based on multiple attributes (Krzysztof Martyn, 2023). Incorporating preference learning into MOO allows decision-makers to customize the optimization process to better reflect human preferences or certain criteria that might be challenging to assess. For preference learning, various algorithms are employed, including support vector machines, neural networks, and decision trees (Krzysztof Martyn, 2023). These algorithms can discover rules and patterns from a variety of data types and features can handle more complex scenarios and scale better with large amounts of data (Sahoo & Goswami, 2023).

Rank-SVM, a preference learning technique, designed to infer the criteria weight vector that a decision-maker have in mind by learning from pairwise comparisons (Fürnkranz & Hüllermeier, 2011). We are motivated to determine the weight vector based on the observed preference information exemplified by $a \succ b$ as our training data, where a denotes the first alternative and b denotes the second alternative. The goal of this thesis is to study the impact of the selection of alternative pairs for comparison while employing the pairwise ranking approach Rank SVM. Three strategies are developed for the selection of the alternative pairs to decision-makers:

1.1 Offering pairs from the Pareto frontier which is a concept that represents the set of optimal solutions where no other solution dominates.

1.2 Offering pairs from the feasible region, meaning dominated and non-dominated solutions that are possible given the constraints.

1.3 Offering pairs from the utopian space, a theoretical region that encompasses both feasible and infeasible solutions i.e., idealized space where the best possible outcomes for all objectives are achieved simultaneously

Our research aims to evaluate the impact of offering pairs from different regions on the learning process of the Rank-SVM. Specifically, we seek to understand whether this approach affects the accuracy and speed of determining the criterion weight vector that the decision maker has in mind. Then, we analyze the impact of data generation strategies utilizing the information learned.

To evaluate the performance and effectiveness of our strategies, we chose a three-echelon supply chain network problem as our test case. The reason behind that is: supply chain networks are complex systems with numerous factors influencing their efficiency, robustness, and resilience. As industries evolve, so do their supply chains, bringing forth new challenges that necessitate innovative approaches for optimization (Vidrova, 2020). Historically, single-objective optimization methods dominated this field, focusing on singular objectives like cost minimization or lead-time reduction (Trisnaa et al., 2016). However, in the dynamic landscape of today's environment, where sustainability, quality, agility, flexibility, customer satisfaction and various other criteria intersect, a singular goal is often insufficient (Vafaeenezhad, Tavakkoli-Moghaddam & Cheikhrouhou, 2019).

The remainder of this thesis is organized as follows. A review is provided from the relevant literature in Section 2. Following this, Section 3 details the identification of the problem along with its background information. The deterministic three-echelon supply chain network model is introduced in Section 4, including its objective functions and constraints. The solution methodology is introduced in Section 5, and the experimental results are discussed in Section 6. Finally, Section 7 concludes the thesis and outlines potential future research directions.

2. LITERATURE REVIEW

Firstly, we review the related literature on multi-objective optimization in Section 2.1. This is followed by an analysis of studies on preference learning within the context of multi-objective optimization, detailed in Section 2.2. Lastly, Section 2.3 presents an overview of how multi-objective optimization is applied to supply chain problems.

2.1 Multi-objective Optimization

Multi-objective optimization (MOO) has emerged as a critical area in the optimization field, addressing problems with several conflicting objectives. Early foundational work by Pareto established the concept of optimality in a multi-criteria context, leading to what is now known as Pareto optimality (Pareto, 1906). Given:

- A set of decision variables: $x = (x_1, x_2, \ldots, x_n)$
- A set of objective functions: $f_i(x)$ for $i = 1, 2, \ldots, m$
- A set of constraints: $g_j(x) \leq 0$ for $j = 1, 2, \ldots, p$ and $h_k(x) = 0$ for $k = 1, 2, \ldots, q$

The multiobjective optimization problem can be written as:

$$\text{Minimize (or Maximize)} : \begin{bmatrix} f_1(x) \\ f_2(x) \\ \vdots \\ f_m(x) \end{bmatrix}$$

Subject to:

$$g_j(x) \leq 0, \quad \text{for } j = 1, 2, \ldots, p$$
$$h_k(x) = 0, \quad \text{for } k = 1, 2, \ldots, q$$

The key distinction between single and MOO lies in the nature of the optimal solutions (Vafaeenezhad et al., 2019). While single objective optimization leads to a unique optimal solution, multi-objective optimization presents a range of equally applicable alternative solutions, each with its own set of trade-offs. This concept is central to many fields, including economics, engineering, and decision science, where trade-offs between competing objectives must be considered (Marler & Arora, 2004). These solutions are referred to as Pareto optimal (non-dominated) solutions. A Pareto optimal solution is one where no objective can be improved without simultaneously degrading another objective (Pareto, 1906). The Pareto frontier is the collection of all these optimal solutions, representing the best trade-offs available to decision-makers (Deb, Pratap, Agarwal & Meyarivan, 2002). An example of Pareto frontier with two objective functions can be seen in 2.1:

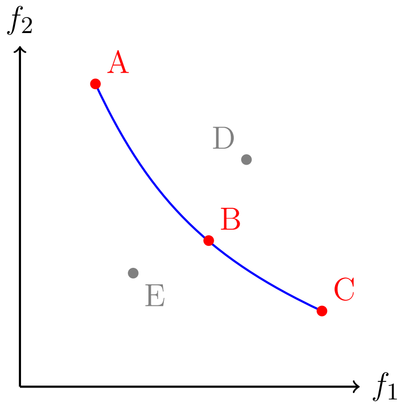

Figure 2.1 Pareto frontier for a minimization problem. Points A, B, and C are on the Pareto frontier (non-dominated solutions), while point D is dominated solution and E is infeasible point.

In the context of Pareto optimization and MOO, utopian space refers to a theoretical or idealized space where the best possible outcomes for all objectives are achieved simultaneously (Zhao, Tang & Yang, 2012). This point is often unreachable in real-world scenarios because in most complicated problems, there is a trade-off between objectives. For example, in a scenario where one objective is to minimize inventory cost and another is to maximize availability of the products, the utopian point would represent the lowest possible inventory cost and the highest possible product availability, a combination that is typically unachievable in reality.

No-preference, a-priori, a-posteriori, and interactive are the four primary categories employing optimization algorithms to address MOPs (Hwang & Masud, 1979). If a decision maker does not specifically provide any preference information, the multi-objective optimization approach employed can be classified as a no-preference method (Hwang & Masud, 1979).

In the a-priori method, various objectives are consolidated into a singular one, highlighting their importance as perceived by decision-makers (Marler & Arora, 2010). Following this aggregation, algorithms designed for single objectives can be applied to determine the optimal solution, without needing to modify the algorithm . In an a-priori context, mathematical programming might be used to formulate and solve a problem where the decision-maker's preferences are stated before the optimization process begins. The a-priori methods include the weighted sum method, where different weights are assigned to each objective, reflecting their relative importance (Zelany, 1974); the epsilon-constraint method, which involves optimizing one objective while setting constraints on the others was first proposed by (Haimes, Lasdon & Wismer, 1971); the goal programming approach, where the aim is to minimize the deviation from target goals for each objective (Trisnaa et al., 2016). These techniques, while straightforward and easy to implement, often require a deep understanding of the problem context to set proper weights or goals. They are particularly useful in scenarios where decision-makers have a clear preference or hierarchy of objectives (Marler & Arora, 2004). This thesis focuses on weighted sum method, which is one of the a priori methods. Weighted sum method can be denoted as follows:

Given m objective functions $f_i(\mathbf{x})$ for $i=1,2,\ldots,m$ and weights w_i for each objective, the weighted sum approach is given in 2.1 (Zelany, 1974):

$$F(\mathbf{x}) = \sum_{i=1}^{m} w_i \cdot f_i(\mathbf{x}) \tag{2.1}$$

where $\mathbf{x}$ is the vector of decision variables and the weights satisfy $w_i \geq 0$ for all i and often $\sum_{i=1}^{m} w_i = 1$.

In practice, choosing the appropriate weights for the weighting method, even by someone well-informed about the problem, can be quite challenging. This difficulty is further exacerbated by the need for scaling among objectives and the fact that minor changes in the weights can lead to significantly distinct solutions (Billal & Hossain, 2020). Ripon et al. (2011) stated several limitations of the weighting method: (1) determining the appropriate weight for each objective function beforehand is challenging; (2) only one Pareto optimal solution is produced in a single run; (3) since all objective functions are summed linearly, this method struggles to identify Pareto optimal solutions not represented in a linear format; (4) different weight combinations might yield the same Pareto optimal solution. Traditional approaches for this include Saaty's Analytical Hierarchy Process (AHP) (Saaty, 1980) and the comparison based approach (Shepetukha & Olson, 2001). Fullér & Majlender (2001) built up a weight vector that aligns with the highest entropy. Nonetheless, the practical use of these methods is constrained since they require additional information which are often challenging to gather in real-world scenarios (Qu, Ma, Clausen & Jørgensen, 2021).

The structure of MOPs is maintained and optimized simulataneously in the a-posteriori category therefore, the algorithms require modifications to address multiple objectives (Huy et al., 2022). This approach enables the acquisition of a set of Pareto optimal solutions in a single simulation. Decision-making follows the optimization process. This highlights the importance of having a diverse range of solutions across all objectives, providing the decision-maker with a wide array of choices. Mathematical programming, evolutionary algorithms and deep learning methods included in the a-posteriori category. Mathematical programming can be part of an a-posteriori approach when it's used to generate a set of Pareto-optimal solutions without pre-defined preferences (Messac & Mattson, 2004). Especially in handling complex problems where objectives are diverse or conflicting, leading to the development of more advanced methods like evolutionary algorithms in the field of MOO (Deb et al., 2002). Over the past thirty years, evolutionary algorithms have emerged as some of the most effective methods for addressing MOO (Zhao et al., 2012). The interest in using evolutionary algorithms has been on the rise since Schaffer (1985)'s pioneering research, which were capable of handling complex and non-convex problems. These evolutionary methods are widely appreciated for their flexibility and robustness, especially in scenarios where objectives conflict or are not well understood (Coello, Brambila, Gamboa, Tapia & Gómez, 2020). Key examples include Genetic Algorithms, which simulate the process of natural evolution using operators like selection, crossover, and mutation to evolve solutions towards optimality; Particle Swarm Optimization, inspired by the social behavior of birds and fish; and Evolutionary Strategies, which emphasize mutation and selection (Qi, Zhang, Ma, Y Quan & Miao, 2017). Another method is the Non-dominated Sorting Genetic Algorithm (NSGA-II), which is specifically designed for MOO and is effective in finding a diverse set of solutions along the Pareto front (Maier, Razavi, Kapelan, Matott, Kasprzyk & Tolson, 2019). However, traditional methods like evolutionary algorithms struggle with the scalability required for contemporary high-dimensional problems (Qu et al., 2021). There are two significant issues that can be: (1) Scalability – the necessity to train an increasing number of models to cover the entire objective space grows exponentially with the number of objectives; and (2) Flexibility – the decision-maker is restricted from freely shifting between preferences unless all the models are pre-trained and stored. Machine learning, an emerging and promising technique in artificial intelligence, is drawing increasing attention for its potential in this context (Qu et al., 2021). Deep learning-based techniques represent innovative methods for producing multiple Pareto optimal solutions. The concept revolves around leveraging the extensive generalization capabilities of deep neural networks to discern and model the entire Pareto front. This is accomplished by learning from a limited set of example trade-offs along that front, a process known as Pareto Front Learning (Navon, Shamsian, Chechik & Fetaya, 2021).

One of the key aspect of MOO is incorporating decision-maker preferences, as purely mathematical solutions may not align with practical or subjective considerations. Research by Cinelli et al. (2020) provided framework for integrating user preferences into MOO solutions. In the interactive category, decision makers' preferences are evaluated and combined through the MOO process. These methods preserve the multi-objective framework but intermittently stop the optimization to collect decision-makers' preferences (Meignan et al., 2015).

2.2 Preference Learning in Multi-objective Optimization

Preference learning is a type of machine learning that focuses on assigning a group of alternatives to preference-ordered classes, labels or ranks in the presence of multiple attributes (Krzysztof Martyn, 2023). These preferences can be provided by users or inferred from their behavior or other types of feedback and can be defined in two ways: a pairwise comparisons (e.g., item A is preferred over item B) and rankings of items (e.g., A is ranked first, B is second) (Fürnkranz & Hüllermeier, 2011). It can be applied in various areas such as (Fürnkranz & Hüllermeier, 2011):

- Recommender systems, where the system learns to recommend items (movies, products, etc.)
- Information retrieval, where the goal is to rank documents by their relevance to a query.
- Decision making and multi-criteria decision analysis, where preferences over multiple criteria need to be aggregated.

Preference learning can be seen as a way to understand or model the preferences that are then used in MOO to make decisions that align with these preferences. By integrating preference learning into MOO, decision-makers can adopt the optimization process to align more closely with human preferences or specific criteria that are not easily quantifiable.

A variety of algorithms are used for preference learning, such as support vector machines, neural networks and decision trees (Krzysztof Martyn, 2023). In this thesis we utilize Rank SVM, a machine learning algorithm, to deduce the weight vector of criteria that a decision-maker has in mind. Rank SVM, short for Ranking Support Vector Machine, is a specialized algorithm in the realm of preference learning that extends the traditional support vector machine framework to handle ranking problems (Krzysztof Martyn, 2023). It can be used in scenarios where the goal is to order or rank a set of alternatives based on preferences. The primary concept behind Rank SVM is to learn from pairwise comparisons – it considers pairs of alternatives and aims to minimize the number of misordered pairs (Fürnkranz & Hüllermeier, 2011). In essence, Rank SVM treats ranking as a binary classification problem; for each pair of alternatives, it predicts which item is preferred over the other. This approach allows it to learn a model that can generalize to rank new items. Rank SVM aims to find a function f that ranks instances such that for any pair of instances (x_i, x_j) with labels $y_i > y_j$, it holds that $f(x_i) > f(x_j)$ (Desmedt, Iliopoulou, Lopez & Grave, 2021).

2.3 Multi-objective Optimization in Supply Chain

The complexity of supply chain management (SCM) has increased with the globalization and technological advancements (Vidrova, 2020). The complexity of SCM is highlighted by the need to take into account the various phases of the supply chain (sourcing, manufacturing, warehousing, distribution, and transportation), various types of supply chains (such as forward, reverse, and closed-loop), multiple levels of decision-making (strategic, tactical, and operational), and the overall supply chain environment (certain, uncertain). (Trisnaa et al., 2016). It's a complicated process that includes a variety of stakeholders, ranging from suppliers to customers, managing products and services while considering variety of objectives simultaneously. This part of the literature review examines the evolution, methodologies and applications of MOO in SCM.

Various researchers have considered several objectives within supply chains (Trisnaa et al., 2016). For instance, Sustainable Supply Chain and Logistics Modeling (SSCLM) is gaining attention in research due to its aim to optimize economic, environmental, and social objectives together (Jayarathna, Agdas, Dawes &

Yigitcanlar, 2021). Table 2.1 presents the objective functions that the researchers, as discussed in this article, have focused on.

Table 2.1 Objectives in Supply Chain Problems

Authors	**Objectives**
Mastrocinque et al. (2013) and Zhao et al. (2012)	Min. total cost and min. delivery lead time
Farahani & Elahipanah (2008)	Min. total cost and max. service level
Amin & Zhang (2013)	Min. total cost and min. environmental impact
Liu & Papageorgiou (2013)	Min. total cost, Min. Process time and min. sale losses
Zhang et al. (2013)	Min. total cost, min. delivery lead time, max. product quality and max. green appraisal score
Ruiz-Femenia et al. (2013)	Max. (net present value) NPV and min. global warning potential (GWP)
Cheshmehgaz et al. (2013)	Min. total cost and min. response time

A-priori methods such as weighted sum, epsilon-constraint, goal programming etc. can be used in MOO in supply chains. Zhang et al. (2013) created a bi-objective model for designing the supply chain of dispersed manufacturing in China, employing the weighted sum method. Liu & Papageorgiou (2013) applied the epsilon-constraint method the lexicographic method for optimizing production, distribution, and capacity planning in multi-product, multi-period global supply chains within the process industry.

Mathematical programming can be utilized to develop and solve MOPs, wherein the preferences of the decision-maker are defined prior to the initiation of the optimization process. Trisnaa et al. (2016) classified supply chain models based on the supply chain environment, between certain and uncertain environments. In scenarios with a certain environment, supply chain models are often constructed using deterministic programming techniques such as linear programming (LP), integer programming (IP), non-linear programming (NLP), mixed integer linear programming (MILP), and mixed integer non-linear programming (MINLP). Besides, in uncertainty supply chain settings, researchers commonly apply methods like fuzzy programming, robust optimization, and stochastic programming (Trisnaa et al., 2016). Jamshidi, Ghomi & Karimi (2012) proposed a model for green supply chain optimization using a MINLP approach. This model was constructed to achieve the dual objectives of minimizing annual cost as well as environmental impact. Amin & Zhang (2013) created a MILP model for optimizing a closed-loop supply chain network. The model's objectives included selecting potential locations for manufacturing and remanufacturing plants, determining the range of products to be produced, identifying demand market locations, and choosing locations for possible collection centers. They utilized a multi-objective function within the linear programming framework to minimize both costs and environmental impacts. Ruiz-Femenia et al. (2013) formulated a Stochastic Mixed-Integer Linear Program (SMILP) to model chemical supply chains. Their analysis focused on the impact of demand uncertainty on the multi-objective optimization of chemical supply chains, assessing both economic and environmental impacts.

Genetic algorithms, ant colony optimization, memetic algorithms, tabu search, and simulated annealing are the metaheuristic methods often used to solve multi-objective optimization for supply chain cases (Trisnaa et al., 2016). Farahani & Elahipanah (2008) applied a genetic algorithm to create and resolve a just-in-time (JIT) distribution model within supply chain management, covering a network of suppliers, wholesalers, and retailers. Their fundamental goal was to maximize the service level by minimizing the total of backorders and excesses of products across all periods. Cheshmehgaz et al. (2013) used the NSGA II algorithm for restructuring supply chain networks, aiming to reduce both the response time to consumers and the overall costs. Their redesigned supply chain configuration included a three-level logistic network, which consist of potential suppliers, distribution centers, and deterministic demand from available customers. Zhao et al. (2012) applied the ant colony optimization technique to enhance supply chain design, adapting to dynamic business environments and diverse customer requirements. Their design aimed to fulfill dual objectives: optimizing both cost and time efficiency in the supply chain.

3. PROBLEM FORMULATION

We are presented with a decision-maker who is well-informed about the nature of the supply chain network problem, objectives and restrictions within the experimental setting we are examining. We know everything except the relative importance of the objectives i.e. weights. Our study aims to reveal the weight vector that reflects the decision maker's preferences by applying the Rank SVM method, which focuses on pairwise ranking. Additionally, we study the impact of offering alternatives from different regions (Pareto front, feasible region, utopian space) on the Rank SVM learning process.

The input for training the Rank SVM model is alternative pairs and information about which alternative is preferred in each pair (Fürnkranz & Hüllermeier, 2011). Decision makers, who are assumed to be rational, are asked to choose between two alternatives. The alternatives are described through feature vectors, which include details on various aspects of the alternative, such as transportation and inventory costs. At every iteration, decision maker is provided with two alternatives (pairs) and she select the alternative with the lowest cost which is recorded as the preference of the decision maker. The alternatives to be offered to each decision-maker are determined by solving an optimization problem which is provided in Section 4.2. Also, we assume that offering fewer alternative pairs to the decision maker will lead to more consistent emergence of preferences, therefore we also seek to find the minimum number of iterations required for an accurate learning outcome.

Instead of using traditional batch learning methods, which typically build a model based on the entire data-set in a single analysis, we used online learning where the model takes data points one by one (Hoi, Sahoo, Lu & Zhao, 2021). At every iteration, alternatives are offered to decision makers and preference information is received, which is used to update the estimated weight vector to be used in subsequent iterations. The goal is to continuously improve the model's prediction accuracy by using information from previous data points and any additional information. This procedure is similar to how decisions are made in the real world, where decisions are frequently made in a sequential fashion and have an impact on one another. We use training data indicative of preference orders, exemplified by $x_1 \succ x_2$, where x_1 denotes the first alternative and x_2 indicates the second alternative.

$$x_1 \succ x_2 \implies w^T x_1 \geq w^T x_2 \tag{3.1}$$

$$w^T (x_1 - x_2) \geq 0 \tag{3.2}$$

$$w^T (x_2 - x_1) \leq 0 \tag{3.3}$$

where w is the weight vector to be learned. Equations (3.1), (3.2), and (3.3) describe how the Rank SVM model interprets preferences between two alternatives based on feature representations x_1 and x_2. Equation (3.1) states that if option x_1 is preferred over x_2 (denoted by $x_1 \succ x_2$), then the score given to x_1 by the weight vector w should be greater than or equal to the score it gives to x_2. Equation (3.2) takes the difference of the feature vectors and asserts that when this difference is projected onto w, the result should be non-negative, which aligns with the preference stated in Equation (3.1). Finally, Equation (3.3) expresses the converse scenario.

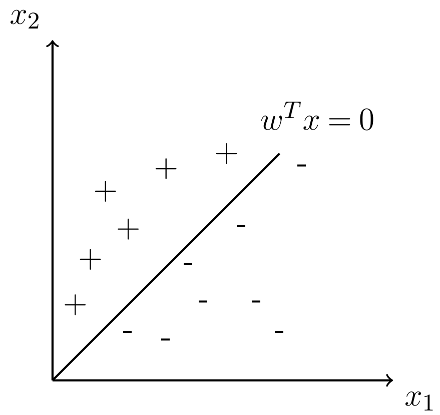

Figure 3.1 Illustration of Linear Rank SVM which is sample plotting to make understand the concepts involved.

Figure 3.1 shows a two-dimensional feature space; where x_1 and x_2 are features. The $w^T x$ line represents the decision boundary learned by the Rank SVM model. The decision boundary is where the model is indifferent between options, meaning it gives them the same score. The points marked with the "+" are the points where the model is preferred, and the points marked with "$-$" are the points where the model is not preferred. Points above the decision boundary line are preferred over points below it because they have higher predicted scores on w (if it is a maximization problem). The goal of the Rank SVM training process is to find the optimal location of this decision boundary so that it best separates higher ranked options from lower ranked options based on the training data provided. As mentioned before, we develop three strategies which are offering pairs from the Pareto front, from the feasible solution set and from the utopian space. One of the main objective is to evaluate the impact of offering pairs from different regions on the decision-making process. This aspect is critical as it explores how varying degrees of information complexity and realism affect the preferences. The detailed analysis of the three strategies can be seen below:

Strategy 1: Pairs from Pareto Frontier

Offering pairs from the Pareto front, shown as the blue line in the Figure 3.2, meaning that only the optimal solutions are presented as alternatives to the decision-makers. This ensures that the decision-maker always chooses between the best possible trade-offs. On the other hand, this strategy might not reveal the decision-maker's preferences in regions where the trade-offs are not very sharp. Moreover, decision-makers may exhibit a bias towards certain objectives because, depending on the nature of the problem, the Pareto front may be dominated by certain types of solutions.

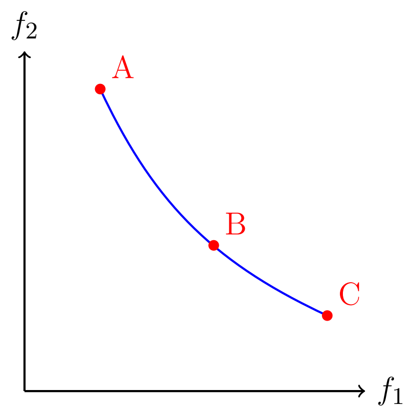

Figure 3.2 The selection region of the alternative pairs for the first strategy

Strategy 2: Pairs from Feasible Region

Proposing pairs from the feasible solution set that includes both non-dominated and dominated solutions can be seen as the shaded area in Figure 3.3. This strategy offers a wider range of options, potentially revealing more diverse preferences. It can better capture the decision maker's preferences across the feasible space, not just at the optimal frontier. However, more options can lead to complexity and difficulty in making choices, possibly leading to decision fatigue or less consistent choices.

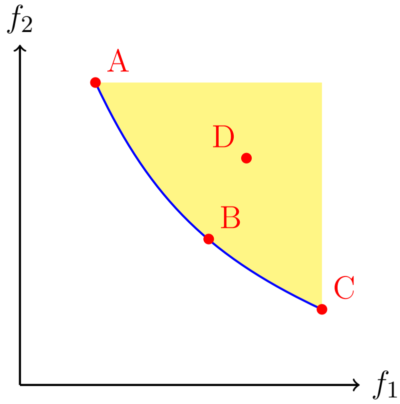

Figure 3.3 The selection region of the alternative pairs for the second strategy

Strategy 3: Pairs from Utopian Region

Offering pairs from the utopian space, which includes both feasible and infeasible solutions shown as the shaded area in 3.4. This can provide insight into the decision maker's ideal preferences, which can reveal underlying values more clearly and encourage the decision maker to think more broadly and creatively. Also, it can be useful to understand the upper bounds of the decision-maker's aspirations or desirable states. However, decision makers may find it difficult to make choices when presented with options that are not based on reality, which can lead to confusion or inconsistency in choices. It can set unattainable expectations, leading to disappointment.

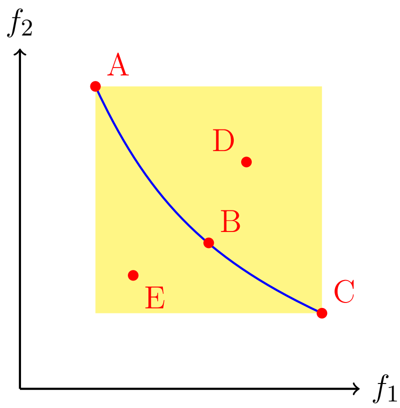

Figure 3.4 The selection region of the alternative pairs for the third strategy

4. APPLICATION IN SUPPLY CHAIN

4.1 Supply Chain Network Problem

Our problem is a deterministic three-echelon supply chain network problem involving suppliers, wholesalers, and retailers which illustrated in Figure 4.1. Consider a multi-product, multi-period, three-echelon supply chain comprised of suppliers who distribute various products to both wholesalers and retailers; wholesalers who obtain these products from suppliers and store them to meet the demands of retailers; and retailers who acquire products from both suppliers and wholesalers. It is characterized by nine objective functions which are:

1. Minimize transportation cost from suppliers to wholesalers
2. Minimize transportation cost from suppliers to retailers
3. Minimize transportation cost from wholesalers to retailers
4. Minimize wholesalers' inventory cost
5. Minimize retailers' inventory cost
6. Minimize retailers' backorder cost
7. Minimize vehicle fixed cost from suppliers to wholesalers
8. Minimize vehicle fixed cost from suppliers to retailers
9. Minimize vehicle fixed cost from wholesalers to retailers

There are trade-offs between these objectives. For example, reducing shipping costs often involves aggregating shipments, less frequent deliveries. This approach can lead to increase in the stock levels at both wholesaler and retailer, thus potentially increasing inventory holding costs. Conversely, to minimize inventory costs, more frequent shipments in smaller quantities might be preferred. This can lead to

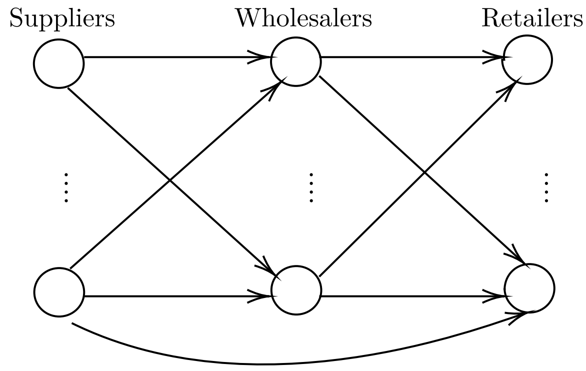

Figure 4.1 Three-Echelon Supply Chain Network

higher transportation costs due to less efficient use of vehicle space and increased trip frequency. Additionally, minimizing backorder costs often requires maintaining higher inventory levels to prevent stockouts. This directly conflicts with the goal of minimizing inventory costs, as higher inventory levels lead to greater holding costs. While all objective functions in this setting are cost-based, it is recognized that this may not always be the case in real-world scenarios. The variety of objectives indicated by the authors can be seen in multi-objective supply chain problems in Section 2, Table 2.1. For example, there could be instances where the authors compare objective functions with different units such as environment impact versus cost (Amin & Zhang, 2013), service level versus cost (Farahani & Elahipanah, 2008).

4.2 Mathematical Model

Our MILP model is constructed with the intent to minimize the cost while adhering to the constraints of the system. Binary decision variables, which determine the vehicle usage, reflecting the discrete decision of whether to use transportation resources between specific nodes in the network. These binary decisions enable the model to substantially reduce unnecessary fixed transportation costs by optimizing the allocation of vehicles only when needed. Furthermore, continuous variables are the product quantities being transported between nodes, the inventory levels to be maintained and backordering volumes. These continuous variables are are crucial in providing a balance between holding sufficient inventory to meet demand and avoiding the financial stress caused by overstocking. Our model was developed by adapting minor changes to the model in the Farahani & Elahipanah (2008)'s article. The parameters and decision variables illustrated in Table 4.1.

Table 4.1 Notation used throughout the Three-echelon Supply Chain Network Model

Sets

I	Set of suppliers
J	Set of wholesalers
K	Set of retailers
T	Set of periods
P	Set of products

Parameters

h_{pj}	Unit inventory holding cost of product $p \in P$ in wholesaler $j \in J$
h'_{kp}	Unit inventory holding cost of product $p \in P$ in retailer $k \in K$
d_{pkt}	Demand quantity of product $p \in P$ of retailer $k \in K$ in period $t \in T$
g_{ij}	Distance between supplier $i \in I$ and wholesaler $j \in J$
e_{ik}	Distance between supplier $i \in I$ and retailer $k \in K$
l_{jk}	Distance between wholesaler $j \in J$ and retailer $k \in K$
S_{pit}	Supply capacity of supplier $i \in I$ for product $p \in P$ in period $t \in T$
Q_{pjt}	Holding capacity of wholesaler $j \in J$ for product $p \in P$ in period $t \in T$
Q'_{pkt}	Holding capacity of retailer $k \in K$ for product $p \in P$ in period $t \in T$
ca_{jt}	Delivery capacity of wholesaler $j \in J$ in period $t \in T$
ca'_{kt}	Delivery capacity of retailer $k \in K$ in period $t \in T$
$backorder_{kp}$	Unit backorder cost of product $p \in P$ of retailer $k \in K$
bl_{pkt}	Max. amount of permitted backorder of product $p \in P$ of retailer $k \in K$ in period $t \in T$
osw_{ij}	Vehicle fixed cost from supplier $i \in I$ to wholesaler $j \in J$
osr_{ik}	Vehicle fixed cost from supplier $i \in I$ to retailer $k \in K$
owr_{jk}	Vehicle fixed cost from wholesaler $j \in J$ to retailer $k \in K$

Decision Variables

y_{pijt}	amount of product $p \in P$ transported from supplier i to wholesaler $j \in J$ in period $t \in T$
u_{pjkt}	amount of product $p \in P$ transported from wholesaler $j \in J$ to retailer $k \in K$ in period $t \in T$
v_{pikt}	amount of product $p \in P$ transported from supplier $i \in I$ to retailer $k \in K$ in period $t \in T$
Inw_{pjt}	inventory level of product $p \in P$ at wholesaler $j \in J$ in period $t \in T$
In_{pkt}	inventory level of product $p \in P$ at retailer $k \in K$ in period $t \in T$
B_{pkt}	amount of product $p \in P$'s backorders of retailer k in period $t \in T$
zsw_{ijt}	$= \begin{cases} 1, & \text{if vehicle is used from supplier } i \in I \text{ to wholesaler } j \in J \text{ in period } t \in T \\ 0, & \text{otherwise} \end{cases}$
zsr_{ikt}	$= \begin{cases} 1, & \text{if vehicle is used from supplier } i \in I \text{ to retailer } k \in K \text{ in period } t \in T \\ 0, & \text{otherwise} \end{cases}$
zwr_{jkt}	$= \begin{cases} 1, & \text{if vehicle is used from wholesaler } j \in J \text{ to retailer } k \in K \text{ in period } t \in T \\ 0, & \text{otherwise} \end{cases}$

The objective functions of the model aim to minimize the total supply chain costs by optimizing transportation, inventory holding, and backordering costs across all echelons and time periods which can be seen from Equation (4.1) to (4.9). To ensure the feasibility and reliability of the supply chain, our model incorporates a series of constraints that regulate the flow of goods through the network as indicated in from Equation (4.10) to (4.23). These constraints are effective in maintaining inventory balance at all echelons and periods, thus preventing scenarios of overstocking or under-stocking. They also ensure that the supply chain operates within capacity limits and adheres to the logistical considerations such as storage space constraints.

$$(4.1)\quad min \sum_{p\in\mathcal{P}}\sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}}\sum_{t\in\mathcal{T}} y_{pijt}g_{ij}$$

$$(4.2)\quad min \sum_{p\in\mathcal{P}}\sum_{i\in\mathcal{I}}\sum_{k\in\mathcal{K}}\sum_{t\in\mathcal{T}} v_{pikt}e_{ik}$$

$$(4.3)\quad min \sum_{p\in\mathcal{P}}\sum_{j\in\mathcal{J}}\sum_{k\in\mathcal{K}}\sum_{t\in\mathcal{T}} u_{pjkt}l_{jk}$$

$$(4.4)\quad min \quad \sum_{p\in\mathcal{P}}\sum_{j\in\mathcal{J}}\sum_{t\in\mathcal{T}} h_{pj}Inw_{pjt}$$

$$(4.5)\quad min \quad \sum_{p\in\mathcal{P}}\sum_{k\in\mathcal{K}}\sum_{t\in\mathcal{T}} h'_{kp}In_{pkt}$$

$$(4.6)\quad min \quad \sum_{p\in\mathcal{P}}\sum_{k\in\mathcal{K}}\sum_{t\in\mathcal{T}} backorder_{kp}B_{pkt}$$

$$(4.7)\quad min \quad \sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}}\sum_{t\in\mathcal{T}} osw_{ij}zsw_{ijt}$$

$$(4.8)\quad min \quad \sum_{i\in\mathcal{I}}\sum_{k\in\mathcal{K}}\sum_{t\in\mathcal{T}} osr_{ik}zsr_{ikt}$$

$$(4.9)\quad min \quad \sum_{j\in\mathcal{J}}\sum_{k\in\mathcal{K}}\sum_{t\in\mathcal{T}} owr_{jk}zwr_{jkt}$$

Subject to

$$(4.10)\quad \sum_{j\in\mathcal{J}} y_{pijt} + \sum_{k\in\mathcal{K}} v_{pikt} \leq S_{pit} \qquad \forall p\in\mathcal{P}, \forall i\in\mathcal{I}, \forall t\in\mathcal{T}$$

$$(4.11)\quad Inw_{pjt} \leq Q_{pjt} \qquad \forall p\in\mathcal{P}, \forall j\in\mathcal{J}, \forall t\in\mathcal{T}$$

$$(4.12)\quad Inw_{pjt} \geq (0.1)Q_{pjt} \qquad \forall p\in\mathcal{P}, \forall j\in\mathcal{J}, \forall t\in\mathcal{T}, t\neq 1$$

$$(4.13)\quad In_{pkt} \leq Q'_{pkt} \qquad \forall p\in\mathcal{P}, \forall k\in\mathcal{K}, \forall t\in\mathcal{T}$$

$$(4.14)\quad \sum_{p\in\mathcal{P}}\sum_{i\in\mathcal{I}} y_{pijt} \leq ca_{jt} \qquad \forall j\in\mathcal{J}, \forall t\in\mathcal{T}$$

$$(4.15)\quad \sum_{p\in\mathcal{P}}\sum_{j\in\mathcal{J}} u_{pjkt} + \sum_{p\in\mathcal{P}}\sum_{i\in\mathcal{I}} v_{pikt} \leq ca'_{kt} \qquad \forall k\in\mathcal{K}, \forall t\in\mathcal{T}$$

$$(4.16)\quad Inw_{pjt} + \sum_{i \in \mathcal{I}} y_{pijt} = \sum_{k \in \mathcal{K}} u_{pjkt} + Inw_{pj(t+1)} \qquad \forall p \in \mathcal{P}, \forall j \in \mathcal{J}, \forall t \in \mathcal{T}$$

$$(4.17)\quad \begin{aligned} & In_{pkt} - B_{pkt} + \sum_{j \in \mathcal{J}} u_{pjkt} \\ & + \sum_{i \in \mathcal{I}} v_{pikt} = In_{pk(t+1)} - B_{pk(t+1)} + d_{pkt} \end{aligned} \qquad \forall p \in \mathcal{P},\ \forall k \in \mathcal{K},\ \forall t \in \mathcal{T}$$

$$(4.18)\quad B_{pkt} \leq bl_{pkt} \qquad \forall p \in \mathcal{P}, \forall k \in \mathcal{K}, \forall t \in \mathcal{T}$$

$$(4.19)\quad Inw_{pj1}, In_{pk1} = 0 \qquad \forall p \in \mathcal{P}, \forall j \in \mathcal{J}, \forall k \in \mathcal{K}$$

$$(4.20)\quad In_{pkT}, B_{pkT} = 0 \qquad \forall p \in \mathcal{P}, \forall k \in \mathcal{K}$$

$$(4.21)\quad y_{pijt}, u_{pikt}, v_{pjkt} \geq 0 \qquad \forall p \in \mathcal{P}, \forall i \in \mathcal{I}, \forall j \in \mathcal{J}, \forall k \in \mathcal{K}, \forall t \in \mathcal{T}$$

$$(4.22)\quad Inw_{pjt}, In_{pkt}, B_{pkt} \geq 0 \qquad \forall p \in \mathcal{P}, \forall j \in \mathcal{J}, \forall k \in \mathcal{K}, \forall t \in \mathcal{T}$$

$$(4.23)\quad zsw_{ijt}, zsr_{ikt}, zwr_{jkt} \in \{0, 1\} \qquad \forall p \in \mathcal{P}, \forall i \in \mathcal{I}, \forall j \in \mathcal{J}, \forall k \in \mathcal{K}, \forall t \in \mathcal{T}$$

Objective functions (4.1), (4.2) and (4.3) minimize the transportation cost. Equation (4.1) minimizes the cost associated with transporting product $p \in P$ from supplier $i \in I$ to wholesaler $j \in J$ over the time period $t \in T$. Equation (4.2) sum minimizes the cost of transporting product $p \in P$ directly from supplier $i \in I$ to retailer $k \in K$. Equation (4.3) minimizes the cost associated with transporting product $p \in P$ from wholesaler $j \in J$ to retailer $k \in K$. Equation (4.4) and (4.5) minimizes the wholesalers' and retailers' inventory holding cost, respectively. Equation (4.4) minimizes the the costs for holding inventory of product $p \in P$ at wholesaler $j \in J$, where h_{pj} is the unit holding cost for product $p \in P$ at wholesaler $j \in J$, and Inw_{pjt} is the inventory level. In the same manner, Equation (4.5) minimizes the costs for holding inventory of product $p \in P$ at retailer $k \in K$, where h'_{kp} is the unit holding cost for product $p \in P$ at retailer $k \in K$, and In_{pkt} is the inventory level. Objective function (4.6) minimizes the backorder cost in retailers in all periods. It aims to minimize the costs associated with backorders for product $p \in P$ at retailer $k \in K$ during time $t \in T$, with $backorder_{kp}$ representing the backorder cost per unit of product $p \in P$, and B_{pkt} the amount of backordered product. Objective function (4.7), (4.8) and (4.9) minimizes the vehicle cost from supplier to wholesaler, from supplier to retailer and from wholesaler to retailer, respectively. Equation (4.7) minimizes the vehicle costs for shipping product $p \in P$ from supplier $i \in I$ to wholesaler $j \in J$ over time $t \in T$, including variable vehicle cost per unit osw_{ij} and a binary variable zsw_{ijt} indicating whether the vehicle is used. Equation (4.8) minimizes the costs of transporting product $p \in P$ directly from supplier $i \in I$ to retailer $k \in K$ over the time period $t \in T$, with osr_{ik} as the vehicle cost per unit and zsr_{ikt} as the binary variable indicating vehicle usage. Equation (4.9) minimizes the vehicle costs of moving product $p \in P$ from wholesaler $j \in J$ to retailer $k \in K$ over time $t \in T$, where owr_{jk} represents the cost per unit and zwr_{jkt} indicates whether the vehicle is used. These objective functions collectively aim to optimize the supply chain by reducing transportation costs, as well as costs associated with holding inventory and managing backorders. The decision variables y_{pijt}, u_{pjkt}, v_{pikt}, In_{pkt}, Inw_{pjt}, B_{pkt} denote the quantity transported, inventory levels and backorder quantity, while the binary variables zsw_{ijt}, zsr_{ikt}, zwr_{jkt}, indicate whether certain transport routes are active. The parameters such as h_{pj}, h'_{kp}, and $backorder_{kp}$ are coefficients that represent the cost per unit for holding wholesalers' inventory, retailers' inventory and backordering respectively.

Constraint (4.10) ensures that the total amounts requested by wholesalers and retailers from each supplier $i \in I$ do not exceed suppliers' supply capacity for product $p \in P$ in period $t \in T$. Constraint (4.11) and (4.13) guarantees that the inventory level does not exceed the holding capacity of the wholesaler $j \in J$ and retailer $k \in K$ for that product $p \in P$ in period $t \in T$, respectively. Constraint (4.12) sets a minimum inventory level at the wholesaler to avoid stockouts. The inventory level of product $p \in P$ at wholesaler $j \in J$ must be at least 10% of the wholesaler's storage capacity for that product in period $t \in T$, ensuring a minimum level of product availability. Constraint (4.14) and (4.15) controls wholesalers' and retailers' delivery capacities in period $t \in T$, respectively. Constraint (4.16) assures that the beginning inventory of the wholesaler $j \in J$ and the amount of products sent by each supplier $i \in I$ to the wholesalers is equal to the ending inventory of the wholesaler $j \in J$ and the amount of products sent by each wholesaler $j \in J$ to retailers. Constraint (4.17) represents the inventory balance equation for retailer $k \in K$ which implies beginning inventory of the retailer $k \in K$ plus the amount of product received from both suppliers and wholesalers, minus the backorders is equal to the ending inventory of retailer $k \in K$ plus the backorder quantity of the next period and demand of the retailer $k \in K$. Constraint (4.18) limits the number of backorders. The backorders of product $p \in P$ at retailer $k \in K$ in period $t \in T$ must not exceed the maximum allowed backorder quantity for that product at that retailer $k \in K$. Constraint (4.19) ensures that the beginning inventory is 0 for wholesaler $j \in J$ and retailer $k \in K$ at the beginning of the time horizon, respectively. Constraint (4.20) set a zero inventory level and zero backorder amount for retailer $k \in K$ at the end of the time horizon, respectively. Constraint (4.21) requires that all transportation decisions result in non-negative product flows and Constraint (4.22) asserts that inventory and backorder levels must also be non-negative. Finally, Constraint (4.23) enforces binary conditions on transportation routes, indicating whether a route is active with a 1 or inactive with a 0, reflecting the discrete nature of transportation decisions.

5. METHODOLOGY

This chapter presents the methodology adopted in this thesis to explore and validate the application of preference learning in MOO within the field of supply chain networks. The multi-criteria decision analysis (MCDA) framework that allows the assessment of various alternatives across multiple criteria is illustrated in 5.1 (Aggarwal, 2015). Three developed strategies which are the Pareto Frontier approach detailed in Section 5.1, the Feasible Region approach in Section 5.2 and the Utopian Region approach in Section 5.3 with corresponding algorithms presented in Algorithm 2, 3 and 4, respectively.

	c_1	c_2	$\cdots$	c_N
x_1	$x_1^{(1)}$	$x_1^{(2)}$	$\cdots$	$x_1^{(N)}$
x_2	$x_2^{(1)}$	$x_2^{(2)}$	$\cdots$	$x_2^{(N)}$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
x_M	$x_M^{(1)}$	$x_M^{(2)}$	$\cdots$	$x_M^{(N)}$

Figure 5.1 Multi-criteria Decision Analysis Model

$\mathcal{C} = \{c_1, c_2, \ldots, c_N\}$ represents objectives (criteria) e.g. minimize transportation cost, minimize inventory cost etc. Each objective c_j, is associated with a weight value, w_j. The weight vector for the objective set $\mathcal{C}$ is represented as:

$$\mathbf{W} = (w_1, w_2, \ldots, w_N) \tag{5.1}$$

where weights satisfy $w_i \geq 0$ for all i and $\sum_{i=1}^{m} w_i = 1$.

A set of alternatives or objective vectors are denoted as $\mathcal{X} = \{x_1, x_2, \ldots, x_M\}$. Each alternative can be represented as a vector:

$$x_i = (x_i^{(1)}, x_i^{(2)}, \ldots, x_i^{(N)}) \in \mathbb{R}^N \tag{5.2}$$

where, $x_i^{(j)}$ is the outcome of an optimization process influenced by weights of each objective. Overall score of each alternative is calculated as:

$$f_i = w_1 x_i^{(1)} + w_2 x_i^{(2)} + \ldots, + w_N x_i^{(N)} \tag{5.3}$$

To achieve our research objectives, we developed a methodology for the generation of training data. The pseudo code represents an algorithm for a learning procedure of Rank SVM can be seen in Algorithm 1.

Algorithm 1 Rank SVM Learning Procedure

1: **for** $i = 1$ to R **do** ▷ Number of replications
2:     **OptParameters()** ▷ Generate optimization parameters
3:     $w^* \leftarrow$ **RandomWeight()** ▷ Initialize w^* randomly
4:     **for** $j = 1$ to T **do** ▷ Number of iterations
5:         $x_1 \leftarrow$ **ObjectiveVector()** ▷ First objective vector (alternative)
6:         $x_2 \leftarrow$ **ObjectiveVector()** ▷ Second objective vector (alternative)
7:         d_{ij}**.append**$[x_1$ **-** $x_2]$ ▷ Add independent variables to learning data
8:         **if** $x_1^\top w^*$ - $x_2^\top w^* > 0$ **then**
9:             **y.append[1]** ▷ Add labels to learning data
10:         **else**
11:             **y.append[0]** ▷ Add labels to learning data
12:         **end if**
13:         $w^{model} \leftarrow$ **RankSVM**(d_{ij}, y) ▷ Call Rank SVM to update w^{model}
14:     **end for**
15: **end for**

The step-by-step explanation of the pseudo-code is as follows:

- **Initialization:** The algorithm starts with a loop that will execute R replications (lines 1-15). In each replication, optimization parameters are set up by calling OptParameters() (line 2) and true weight vector (w^*) is initialized randomly (line 3).
- **Rank SVM Training:** Another loop starts which will perform T iterations (line 4). In each iteration, two objective vectors x_1 and x_2 are generated (lines 5-6). These represent different alternatives that need to be ranked. The only

variation among the algorithms of three strategies lies in the method of generating these two objective vectors which are the outputs of the optimization. The difference between two objective vectors is calculated and appended to a list d_{ij} (line 7) which will be used as input data for the Rank SVM. A label is generated based on the product of the true weight vector (w^*) with the objective vectors. If x_1 multiplied by w^* is greater than x_2 multiplied by w^*, a label of 1 is appended to y; otherwise, a label of 0 is appended (lines 8-12) to y. These labels represent which objective vector is "better" according to the true weights.

- **Model Update:** The Rank SVM is called with the differences d_{ij} and labels y to update the model weights w_{model} (line 13).

Next, each strategy is assessed individually in Section 5.1, 5.2 and 5.3 noting that the only difference lies in the area where pairs are chosen. The pseudo codes of the algorithms corresponding to each strategy can be seen in Algorithm 2, 3 and 4 below.

5.1 Strategy 1: Pairs from Pareto Frontier

First strategy, which shown in Algorithm 2, involves offering decision-makers with alternative pairs from the Pareto frontier, where each alternative represents an optimal solution. For each iteration, two weight vectors are randomly generated (line 5 and line 7) and the optimization problem (4.1) - (4.23) is solved to find the alternatives to be offered to each decision-maker. The first objective vector (x_1) is the vector of optimized objective values of the optimization problem with respect to the first random weight vector (w_1). This function represents an optimization model that uses w_1 as an input to determine an optimal objective vector by using weighted sum method (line 6). In the same manner, the second objective vector (x_2) is the vector of optimized objective values of the optimization problem with respect to the second random weight vector (w_2).

Algorithm 2 Rank SVM Learning Procedure - Pairs from Pareto Frontier

for $i = 1$ to R **do** ▷ Number of replications
    OptParameters() ▷ Generate optimization parameters
    $w^* \leftarrow$ **RandomWeight()** ▷ Initialize w^* randomly
    for $j = 1$ to T **do** ▷ Number of iterations
        $w_1 \leftarrow$ **RandomWeight()** ▷ First random weight
        $x_1 \leftarrow$ **Optimize**(w_1) ▷ First objective vector
        $w_2 \leftarrow$ **RandomWeight()** ▷ Second random weight
        $x_2 \leftarrow$ **Optimize**(w_2) ▷ Second objective vector
    end for
end for

5.2 Strategy 2: Pairs from Feasible Region

This strategy broadens the scope by including pairs from the feasible region of solutions, thereby capturing a wider range of potential preferences and allowing a more comprehensive understanding of decision-making criteria. To find feasible solutions, we reduced our problem to LP by giving binary decision variables as parameters which indicate the vehicle usage (zsw_{ijt}, zsr_{ikt} and zwr_{jkt}). Different methods can also be used to find feasible solutions.

The second strategy, which shown in Algorithm 3, involves offering decision-makers with alternative pairs from both dominated and non-dominated solutions. For each iteration, 1 or 0 is assigned randomly to binary decision variables which are zsw_{ijt}, zsr_{ikt} and zwr_{jkt} (line 5 and line 8). Next, two weight vectors are randomly generated (line 6 and line 9) and the optimization problem is solved to find the alternatives to be offered to each decision-maker. The first objective vector (x_1) is the vector of optimized objective values of the optimization problem with respect to the first random weight vector (w_1) and z_1 (line 7). In the same manner, the second objective vector (x_2) is the vector of optimized objective values of the optimization problem with respect to the second random weight vector (w_2) and z_2 (line 10). The result of the optimization problem may be infeasible (line 7 and line 10), therefore random z_1 or z_2 is generated until the feasible solutions is found.

Algorithm 3 Rank SVM Learning Procedure - Pairs from Feasible Region

```
for i = 1 to R do                              ▷ Number of replications
    OptParameters()                            ▷ Generate optimization parameters
    w* ← RandomWeight()                        ▷ Initialize w* randomly
    for j = 1 to T do                          ▷ Number of iterations
        z1 ← RandomZ(0,1)                      ▷ First Z value
        w1 ← RandomWeight()                    ▷ First random weight
        x1 ← Optimize(z1, w1)                  ▷ First objective vector
        z2 ← RandomZ(0,1)                      ▷ Second Z value
        w2 ← RandomWeight()                    ▷ Second random weight
        x2 ← Optimize(z2, w2)                  ▷ Second objective vector
    end for
end for
```

5.3 Strategy 3: Pairs from Utopian Region

The third strategy introduces pairs from a utopian space that includes both feasible and infeasible solutions. Algorithm 4 is for generating training data from the Utopian region, which is a conceptual region representing ideal solutions that might not be feasible or practical. For each replication, when calculating the minimum value (obj_min) in a minimization problem, we take the weight of the calculated objective as 1 and the rest as 0 (line 4); when calculating the maximum value (obj_max), we take the weight of the calculated objective as -1 and the rest as 0 (line 5). For each replication, the optimization problem (4.1) - (4.23) is solved to find the bounds of the problem. For each iteration, x_1 and x_2 are selected according to uniform random distribution between minimum and maximum objective values (obj_min, obj_max) (line 7 and line 8).

Algorithm 4 Rank SVM Learning Procedure - Pairs from Utopian Region

1: **for** $i = 1$ to R **do** ▷ Number of replications
2: **OptParameters**() ▷ Generate optimization parameters
3: $w^* \leftarrow$ **RandomWeight**() ▷ Initialize w^* randomly
4: $obj\_min \leftarrow$ **Optimize**(w_{min}) ▷ Optimize to get minimum objectives
5: $obj\_max \leftarrow$ **Optimize**(w_{max}) ▷ Optimize to get maximum objectives
6: **for** $j = 1$ to T **do** ▷ Number of iterations
7: $x_1 \leftarrow$ **Random**(obj_min, obj_max) ▷ First objective vector
8: $x_2 \leftarrow$ **Random**(obj_min, obj_max) ▷ Second objective vector
9: **end for**
10: **end for**

5.4 Using Learning Information to Generate Training Data

So far, we do not use what we learned while generating training data and two objective vectors x_1 and x_2 are generated independently from the previous iterations. Now, we create x_1 as the best estimated solution according to the current model weight vector. In this way, one of the alternatives we compare becomes the incumbent best. In order to achieve this objective, we replace line 5 in Algorithm 1, lines 5 and 6 in Algorithm 2, lines 5, 6, 7, and 8 in Algorithm 3 and line 7 in Algorithm 4 with

$$x_1 = Optimize(w^{model}) \tag{5.4}$$

To clarify, x_1 is the objective vector corresponding to decision that decision maker makes based on the current estimated weight as stated in Equation 5.4.

6. EXPERIMENTAL RESULTS

This section outlines the experiments and provides inferences of the comparisons for different strategies as discussed in Section 3 and Section 5. Models and algorithms in the experiments are coded and run in Python 3.9.16 using Spyder along with Gurobi version 10 (ID:2427355). The computations are performed with Intel Core i7- 1255U CPU @1.70 GHz and 16 GB RAM.

The structure of the test problems is explained in Section 6.1. The results of analysis on various strategies are demonstrated according to inferences in Section 6.2.

6.1 Designing Test Problems

A range of replications is created using various mixes of parameter values to replicate different scenarios in real-world cases. Each replication consists of $P = 2$ products, $I = 4$ suppliers, $J = 3$ wholesalers, $K = 6$ retailers and $T = 4$ periods. The values for each parameters are randomly generated within their respective minimum and maximum limits, as outlined in Table 6.1. The ranges of the parameters adopted with minor modifications from Farahani & Elahipanah (2008)'s article.

Table 6.1 Parameters and Parameter Ranges

Parameter	Range	Parameter	Range
Q	[2000 - 3000]	g	[5 - 10]
Q'	[2000 - 3000]	e	[15 - 30]
h	[10 - 20]	l	[10 - 20]
h'	[5 - 10]	d	[80 - 120]
s	[800, 800 + 240*K/I]	bl	[500 - 600]
ca'	[800, 800 + 160*P]	osw	[10000 - 20000]
ca	[1000, 1000 + 160*P*K/J]	osr	[8000 - 15000]
$backorder$	[5 - 10]	owr	[10000 - 20000]

The objective functions (4.1) - (4.9) are categorized to test our strategies in different settings. 7-objective, 4-objective and 3-objective are the settings which are detailed in below. Our strategies are evaluated for each problem setting, which discussed in Section 6.2. Finally, the results obtained with smart training stated in Section 6.3.

Setting 1: 7-Objective

Objective functions are categorized into seven groups as follows:

1. Minimize transportation cost (4.1) + (4.2) + (4.3)
2. Minimize wholesalers' inventory cost (4.4)
3. Minimize retailers' inventory cost (4.5)
4. Minimize retailers' backorder cost (4.6)
5. Minimize vehicle fixed cost from suppliers to wholesalers (4.7)
6. Minimize vehicle fixed cost from suppliers to retailers (4.8)
7. Minimize vehicle fixed cost from wholesalers to retailers (4.9)

Setting 2: 4-Objective

Objective functions are categorized into four groups as transportation, inventory, backorder and vehicle fixed cost as follows:

1. Minimize transportation cost (4.1) + (4.2) + (4.3)
2. Minimize inventory cost of wholesalers and retailers. (4.4) + (4.5)
3. Minimize backorder cost of retailers. (4.6)
4. Minimize vehicle fixed cost of suppliers, wholesaler and retailers. (4.7) + (4.8) + (4.9)

Setting 3: 3-Objective

Objective functions are categorized into three groups as suppliers, wholesaler and retailers cost as follows:

1. Minimize suppliers' cost which indicates the minimization of transportation and vehicle fixed cost of suppliers. (4.1) + (4.2) + (4.7) + (4.8)
2. Minimize wholesalers' cost which indicates the minimization of transportation, vehicle fixed and inventory cost of wholesalers. (4.3) + (4.4) + (4.9)
3. Minimize retailers' cost which indicates the minimization of inventory and backorder cost of retailers. (4.5) + (4.6)

In the weighted sum approach, if a single objective dominates the others, it may create an imbalance by skewing the solutions towards that leading objective. This imbalance may restrict a search of the solution space and the specific needs represented by the less dominant objectives may be neglected. Therefore, we examined the distribution of all objective function values of R = 100 random instances for all settings using box plots to ensure that no single objective dominates the others. The box plots for each setting can be seen in Figures 6.1, 6.2 and 6.3.

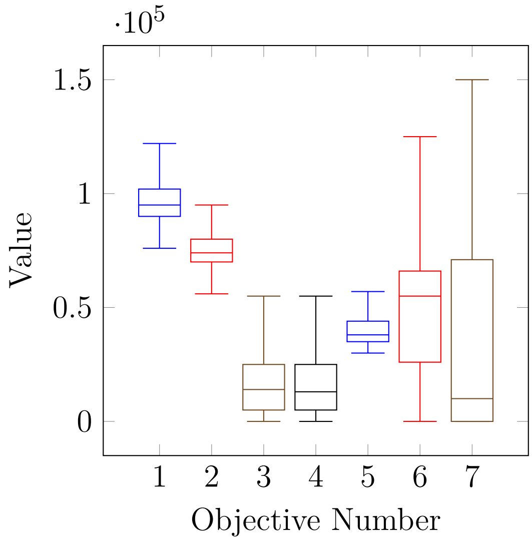

Figure 6.1 Box Plot of 7-objective Setting

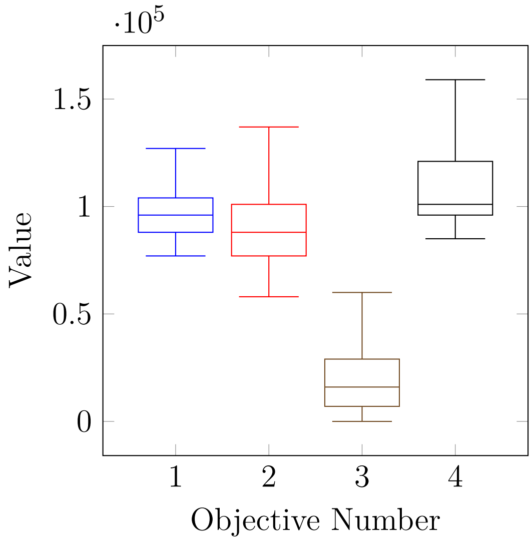

Figure 6.2 Box Plot of 4-objective Setting

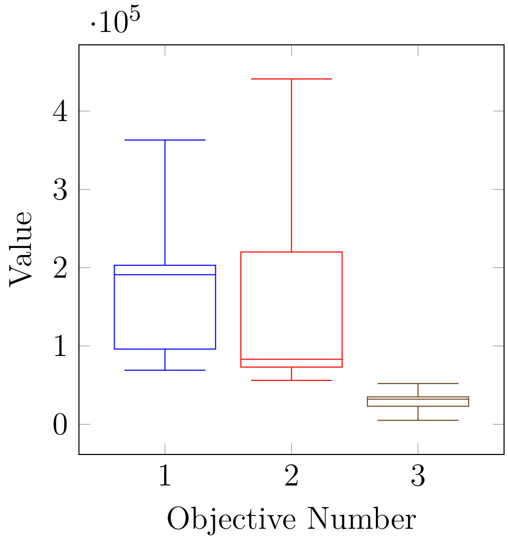

Figure 6.3 Box Plot of 3-objective Setting

The evaluation procedure is illustrated in Algorithm 5. The simulation is repeated for $R = 100$ replications. In each replication, parameters initialized within the predefined ranges in Table 6.1 and weight vectors (w^* and w^{model}) initialized randomly (lines 2, 3 and 6).

Algorithm 5 Evaluation Procedure

```
1:  for i = 1 to R do                                          ▷ Number of replications
2:      OptParameters()                                        ▷ Generate optimization parameters
3:      w* ← RandomWeight()                                    ▷ Initialize w* randomly
4:      true_objective ← Optimize(w*)                          ▷ Optimize to get true objective vector
5:      true_score ← true__objective^⊤ w*
6:      w^model ← RandomWeight()                               ▷ Initialize w^model randomly
7:      for j = 1 to T do                                      ▷ Number of iterations
8:          model_objective ← Optimize(w^model)                ▷ Optimize to get model objective vector
9:          model_score ← model__objective^⊤ w*
10:         x_1 ← ObjectiveVector()                            ▷ First objective vector
11:         x_2 ← ObjectiveVector()                            ▷ Second objective vector
12:         result_star ← Compare(x_1^⊤ w*, x_2^⊤ w*)
13:         result_model ← Compare(x_1^⊤ w^model, x_2^⊤ w^model)
14:         if result_star ≠ result_model then
15:             Accuracy.append[0]
16:         else
17:             Accuracy.append[1]
18:         end if
19:         deviation percentage ← (model score−true score) / (true score)
20:         distance ← linalg.norm(w^star − w^model)
21:         d_ij.append[x_1 - x_2]                             ▷ Add independent variables to learning data
22:         if x_1^⊤ w* - x_2^⊤ w* > 0 then
23:             y.append[1]                                    ▷ Add labels to learning data
24:         else
25:             y.append[0]                                    ▷ Add labels to learning data
26:         end if
27:         w^model ← RankSVM(d_ij, y)                         ▷ Call Rank SVM to update w^model
28:     end for
29: end for
```

The *true_objective* holds the result of the optimization problem (4.1) - (4.23) (line 4). The *true_score* is obtained by calculating the dot product of w^* and *true_objective* (line 5).

Another loop starts which will perform $T = 100$ iterations (line 7). In each iteration, the *model_objective* holds the result of the optimization problem (4.1) - (4.23) (line 8). The *model_score* is obtained by calculating the dot product of w^{model} and *model_objective* (line 9). Also, two alternatives are offered to decision makers and receives preference information from decision makers (lines 10-11).

In addition to the three strategies we previously defined, we also added a random strategy to the comparison. It randomly decides which strategy to use in each iteration. Three evaluation metrics — accuracy, percentage deviation, and Euclidean distance — are employed to assess the strategies, as discussed in Sections 6.2.1, 6.2.2, and 6.2.3. The graphs shown in the following for each metric represent the average of the values obtained in each iteration over $R = 100$ replications.

6.2.1 Accuracy

Accuracy reflects how often the model correctly ranks pairs of two alternatives. The algorithm compares the ranking produced by the true weights (w^*) and the model weights (w^{model}) against the objective vectors x_1 and x_2 (lines 12-13). If the model ranking (*result_model*) matches the true ranking (*result_star*), the accuracy value is recorded as 1; if they differ, the value is 0 (lines 14-18). This binary indicator reflects how well the model's predictions agree with the decision-maker's actual preferences.

To understand the underlying trend easily, 3-point moving average of the accuracy is calculated. All settings (Figures 6.4, 6.5 and 6.6) display a similar trend where the average accuracy for each strategy rises, indicating an enhancement in performance as the iterations progress. The 7-objective setting (Figure 6.4) demonstrates a more evident difference between the strategies early in the iterations. The 4-objective and 3-objective settings (Figures 6.5 and 6.6) display more closely grouped accuracy values for the strategies, implying that the reduction in objectives may lead to less differentiation in accuracy performance. The Random strategy exhibits the higher fluctuation in accuracy across all three settings, generally performing the least consistent performance. The utopian and feasible region strategies consistently lead in performance which may due to expansion of selection region and sharper trade-offs. To summarize, while the performance of the strategies presents variation across different objective settings, the relative performance between the strategies remains stable.

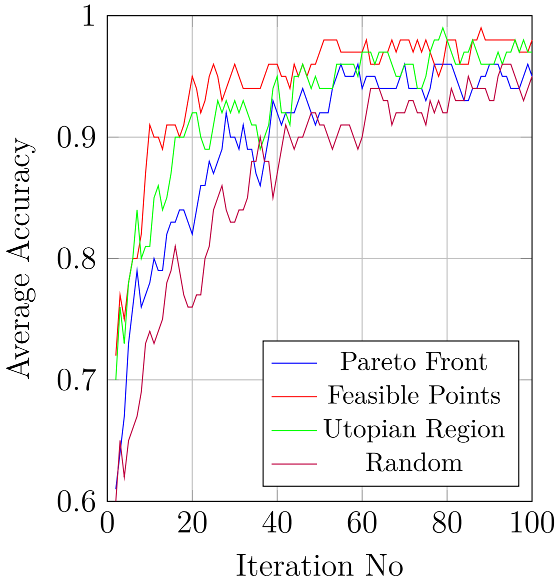

Figure 6.4 Accuracy of 7-objective Setting

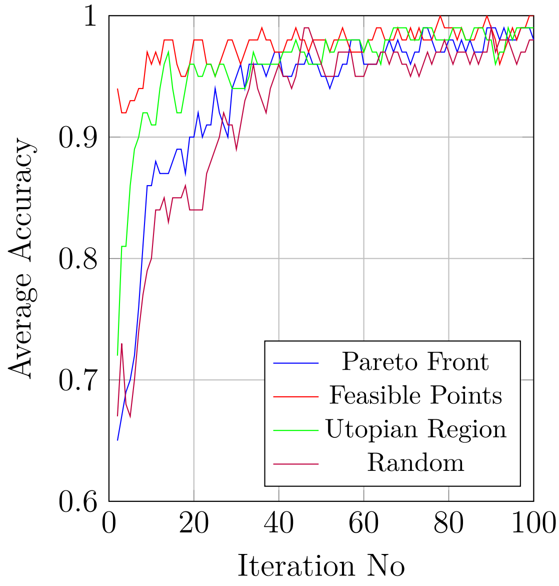

Figure 6.5 Accuracy of 4-objective Setting

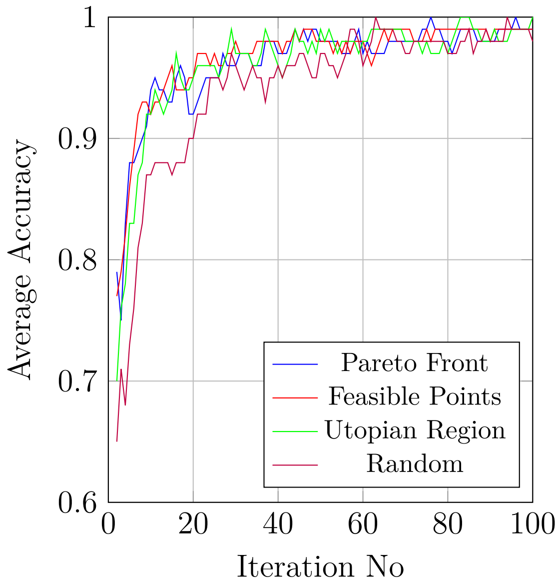

Figure 6.6 Accuracy of 3-objective Setting

6.2.2 Percentage Deviation

The most crucial metric is the deviation percentage because it is the one that directly affects the decision maker's realized cost. Deviation refers to the difference between the costs of decisions made based on the true weights (*true_score*) and the costs of decisions made based on the weights predicted by the Rank SVM model (*model_score*). It is a measure of how much the model score deviates from the true score which is calculated as follows (line 19):

$$true\_score \leftarrow true\_objective^\top w^* \tag{6.1}$$

$$model\_score \leftarrow model\_objective^\top w^* \tag{6.2}$$

$$deviation\ percentage \leftarrow \frac{model\ score - true\ score}{true\ score} \tag{6.3}$$

where *true_objective* holds the result of the optimization problem (4.1) - (4.23) with respect to w^* and *model_objective* holds the result of optimization problem (4.1) - (4.23) with respect to w^{model}.

The closer the model weights are to the decision maker's actual weights, the more likely they make similar decisions therefore the deviation be lower. Conversely, high deviation indicates a significant difference between the model's predictions and the decision maker's actual preferences implying that the model needs to be improved or a more accurate understanding of the decision maker's preferences is needed.

After the initial sharp decline, the strategies converge to stabilize at lower percentages of deviation in all settings (Figures 6.7, 6.8 and 6.9). There is slightly higher initial deviation in the Pareto front strategy in the 3-objective setting (Figure 6.9) than in the 4- and 7-objective setting. Except from this, all three strategies perform similarly in terms of the order of the strategies. The feasible and random strategy is the best performer across all objective settings. Multiple reasons could be attributed to the performance of the these strategies such as more diverse and informative trade-offs. Pareto front is the worst strategy which limits the range of trade-offs and reduces the richness of the information presented.

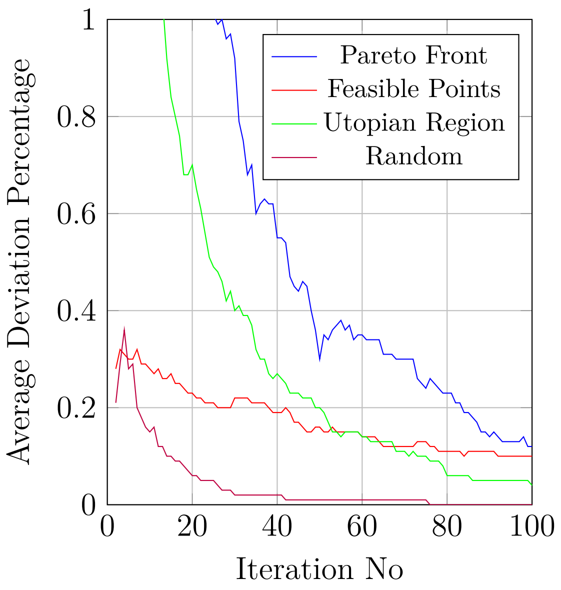

Figure 6.7 Deviation Percentage of 7-objective Setting

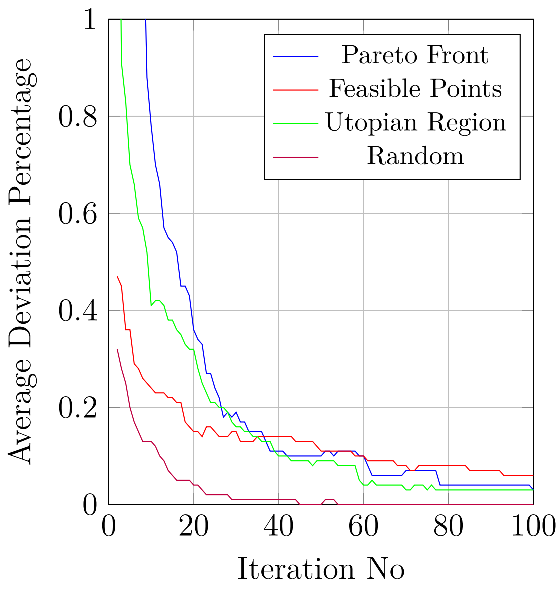

Figure 6.8 Deviation Percentage of 4-objective Setting

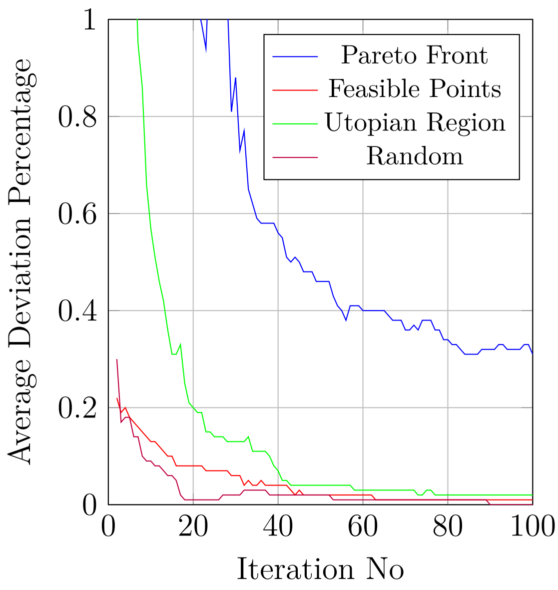

Figure 6.9 Deviation Percentage of 3-objective Setting

6.2.3 Euclidean Distance

Euclidean distance calculates the distance between the true weights (w^*) and the weight vector found by Rank SVM (w^{model}) using a linear algebra norm function (line 20). It is an indicator of the alignment between the weights predicted by the model and the decision-maker's actual preferences which is expressed as:

$$distance \leftarrow linalg.norm(w^{star} - w^{model}) \tag{6.4}$$

In all three settings (Figures 6.10, 6.11 and 6.12), the average Euclidean distance tends to decrease as the number of iterations increases, implying that the strategies are converging towards the desired outcomes over time. It appears that all strategies in the three objective settings reach a stabilization point implying that further iterations beyond this point yield diminishing returns in terms of improvement. As the number of objectives decreases, the distance between true and model weights at the last iteration decreases implying that it may easier to converge the true weights in smaller search spaces. The 7-objective setting (Figure 6.10) and 4-objective setting (Figure 6.11) are very similar to each other in terms of the order of the strategies which is Pareto $\succ$ Utopian $\succ$ Random $\succ$ Feasible. However, in 3-objective setting (Figure 6.12), this order changes to Random $\succ$ Utopian $\succ$ Feasible $\succ$ Pareto. Since multiple weight configurations can yield similar solutions, comparison should be made by assessing other metrics.

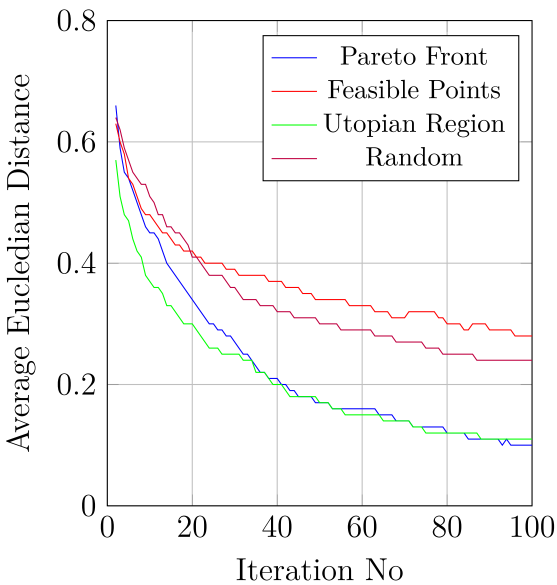

Figure 6.10 Eucledian Distance of 7-objective Setting

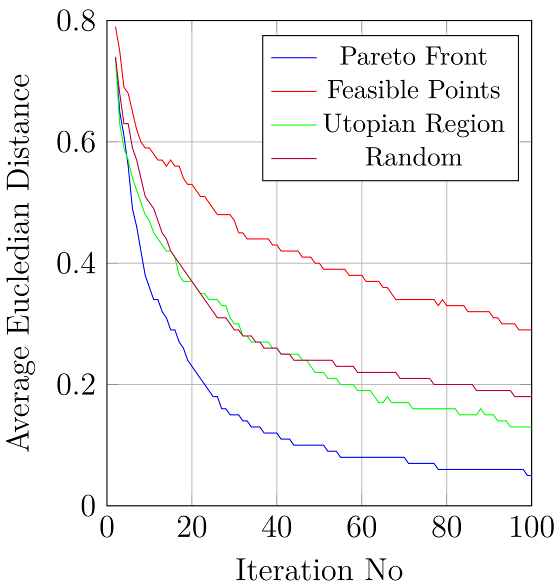

Figure 6.11 Eucledian Distance of 4-objective Setting

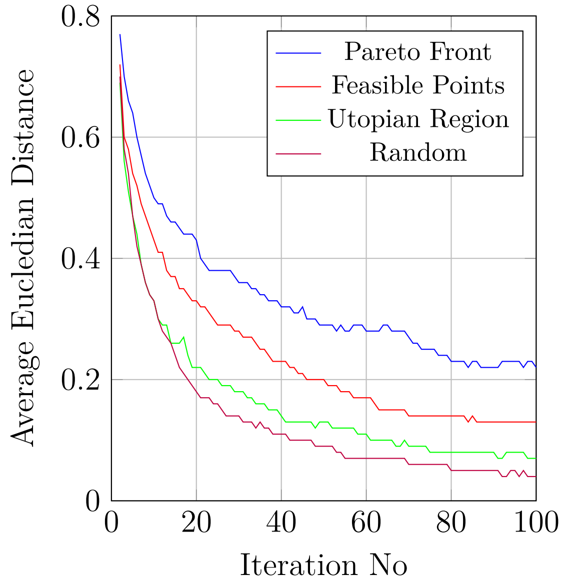

Figure 6.12 Eucledian Distance of 3-objective Setting

6.3 Impact of Smart Training Data

In this section, the results of incorporating learning information in generating training data is examined. According to the evaluation metrics we defined, the comparison of four strategies with and without smart training is as follows:

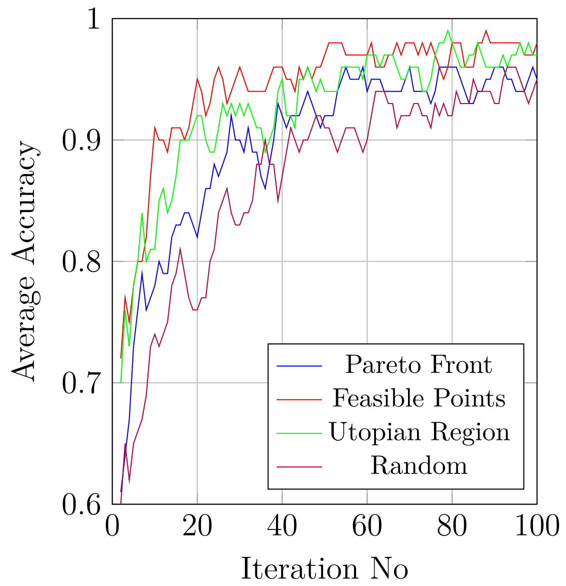

Figure 6.13 Accuracy of 7-objective Setting

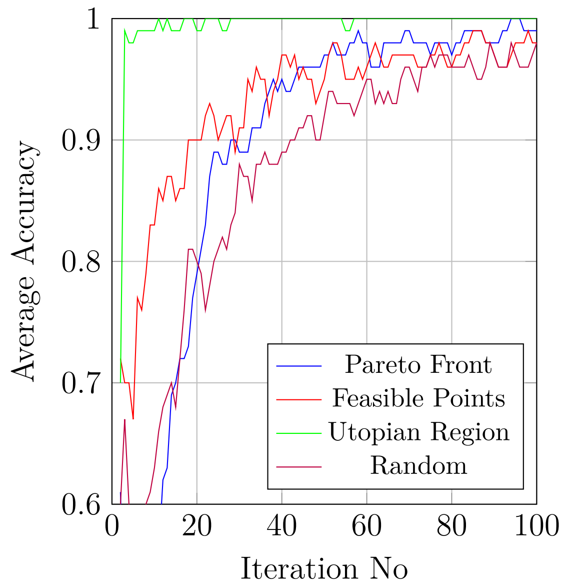

Figure 6.14 Accuracy of Smart 7-objective Setting

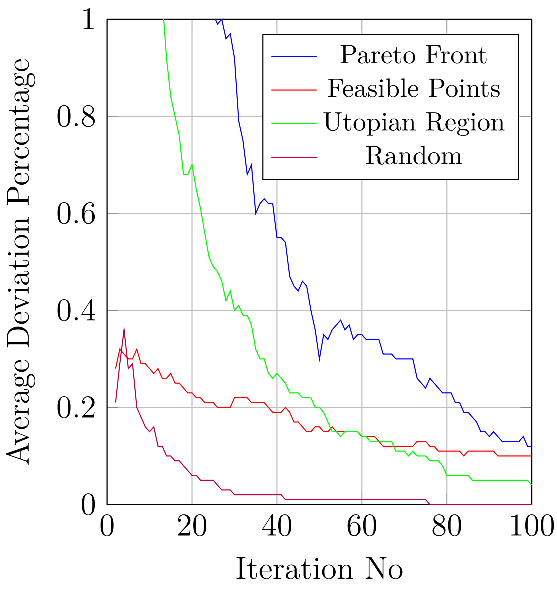

Figure 6.15 Deviation Percentage of 7-objective Setting

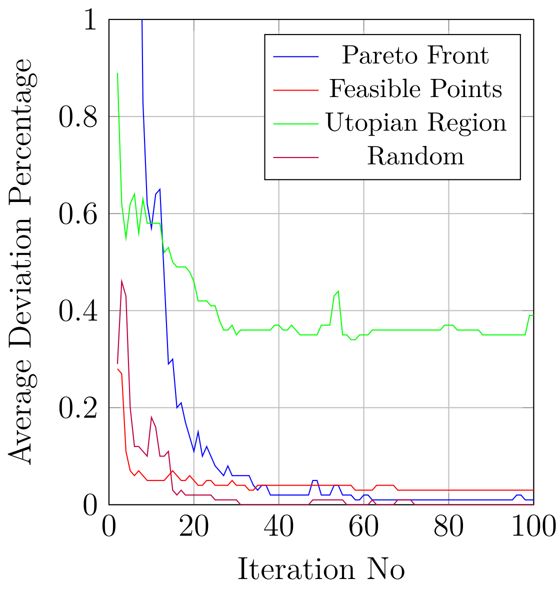

Figure 6.16 Deviation Percentage of Smart 7-objective Setting

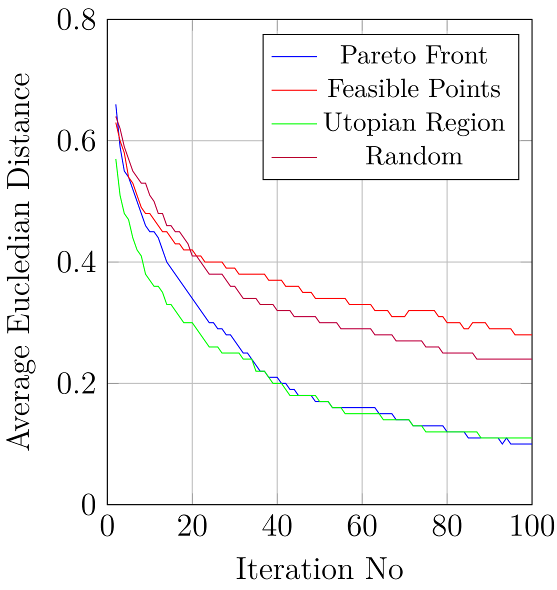

Figure 6.17 Euclidean Distance of 7-objective Setting

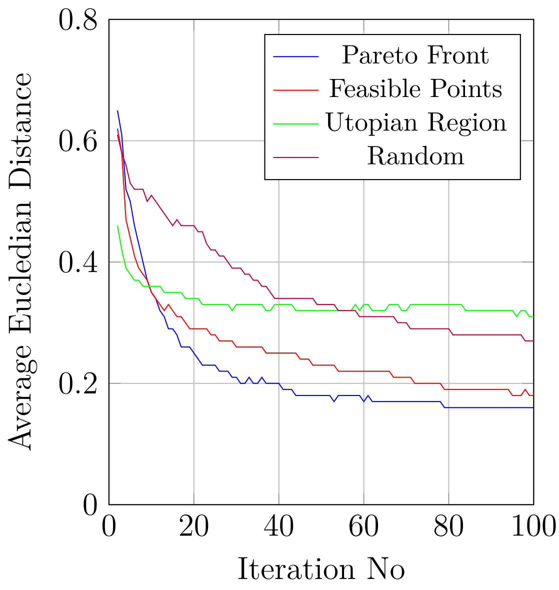

Figure 6.18 Euclidean Distance of Smart 7-objective Setting

Except for the Utopian region strategy, we generally observe that there is an improvement in performance and the order of the strategies does not change with the smart training. As can be seen in Figure 6.19 and Figure 6.20; the Utopian region strategy may get stuck and cannot be able to make any mistakes. If it doesn't make any mistakes, the estimation cannot be updated, therefore the performance of the metrics for the Utopian region strategy is not as expected.

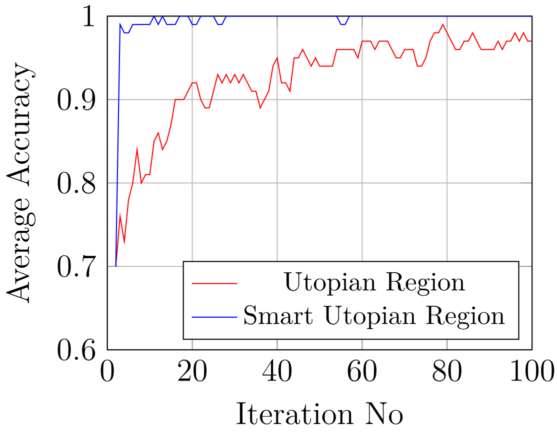

Figure 6.19 Accuracy of 7-objective Setting

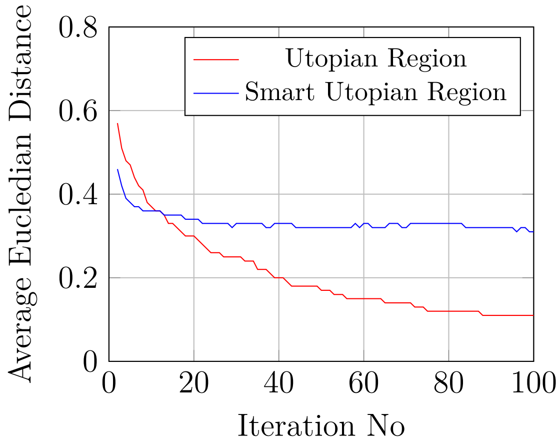

Figure 6.20 Euclidean Distance of 7-objective Setting

These explanations pertain to the 7-objective setting, however, they are applicable to the 4- or 3-objective settings. Table 6.2 and Table 6.3. illustrate in which iteration the percentage deviation fell below the threshold which is determined as 10%, 5% and 1% in 3- and 4-objective setting, respectively. For example, without smart training it drops below 5% in the 32nd iteration, while with smart training it drops in the 5th iteration for feasible region strategy in the 3-objective setting. While the use of smart training improved performance in all strategies, a deterioration is observed in the utopian region strategy. This observation is consistent with the results of the 7-objective setting mentioned earlier.

Table 6.2 Comparison of Percentage Deviation in 4-objective Setting

	10%	5%	1%
Pareto	59	78	-
Pareto (Smart)	**10**	13	28
Feasible	58	-	-
Feasible (Smart)	5	11	60
Utopian	41	59	-
Utopian (Smart)	-	-	-
Random	12	18	31
Random (Smart)	6	10	16

Table 6.3 Comparison of Percentage Deviation in 3-objective Setting

	10%	5%	1%
Pareto	-	-	-
Pareto (Smart)	11	15	29
Feasible	16	32	70
Feasible (Smart)	4	5	22
Utopian	39	43	-
Utopian (Smart)	-	-	-
Random	8	17	89
Random (Smart)	4	6	18

7. CONCLUSION

In this thesis we consider decision-maker who is well-informed about the supply chain network problem within our experimental framework. The one aspect that remains unknown is the relative importance of the objectives and we aim to learn the weights corresponding to each objective by applying the the pairwise ranking approach Rank SVM. The main goal of this thesis is to study the impact of the selection of alternative pairs from different regions. To this end, we developed three strategies for selecting alternative pairs: from the Pareto frontier, the feasible region, and the utopian space. At every iteration, decision-maker is provided with alternative pairs and she select the alternative with the lowest cost. Next, the preference information is used to update the estimated weight vector to be used in the following iterations. Within this framework, we conduct an experimental study on the impact of the selection of alternative pairs on the Rank SVM learning process. Our experimental study assessed these strategies against three metrics: accuracy, percentage deviation, and Euclidean distance. Based on the experiments, we reach the following conclusions.

For accuracy, which indicates whether a decision maker is correctly predicted which alternative she would prefer, the Utopian and feasible region strategy leads in performance. The random strategy indicates the most inconsistent performance, indicating higher variations in accuracy across various settings. Euclidean distance, which is the distance between the model and true weight vectors, the feasible region strategy shows the worst performance. Since multiple weight configurations can yield similar solutions it does not have as direct impact as the percentage deviation metric when evaluating the decision-making process. For the most crucial metric is percentage deviation, which refers to the difference between the costs of decisions made based on the true weights and the weights predicted by the Rank SVM model, all three strategies perform similarly across all settings in terms of the order of the strategies. The feasible region and random strategy shows the best performance in percentage deviation across all objective settings while the Pareto front strategy is the worst.

To summarize, generally the performance of these strategies was consistent across different objective settings, indicating the adaptability of our model. Furthermore, experimental results obtained from applying the strategies to a three-echelon supply chain network problem highlighted the importance of the source of alternative pairs in the effectiveness of preference learning algorithms. Different strategies can be used for different purposes, but considering the ease of calculation and performance according to the percentage deviation, the feasible region strategy is better than others, while the Pareto front strategy is the worst. The feasible region represents both dominated and non-dominated solutions therefore, selecting pairs from this region might provide a wider range of alternatives, exposing the decision-maker to more diverse and informative trade-offs. On the other hand, the Pareto front consists of solutions that are non-dominated, meaning no other solutions are better in all objectives. This might limit the range of trade-offs presented to the decision-maker, reducing the richness of information available for learning preferences. It is also noted that if we include learning information in the training data generation process, there is an increase in performance for all metrics except for the Utopian region strategy. For managerial point of view, it would be more logical to present the alternatives to customers from a wider and realistic area in order to understand their preferences correctly. This type of study of a broad customer base may help an organization shape itself and its production or marketing strategies.

A future direction for this thesis is to extend the model to a stochastic decision-making process. At each iteration, the decision-maker can select the preferred option based on a certain probability which brings additional complexity to the model. Another extension can be the consideration of qualitative objective functions such as sustainability or customer experience in addition to quantitative objective functions. Also, hybrid strategies can be developed. For example, applying the feasible region strategy in the first T iterations and the Pareto front strategy in the remaining iterations. Moreover, the integration of other machine learning techniques such as reinforcement learning or deep learning can be investigated to handle more complex, dynamic supply chain networks with more objectives and decision variables. In future studies, the strategies can be validated through case studies in different industries to understand their practical implications and limitations in the real-world. Additionally, the strategies developed in this thesis can be implemented in a variety of other domains, including healthcare, finance, and urban planning.

BIBLIOGRAPHY

Aggarwal, M. (2015). On learning of weights through preferences. *Information Sciences*, *321*, 90–102.

Amin, S. H. & Zhang, G. (2013). A multi-objective facility location model for closed-loop supply chain network under uncertain demand and return. *Applied Mathematical Modelling*, *37*(6), 41654176.

Billal, M. M. & Hossain, M. M. (2020). Multi-objective optimization for multi-product multi-period four echelon supply chain problems under uncertainty. *Journal of Optimization in Industrial Engineering*, *13*(1), 1–17.

Cheshmehgaz, H. R., Desa, M. I., & Wibowo, A. (2013). A flexible three-level logistic network design considering cost and time criteria with a multi-objective evolutionary algorithm. *Intelligent Manufacturing*, *24*, 277–293.

Cinelli, M., Kadziński, M., Gonzalez, M., & Słowiński, R. (2020). How to support the application of multiple criteria decision analysis? let us start with a comprehensive taxonomy. *Omega*, *96*.

Coello, C. A. C., Brambila, S. G., Gamboa, J. F., Tapia, J. M. G. C., & Gómez, R. H. (2020). Evolutionary multiobjective optimization: open research areas and some challenges lying ahead. *Complex Intelligent Systems*, *6*, 221–236.

Deb, K., Pratap, A., Agarwal, S., & Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE Transactions on Evolutionary Computation*, *6*, 182–197.

Desmedt, N., Iliopoulou, V., Lopez, C., & Grave, K. D. (2021). Active preference learning in product design decisions. *Procedia CIRP*, *100*, 277–282.

Dias, L. C., Dias, J., Ventura, T., Rocha, H., Ferreira, B., Khouri, L., & Lopes, M. D. (2022). Learning target-based preferences through additive models: an application in radiotherapy treatment planning. *European Journal of Operation Research*, *302*(1), 270–279.

Farahani, R. Z. & Elahipanah, M. (2008). A genetic algorithm to optimize the total cost and service level for just-in-time distribution in a supply chain. *International Journal of Production Economics*, *111*, 229–243.

Fullér, R. & Majlender, P. (2001). An analytic approach for obtaining maximal entropy owa operator weights. *Fuzzy Sets and Systems*, *124*(1), 53–57.

Fürnkranz, J. & Hüllermeier, E. (2011). *Preference Learning and Ranking by Pairwise Comparison*. Springer.

Govindan, K., Kadziński, M., & Sivakumar, R. (2017). Application of a novel promethee-based method for construction of a group compromise ranking to prioritization of green suppliers in food supply chain. *Symmetry*, *71*, 129–145.

Haimes, Y., Lasdon, L., & Wismer, D. (1971). On a bicriterionformulation of the problems of integrated systemidentification and system optimization. *IEEE-Transactions on Systems, Man, and Cybernetics*, *1*, 296–297.

Hoi, S. C. H., Sahoo, D., Lu, J., & Zhao, P. (2021). Online learning: A comprehensive survey. *Neurocomputing*, *459*, 249–289.

Huy, T. H. B., Nallagownden, P., Truong, K. H., Kannan, R., Vo, D. N., & Ho, N. (2022). Multi-objective search group algorithm for engineering design problems. *Applied Soft Computing*, *126*.

Hwang, C. L. & Lin, M. J. (1987). *Group Decision Making under Multiple Criteria: Methods and Applications*. Springer–Verlag.

Hwang, C.-L. & Masud, A. S. M. (1979). *Multiple objective decision making, methods and applications: a state-of-the-art survey*. Springer-Verlag.

Jamshidi, R., Ghomi, S. M. T. F., & Karimi, B. (2012). Multi-objective green supply chain optimization with a new hybrid memetic algorithm using the taguchi method. *Scientia Iranica*, *19*(6), 1876–1886.

Jayarathna, C. P., Agdas, D., Dawes, L., & Yigitcanlar, T. (2021). ulti-objective optimization for sustainable supply chain and logistics: A review. *Sustainability*, *13*(24), 13617.

Kersuliene, V., Zavadskas, E. K., & Turskis, Z. (2010). Selection of rational dispute resolution method by applying new step–wise weight assessment ratio analysis (swara). *Journal of Business Economics and Management*, *11*(2), 243–258.

Keshavarz-Ghorabaee, M., Amiri, M., Zavadskas, E. K., & Turskis, Z. (2021). Determination of objective weights using a new method based on the removal effects of criteria (merec). *Symmetry*, *13*, 525.

Krzysztof Martyn, M. K. (2023). Deep preference learning for multiple criteria decision analysis. *European Journal of Operation Research*, *305*(2), 781–805.

Liu, S. & Papageorgiou, L. G. (2013). Multiobjective optimisation of production, distribution and capacity planning of global supply chains in the process industry. *Omega*, *41*, 369–382.

Maier, H. R., Razavi, S., Kapelan, Z., Matott, L. S., Kasprzyk, J., & Tolson, B. A. (2019). Introductory overview: Optimization using evolutionary algorithms and other metaheuristics. *Environmental Modelling Software*, *114*, 195–213.

Marler, R. T. & Arora, J. S. (2004). Survey of multi-objective optimization methods for engineering. *Structural and Multidisciplinary Optimization*, *26*(6), 369–395.

Marler, R. T. & Arora, J. S. (2010). The weighted sum method for multi-objective optimization: new insights. *Structural and Multidisciplinary Optimization*, *41*(6), 853–862.

Mastrocinque, E., Yuce, B., Lambiase, A., & Packianather, M. S. (2013). A multi-objective optimization for supply chain network using the bees algorithm. *International Journal of Engineering Business Management*, *5*(28), 1–11.

Meignan, D., Knust, S., Frayret, J. M., & Gaud, G. P. N. N. (2015). A review and taxonomy of interactive optimization methods in operations research. *ACM Transactions on Interactive Intelligent Systems*, *5*, 17:1–17:43.

Messac, A. & Mattson, C. A. (2004). Normal constraint method with guarantee of even representation of complete pareto frontier. *AIAA Journal*, *42*(10), 2101–2111.

Navon, A., Shamsian, A., Chechik, G., & Fetaya, E. (2021). Learning the pareto front with hypernetworks. In *Proceedings of International Conference on Learning Representations (ICLR)*.

Odu, G. O. (2019). Weighting methods for multi-criteria decision making technique. *Journal of Applied Sciences and Environmental Management*, *23*(8), 1449.

Pareto, V. (1906). Manuale di economia politica. *Societa Editrice*, *13*.

Qi, Y., Zhang, Q., Ma, X., Y Quan, Y., & Miao, Q. (2017). Utopian point based decomposition for multi-objective optimization problems with complicated pareto fronts. *Applied Soft Computing*, *61*, 844–859.

Qu, Q., Ma, Z., Clausen, A., & Jørgensen, B. N. (2021). A comprehensive review of machine learning in multi-objective optimization. In *2021 IEEE 4th International Conference on Big Data and Artificial Intelligence (BDAI)*, (pp. 7–14)., Qingdao, China.

Ripon, K. S. N., Khan, K. N., Glette, K., Hovin, M., & Torresen, J. (2011). Using pareto optimality for solving multi-objective unequal area facility layout problem. In *Proceedings of the 13th Annual Genetic and Evolutionary Computation Conference*, (pp. 12–16)., Dublin, Ireland.

Ruiz-Femenia, R., Guillen-Gosalbez, G., Jimenez, L., & Caballero, J. A. (2013). Multi-objective optimization of environmentally conscious chemical supply chains under demand uncertainty. *Chemical Engineering Science*, *95*, 1–11.

Saaty, T. L. (1980). *The Analytic Hierarchy Process.* McGraw-Hill.

Sahoo, S. K. & Goswami, S. S. (2023). A comprehensive review of multiple criteria decision-making (mcdm) methods: Advancements, applications, and future directions. *Decision Making Advances*, *1*(1), 25–48.

Schaffer, J. D. (1985). Multiple objective optimization with vector evaluated genetic algorithms. In *Proceedings of the 1st International Conference on Genetic Algorithms*, (pp. 93–100)., Pittsburgh, PA, USA.

Shepetukha, Y. & Olson, D. L. (2001). Comparative analysis of multiattribute techniques based on cardinal and ordinal inputs. *Math. Comput. Modell.*, *34*, 229–241.

Trisnaa, T., Mariminb, M., Arkemanb, Y., & Sunartib, T. C. (2016). Multi-objective optimization for supply chain management problem: A literature review. *Decision Science Letters*, *5*, 283–316.

Tseng, M. L., Ardaniah, V., Sujanto, R. Y., Fujii, M., & Lim, M. K. (2021). Multicriteria assessment of renewable energy sources under uncertainty: barriers to adoption. *Technological Forecasting and Social Change*, *171*, 525.

Vafaeenezhad, T., Tavakkoli-Moghaddam, R., & Cheikhrouhou, N. (2019). Multi-objective mathematical modeling for sustainable supply chain management in the paper industry. *Computers and Industrial Engineering*, *135*, 1092–1102.

Vidrova, Z. (2020). Supply chain management in the aspect of globalization. In *SHS Web of Conferences*, volume 74, (pp. 04031).

Wang, Z., Li, J., Rangaiah, G. P., & Wu, Z. (2022). Machine learning aided multi-objective optimization and multi-criteria decision making: Framework and two applications in chemical engineering. *Computers and Chemical Engineering*, *165*, 107945.

Zavadskas, E. K. & Podvezko, V. (2016). Integrated determination of objective criteria weights in mcdm. *International Journal of Information Technology Decision Making*, *15*(02), 267–283.

Zelany, M. (1974). A concept of compromise solutions and the method of the displaced ideal. *Computers Operations Research*, *1*(3-4), 479–496.

Zhang, A., Luo, H., & Huang, G. Q. (2013). A bi-objective model for supply chain design of dispersed manufacturing in china. *International Journal of Production Economics*, *146*(1), 48–58.

Zhao, F., Tang, J., & Yang, Y. (2012). A new approach based on ant colony optimization (aco) to determine the supply chain (sc) design for a product mix. *Journal of Computers*, *7*(3), 736–743.