

Multi-Objective Optimization Problems with Well-Conditioned Solutions

*A thesis submitted in partial fulfillment of
the requirements for the degree of*
Integrated Bachelors and Masters of Science
in
Mathematics and Computing
by
Harshal Bharat Dupare
18MA20015

Under the guidance of
Dr. Geetanjali Panda



Mathematics and Computing
Indian Institute of Technology Kharagpur
West Bengal, India
May, 2023

Self Declaration

*I hereby declare that the research work presented in this project titled “**Multi-Objective Optimization Problems with Well-Conditioned Solutions**” is a bonafide work of **Harshal Bharat Dupare, Roll no. 18MA20015**, carried out in the Department of Mathematics, Indian Institute of Technology Kharagpur under the supervision of **Dr. Geetanjali Panda** and that it has not been submitted elsewhere for a degree.*

Harshal Bharat Dupare, 18MA20015

Department of Mathematics
Indian Institute of Technology, Kharagpur
West Bengal

Dr. Geetanjali Panda

Department of Mathematics
Indian Institute of Technology, Kharagpur
West Bengal

Acknowledgements

For their extraordinary guidance and assistance, without which this research would not have been feasible, I would like to extend my appreciation to my guide and faculty supervisor, **Dr. Geetanjali Panda**. She has always encouraged me to do as much research as I can, read as many papers as I can, and test out as many concepts as I can. Whatever challenges I had while working on the project, she was always there to support me.

Finally, I want to thank the Department of Mathematics, IIT Kharagpur for giving me the chance to complete the work I have done and my family and friends for their emotional support throughout this pandemic.

Harshal Bharat Dupare

Abstract

This thesis addresses the problem of finding well-conditioned solutions to the Linear Factor Model for time series, satisfying $AA^\top = I$. It introduces the problem of Functional Multi-Objective Optimization Problem (fMOOP), in which functions are sought as solutions of Multi-Objective Optimization Problem. The thesis provides motivation for studying such problems and why they are important. It surveys different methodologies for solving MOOP in general and summarizes a few results on the condition number of matrices and functions. The thesis addresses important questions regarding such problems and analyzes different methods for finding well-conditioned solutions to fMOOP. Based on the analysis, the thesis selects a few methods and compares their experimental results to solve the problem of finding well-conditioned solutions to the Linear Factor Model for time series. Using the property of the space of matrices satisfying $AA^\top = I$, a set of extensions of methods are proposed and compared. The methods are implemented and the results show that the method which utilizes the property performs better compared to other methods. Hence, the thesis demonstrates that it is possible to design good extensions of known methods to solve the problem. This work highlights the importance of studying fMOOP and provides valuable insights into solving these problems.

Contents

Abstract	iii
1 Introduction	1
1.1 Preliminaries	2
1.2 Problem Definition	3
1.3 Motivation and Applications	4
1.3.1 Linear Factor Model	4
1.3.2 Defense Against Adversarial Attacks on Neural Networks	5
2 Related Works	7
2.1 Multi-Objective Optimization	7
2.1.1 Definitions	7
2.1.2 No Preference Methods	9
2.1.3 A Priori Methods	9
2.1.4 A Posteriori Methods	10
2.1.4.1 Evolutionary Algorithms (EA)	10
2.2 Condition Number	11
2.2.1 Condition Number of Matrices	11
2.2.2 Condition Number of Functions	12
3 Methodology and Results	13
3.1 Theoretical Results	14
3.1.1 Condition Number over Composition of Function	14
3.1.2 L_2 Regularization & Absolute Condition Number	14
3.2 Experimental Results	15
3.2.1 Methodologies	15
3.2.1.1 Baseline Method	15
3.2.1.2 Gradient Based Methods	16

CONTENTS

3.2.1.3	No Preference Method	23
3.2.1.4	Evolutionary Algorithms	24
3.2.1.4.1	NSGA-II	25
3.2.1.4.2	Multi-Objective Particle Swarm Optimization	27
3.2.2	Data	30
3.2.3	Results	31
3.2.3.1	Baseline Method	31
3.2.3.2	OPI Method : Type 1	31
3.2.3.3	OPI Method : Type 2	32
3.2.3.4	DOI Method : Type 1	32
3.2.3.5	DOI Method : Type 2	33
3.2.3.6	No Preference Method (NPM) : Type 1	34
3.2.3.7	No Preference Method (NPM) : Type 2	34
3.2.3.8	NSGA-II	35
3.2.3.9	MOPSO	35
3.2.3.10	Comparison Table	36
3.2.3.11	Discussion	36
4	Conclusion and Future Work	37
4.0.0.0.1	37

Chapter 1

Introduction

Optimization problems are of great interest academically along with being core to the functioning of today's society by having application in a variety of fields such as social studies, economics, finance, agriculture, automotive, engineering, computer science, networking, and many others.

In general, there can be more than one objective and constraint in the formulation of optimization problems. Optimization problems are classified into 2 broad categories namely *Single Objective Optimization Problems*(SOOP) and *Multi-Objective Optimization Problems* (MOOP), the latter can be considered a superset of the former. Our focus will be on a special subset of MOOP where the *Unknown/Decision Variable* is a *function* let's call them fMOOP. fMOOP are of great interest because of their vast applications in modeling the relation between 2 sets, e.g. all of the Neural Networks so heavily researched today fall into this category with their myriads of applications.

Generally when we have a function, we want it to have some good properties like being correct, easy to compute, easy to store, behave nicely over the input set, or any other desirable properties based on the context. One such very important property for most of such functions of interest is being *Well-Conditioned*, meaning small changes in the input should not result in too big changes in the output. We focus on this problem namely the problem to find function solutions of fMOOP which are well-conditioned lets call this class of problem as cond-fMOOP.

Goal of this study is to analyze one such problem of interest belonging to cond-fMOOP, and study how the existing methodologies perform on such problems and device new methodologies for finding solution to this problem belonging to cond-fMOOP.

1. INTRODUCTION

1.1 Preliminaries

Aggregation Scheme: Given a value function $value(\cdot)$ over x , for $x \in X$, a method to aggregate those values to one value. Denoted as follows:

$$\underset{\forall x \in X}{\text{Agg value}}(x) \quad (1.1)$$

e.g. expectation over given probability distribution $p(\cdot)$ over X .

$$\underset{\forall x \in X}{\text{Agg value}}(x) = \int_{\forall x \in X} value(x)p(x) dx \quad (1.2)$$

e.g. exponential decaying average, given a index-function $t : X \rightarrow [|X|]$, where $[n] = \{0, 1, 2, \dots, n - 1\}$.

$$\underset{\forall x \in X}{\text{Agg value}}(x) = \sum_{\forall x \in X} \lambda^{t(x)} value(x) \quad (1.3)$$

Absolute Condition Number: *Absolute Condition Number* is defined for a function $f : X \rightarrow Y$, where X and Y have a norm $\|\cdot\|$ defined over them. Denoted as follows:

$$cond_{abs}(f, x) = \lim_{\epsilon \rightarrow 0} \sup_{\|\delta x\| \leq \epsilon} \frac{\|\delta f(x)\|}{\|\delta x\|} \quad (1.4)$$

Relative Condition Number: *Relative Condition Number* is defined for a function $f : X \rightarrow Y$, where X and Y have a norm $\|\cdot\|$ defined over them. Denoted as follows:

$$cond_{rel}(f, x) = \lim_{\epsilon \rightarrow 0} \sup_{\|\delta x\| \leq \epsilon} \frac{\|\delta f(x)\| / \|f(x)\|}{\|\delta x\| / \|x\|} \quad (1.5)$$

Induced Norm: Given a Matrix Transform $T^{m \times k} : \mathbb{R}^{n \times m} \rightarrow \mathbb{R}^{n \times k}$ and a norm $\|\cdot\|$ defined over $\mathbb{R}^{n \times m}$ and $\mathbb{R}^{n \times k}$ we define norm of the matrix transform as

$$\|T^{m \times k}\| = \sup_{X \in \mathbb{R}^{n \times m} / \{0\}} \frac{\|X \times T^{m \times k}\|}{\|X\|} \quad (1.6)$$

Note: Whenever the dimensions of the matrix in the context are already defined or are obvious from the context we will omit the superscript denoting its dimension.

1.2 Problem Definition

Given:

Domain $X_1 \cup X_2 \subseteq X$ and their associated Ranges $Y_1 \cup Y_2 \subseteq Y$, where $X_i \equiv Y_i$ such that for each $x \in X_i$ we have a corresponding $y \in Y_i$ i.e. $y \equiv x$.

Aggregation schemes $\underset{\forall x \in X_1}{Agg_1}$ defined over $\mathcal{L} : Y \times Y \rightarrow \mathbb{R}$ call $\mathcal{L}(\cdot, \cdot)$ the *Loss* function and $\underset{\forall x \in X_2}{Agg_2}$ defined over $\mathcal{R} : Y \times Y \rightarrow \mathbb{R}$ call $\mathcal{R}(\cdot, \cdot)$ the *Reward* function.

A scalar $\alpha \in \mathbb{R}^+$ and subsets $X_i \subseteq X$, $i \in \{3, 4, 5, 6\}$.

For a given \mathcal{F} a family of functions $f : X \rightarrow Y$ we need to find a function $f \in \mathcal{F}$ which solves the following

Conditioned Functional Multi-Objective Optimization Problem:

$$\min_{f \in \mathcal{F}} \underset{\forall x \in X_1, y \in Y_1, y \equiv x}{Agg_1} \mathcal{L}(y, f(x)) \quad (1.7)$$

$$\max_{f \in \mathcal{F}} \underset{\forall x \in X_2, y \in Y_2, y \equiv x}{Agg_2} \mathcal{R}(y, f(x)) \quad (1.8)$$

including other problems specific objectives, but it must have atleast one of the 1.9, 1.10 as objective

$$\min_{f \in \mathcal{F}} \max_{x \in X_3} cond_{abs}(f, x) \quad (1.9)$$

$$\min_{f \in \mathcal{F}} \max_{x \in X_4} cond_{rel}(f, x) \quad (1.10)$$

or is subject to at least one of the 1.11, 1.12 as constraints

$$\max_{x \in X_5} cond_{abs}(f, x) \leq \alpha \quad (1.11)$$

$$\max_{x \in X_6} cond_{rel}(f, x) \leq \alpha \quad (1.12)$$

It is very important to mention that the above problem formulation highly depends on the nature of the mapping f and the family \mathcal{F} that it belongs to. For example if f models a function for which we expect smooth change in its value w.r.t. its domain then its a reasonable formulation, and if there are no reasons to believe that it should be smooth over its domain then its not a reasonable formulation. Similarly for a chosen family \mathcal{F} to model f , if \mathcal{F} doesn't have smooth functions belonging to it then its not a reasonable formulations irrespective of the nature of the function that f is modeling.

Note: What we mean by minimizing over a functions $f \in \mathcal{F}$ of a family of functions \mathcal{F} is that we minimize over the parameters of that family of function \mathcal{F} .

1. INTRODUCTION

1.3 Motivation and Applications

1.3.1 Linear Factor Model

The problems defined in the former section was motivated from the problem on linear factor models with added constraint for the returns-to-factor matrix be orthonormal columns. We describe the problem below.

A data of returns for n time series $R_t^{n \times 1} \in \mathbb{R}^{n \times 1}$, for $t \in [T]$ and denote $R_{t \times d}^{n \times d} = [R_{t-1}^{n \times 1}, R_{t-2}^{n \times 1}, \dots, R_{t-d}^{n \times 1}] \in \mathbb{R}^{n \times d}$ is given.

We need to design $F_t^{n \times k} \in \mathbb{R}^{n \times k}$ as a function of $R_{t \times d}^{n \times d}$, i.e. $F_t^{n \times k} = f(R_{t \times d}^{n \times d})$ and from that we can derive $P_t^{n \times 1} = g(F_t^{n \times k})$ such that $P_t^{n \times 1} \approx R_t^{n \times 1}$ which we can measure by *Average* aggregation scheme over the loss function $\mathcal{L}(y, \hat{y}) = \|y - \hat{y}\|_2$ which we need to minimize and another *Average* aggregation scheme over the return function $\mathcal{R}(y, \hat{y}) = \frac{\langle y, \hat{y} \rangle}{\|y\|_2 \|\hat{y}\|_2}$ defined over the testing period of $t \in [T, T + S] = \{T, T + 1, \dots, T + S - 1\}$ which we need to maximize.

$$\begin{array}{ccc}
 R_{t \times d}^{n \times d} & \xrightarrow{f} & F_t^{n \times k} \\
 & \downarrow g & \\
 R_t^{n \times 1} & \xlongequal{\mathcal{L}} & P_t^{n \times 1} \\
 & \searrow \mathcal{R} & \swarrow \\
 & \mathcal{R} &
 \end{array} \tag{1.13}$$

lets assume that the functions $f(\cdot)$ and $g(\cdot)$ are linear w.r.t. their argument. In this case it can be written as

$$F_t^{n \times k} = f(R_{t \times d}^{n \times d}) = R_{t \times d}^{n \times d} \times A^{d \times k} \tag{1.14}$$

$$P_t^{n \times 1} = g(F_t^{n \times k}) = F_t^{n \times k} \times \beta^{k \times 1} \tag{1.15}$$

and the orthonormal condition requires

$$A^{d \times k} (A^{d \times k})^\top = I^{d \times d} \tag{1.16}$$

From the 1.16 we can infer that $(A^{d \times k})^T$ belongs to the set of right inverses of $A^{d \times k}$.

1.3 Motivation and Applications

Further if we relax the 1.16 to bounds on condition number by $\alpha \geq 1$. Here we note that $d \leq k$ for 1.16 to have a solution.

$$\max_{R_{t \times d}^{n \times d} \in \mathbb{R}^{n \times d}} \text{cond}_{\text{rel}}(f, R_{t \times d}^{n \times d}) = \lim_{\epsilon \rightarrow 0} \sup_{\|\delta R_{t \times d}^{n \times d}\| \leq \epsilon} \frac{\|\delta R_{t \times d}^{n \times d} \times A^{d \times k}\|}{\|\delta R_{t \times d}^{n \times d}\|} \frac{\|F_t^{n \times k} \times (A^{d \times k})^\top\|}{\|F_t^{n \times k}\|} \leq \alpha \quad (1.17)$$

which implies

$$\max_{R_{t \times d}^{n \times d} \in \mathbb{R}^{n \times d}} \text{cond}_{\text{rel}}(f, R_{t \times d}^{n \times d}) \leq \|A^{d \times k}\| \|(A^{d \times k})^\top\| \leq \alpha \quad (1.18)$$

note that $\alpha = 1$ contains the set which satisfies 1.16 equation since

$$1 = \|I^{d \times d}\| = \|A^{d \times k}(A^{d \times k})^\top\| \leq \|A^{d \times k}\| \|(A^{d \times k})^\top\| \leq \alpha \quad (1.19)$$

so any $\alpha \geq 1$ will contain the set specified by the condition 1.16.

combining all of that together gives us the **cond-fMOOP** formulation of this problem as

$$\min_{\beta^{k \times 1} \in \mathbb{R}^{k \times 1}, A^{d \times k} \in \mathbb{R}^{d \times k}} \frac{1}{T - d + 1} \sum_{\forall t \in [d, T]} \|R_t^{n \times 1} - R_{t \times d}^{n \times d} \times A^{d \times k} \times \beta^{k \times 1}\|_2^2 \quad (1.20)$$

$$\max_{\beta^{k \times 1} \in \mathbb{R}^{k \times 1}, A^{d \times k} \in \mathbb{R}^{d \times k}} \frac{1}{S} \sum_{\forall t \in [T, T+S]} \frac{\langle R_t^{n \times 1}, R_{t \times d}^{n \times d} \times A^{d \times k} \times \beta^{k \times 1} \rangle}{\|R_t^{n \times 1}\|_2 \|R_{t \times d}^{n \times d} \times A^{d \times k} \times \beta^{k \times 1}\|_2} \quad (1.21)$$

subject to

$$\max_{R_{t \times d}^{n \times d} \in \mathbb{R}^{n \times d}} \text{cond}_{\text{rel}}(f, R_{t \times d}^{n \times d}) \leq \alpha \quad (1.22)$$

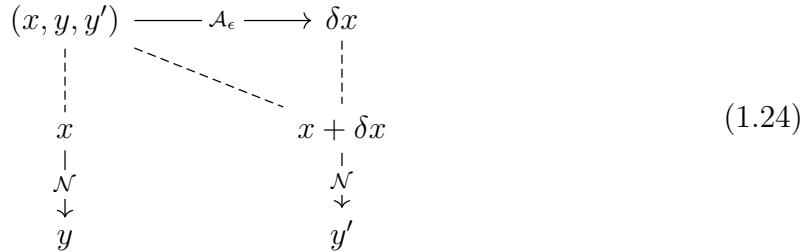
1.3.2 Defense Against Adversarial Attacks on Neural Networks

Say we are given a trained neural network $\mathcal{N} : X \rightarrow Y$ which has learnt a mapping from X , the input set to Y , the output set. If the underlining mapping that \mathcal{N} was modeled to learn was smooth w.r.t. its input then we expect that for a well trained network \mathcal{N} for any input $x \in X$ and for small enough $\delta x \in X$ δx the output of the network wont change much, lets say that it won't change more than a constant $\alpha > 0$ times the norm of x i.e. $\|\mathcal{N}(x + \delta x) - \mathcal{N}(x)\| = \|\delta \mathcal{N}(x)\| \leq \alpha \|\delta x\|$. Which we can reformulate as follows

$$\max_{x \in X} \text{cond}_{\text{abs}}(\mathcal{N}, x) = \lim_{\epsilon \rightarrow 0} \sup_{\|\delta x\| \leq \epsilon} \frac{\|\delta \mathcal{N}(x)\|}{\|\delta x\|} \leq \alpha \quad (1.23)$$

1. INTRODUCTION

Most of the networks which are being trained today don't account for such constraints giving the way to adversarial attacks on them, which exploit this drawbacks of the network to force them to output unreasonably wrong value.



Here \mathcal{A}_ϵ the adversary which takes the input (x, y, y') being the input data x , the associate label y , and the expected forced wrong output y' and outputs the required perturbation δx , such that $\|\delta x\| \leq \epsilon$. When δx is added to original input data the new malicious input $x_m = x + \delta x$ forces the network \mathcal{N} to output y' instead of y whereas it would have given output of y on the input x . Note that the adversary prefers smaller values of ϵ , since that implies that it can generate malicious input with as little changes as possible.

This problem can be avoided if we can train the network to also minimize the absolute condition number, $cond_{abs}(\mathcal{N}, x)$ over all the input space. We can also provide the constraints on their maximum values and model accordingly.

So if the network has been trained such that $\|\delta\mathcal{N}(x)\| \leq \alpha \|\delta x\|$ holds, then for the adversary \mathcal{A}_ϵ to make the network's \mathcal{N} output perturb by $\|\delta\mathcal{N}(x)\|$ it will have to change the input x value perturbation of norm more than $\frac{\|\delta\mathcal{N}(x)\|}{\alpha} \leq \|\delta x\|$, which will not be possible for an adversary with $\epsilon < \frac{\|\delta\mathcal{N}(x)\|}{\alpha}$. Hence training with such constraints will force the adversary to make large changes to the input for designing an malicious input, which is unfavourable for the adversary.

Chapter 2

Related Works

Before we look for the related works to this problems it is important to understand the different dimensions of this problem and structure the related works accordingly. The problems stated in the section 1.2 has 1 major characteristic apart from being fMOOP is that it requires the solution to have bounded condition number (absolute or relative), and on top of that since such formulation has many applications it is desirable to have efficient algorithms to compute such solutions. So the following dimensions of the literate that interests us and can have impact of the problem are:

1. Methods to solve MOOP
2. Literature related to condition number
3. Efficiency in the computational aspects of both 1, and 2

As we will see that lot of research has been done in solving MOOP and we have fairly efficient methods to get the solutions. But when it comes to the implications of condition number most of the research is focused on the condition number of matrices compared to that of condition number of general functions.

2.1 Multi-Objective Optimization

2.1.1 Definitions

In MOOP since there are more than one objectives its unlikely that all of them achieve their optima at the same point, hence typically there is no single global solution. Which makes its necessary to rethink about the definition for an optimum and accordingly determine a set of points that can be deemed as the solutions. The most widely used concept

2. RELATED WORKS

concept in defining an optimal point is that of the *Pareto optimality* [3], which is defined as follows:

Below we will use $F : X \rightarrow \mathbb{R}^k$ as the objective function that we need to minimize over the input $x \in X$ and we will use minimization over all the objectives since maximization can be converted to minimization by negating the sign.

Pareto Optimal: A point, $x^* \in X$, is Pareto optimal iff there does not exist another point, $x \in X$, such that $F(x) \leq F(x^*)$, and $F_i(x) < F_i(x^*)$ for at least one function.

All Pareto optimal points lie on the boundary of the feasible criterion space [9]. Sometimes the pareto optamility is too strong of a requirement hence we define *weakly Pareto optimal* as follows:

Weakly Pareto Optimal: A point, $x^* \in X$, is weakly Pareto optimal iff there does not exist another point, $x \in X$, such that $F(x) < F(x^*)$.

Pareto optimal points are weakly Pareto optimal, but weakly Pareto optimal points are not Pareto optimal.

Alternatively we define the compromise solution, which minimizes the difference between the utopia point/ideal point, defined as follows [6]:

Utopia Point : A point $F^* \in \mathbb{R}^k$ is objective space, is a utopia point iff for each $i = 1, 2, \dots, k$, $F_i^* = \min_{x \in X} \{F_i(x)\}$.

In general, F^* is infeasible due to other constrains. Then the next best solution that we can attain is a solution that is as close as possible to the utopia point. We call such solution a **compromise solution** and it is Pareto optimal. The meaning of close in this context needs to be clarified and generally we need to define norms to measure closeness of the solutions for example use of Euclidean norm [7]. Another problem with this approach is of different objective have different scales so generally we need to transform the objectives to a single scale for any meaningful use of the defined norm.

The methods to solve MOOP are classified into 4 classes [22] based on the availability and involvement of a external decision maker(*DM*) used to convey the preference over different pareto optimal solution.

1. **No preference methods:** No *DM* is available and a neutral compromise solution is identified without any specification of the preference information.

2. **A priori methods:** based on the preference information given by the *DM* the optimal solution is found.
3. **A posteriori methods:** a good representative set of Pareto optimal solutions is found, and from among them the *DM* must choose the best solution.
4. **Interactive methods:** Pareto optimal solution(s) are shown to the *DM*, then the *DM* describes how the solution(s) could be improved. Then the next set of Pareto optimal solution(s) is generated based on the *DM*'s feedback and iteratively the solutions are improved.

2.1.2 No Preference Methods

Since no preference information is provided the general one the approaches followed in [5] uses the definition of **utopia point** in 2.1.1 and then by properly scaling the objective function F to \hat{F} and using a $\|\cdot\|$ norm defined over the objective space the following optimizations problem is formulated to minimize the following objective.

$$\min_{x \in X} \|\hat{F}(x) - \hat{F}^*\| \quad (2.1)$$

Other similar methods are described in [24].

2.1.3 A Priori Methods

Methods which come under this class can further divided into other smaller classes but all of them have a common feature that is, enough information is provided a priori to compare any candidate pareto optimal solutions.

Following are some of the Important methods under this class:

ϵ -Constraint Method: In this method[4] we have a single most important objective function $F_s(x)$. and the remaining objective functions are used to form additional constraints $F_i(x) \leq \epsilon_i, \forall i \in [k]/\{s\}$.

$$\min_{x \in X} F_s(x) \quad (2.2)$$

subject to the constraints

$$F_i(x) \leq \epsilon_i, \quad \forall i \in [k]/\{s\} \quad (2.3)$$

2. RELATED WORKS

It is proven that by a systematic variation of ϵ_i one can generate a set of Pareto optimal solutions[21]. If the solution of 2.2,2.3 exists then it is a weakly Pareto optimal solution [25], and if the solution is unique, then it is Pareto optimal [25].

2.1.4 A Posteriori Methods

As the name suggest that the DM is involved a posteriori of finding solution, these approaches are also known as generate-first-choose-later approaches [11]. The goal is to produce a good enough representative subset of solutions which are Pareto optimal. In general they are classified into 2 classes.

1. **Mathematical programming methods**, which generally work by producing 1 pareto optimal solution per iteration/run of the algorithm.
2. **Evolutionary algorithms**, which produce a set of Pareto optimal solutions per iteration/run of the algorithm.

2.1.4.1 Evolutionary Algorithms (EA)

Evolutionary algorithms are one of the very actively researched methods for solving MOOP and finding pareto optimal solutions. EA are subset of the paradigm which is inspired by nature and evolution in designing algorithms for various purposes including solving optimization problems. The general procedure that an EA follows is described below:

Generic EA	
function EA(\mathcal{I})	▷ gets input parameters \mathcal{I}
$\mathcal{P} \leftarrow initialize(\mathcal{I})$	▷ initialize solution population \mathcal{P}
while converges(\mathcal{P}) \vee terminate(\mathcal{P}) do	▷ till convergence or termination
$\mathcal{P} \leftarrow evolve(\mathcal{P}, \mathcal{I})$	▷ evolve the population to next generation
return \mathcal{P}	

Some of the notable methods in EA which are commonly used are:

1. Non-dominated Sorting Genetic Algorithm-II (NSGA-II)
2. Particle swarm optimization (PSO)

Non-dominated Sorting Genetic Algorithm-II (NSGA-II): Proposed by K. Deb et al. in the paper [10], NSGA-II is based on elitist principle meaning only the elites

of the populations survive to the next generation based on a partial-order sorting of the population, to decide the elites.

Particle Swarm Optimization (PSO): PSO is based on the flocking behaviour of the birds where each particle has a position (the solution) and a velocity (change in the solution) where the velocity is influenced by some neighbourhood of the particle [13].

The advantage the EA provides is that it can quickly provide a sets of solutions which even though are not guaranteed to be pareto optimal, but are non-dominant set and serve as good approximation to the entirety of the Pareto front. The disadvantages being that these algorithms are relatively slow and pareto optimality cant be guaranteed.

2.2 Condition Number

The term **Condition Number** was first coined by Alan Turing in 1947 in his paper on Rounding-Off Errors in Matrix Processes [1]. He defined the condition number for matrices, further in 1966, John Rice in his paper *A Theory of Condition* [2] showed how to formulate the definition of condition number other classes of problems.

In our problem formulation may we not only require to compute the condition number of the function f to check if its bounded or not 1.11,1.12; but also to minimize the maximum over a set X_3 1.9,1.10.

Hence, we need to methods which can either be used to

1. Bound the relative/absolute condition number
2. Compute relative/absolute condition number efficiently

Extensive amount of literature is available on condition number of matrices compared to that of general functions. Below we will see some results on condition number of matrix and also for general function.

2.2.1 Condition Number of Matrices

There are several papers which have proved various important properties of matrix condition number. Pierre Maréchal and Jane J. Ye in their paper *Optimizing Condition*

2. RELATED WORKS

Numbers[15] showed that for a symmetric positive semi-definite $n \times n$ matrix A minimizing the condition number $\kappa(A)$.

$$\kappa_2(A) = \|A\|_2 \|A^{-1}\|_2 \quad (2.4)$$

$$\min_A \kappa_2(A) \quad (2.5)$$

2.5 is euqivalent to minimizing the following objective

$$\min_A \lambda_1(A) - \kappa_2(\bar{A}) \lambda_n(A) \quad (2.6)$$

$\|\cdot\|_F$ is Frobenius norm and $\sigma_i, i \in [n]$ are the singular value of A in decreasing order of their magnitude.

$$\|A\|_F = \sqrt{\sum_{i \in [n]} \sum_{j \in [m]} A_{ij}^2} \quad (2.7)$$

another interesting relation between $\kappa_2(A)$ and that of $\kappa_F(A)$ i.e. the condition number induced by Frobenius norm [8][14].

$$\frac{\kappa_F(A)}{n} \leq \kappa_2(A) \leq \kappa_F(A) \quad (2.8)$$

2.2.2 Condition Number of Functions

Edvin Deadman and Samuel D. Relton in their paper [16] extended the taylor's theorem for a complex function of matrices where $f : \mathbb{C} \rightarrow \mathbb{C}$ extended over matrices over $\mathbb{C}^{n \times n}$ i.e. $f : \mathbb{C}^{n \times n} \rightarrow \mathbb{C}$. For f satisfying some conditions they proved the bound on the absolute condition number of f at A for $\epsilon > 0$.

$$cond_{abs}(f, A) \leq \frac{L_\epsilon}{2\pi\epsilon^2} \max_{z \in \Gamma_\epsilon} |f(z)| \quad (2.9)$$

where Γ_ϵ is a closed counter of length L_ϵ which satisfies some condition, refer [16] for the specific conditions. A more extensive treatment and analysis of conditioning of such function is also done in the book by Nicholas J. Higham [20], based on the book a Matlab toolbox has also been made available by the name of *Matrix Function Toolbox* [19].

David H. Gutman and Javier F. Peña in their paper [17] extended the idea of condition number (relative condition number specifically) of a scalar function $f : \mathbb{R} \rightarrow \mathbb{R} \cup \{\infty\}$ extended over $f : \mathbb{R}^n \rightarrow \mathbb{R} \cup \{\infty\}$ restricted over a subset of the domain of f and sowed that the restriction has simialr properties as that of the condition number.

Chapter 3

Methodology and Results

We will focus on the problem that motivated this formulation - finding a well-conditioned solution for a Linear Factor Model 1.3.1.

We derive interesting results that show the importance of cond-fMOOP, and show how machine learning practices such as L_2 regularization approximate a form of cond-fMOOP. For the general class of cond-fMOOP problems, we note a list of questions, whose answers we think will be important contributors in solving the problems belonging to cond-fMOOP.

1. For a solvable $\mathcal{P} \in \text{fMOOP}$ and $\mathcal{P}' \in \text{cond-fMOOP}$, derived from \mathcal{P} . Is \mathcal{P}' solvable?
2. For $\mathcal{P} \in \text{cond-fMOOP}$ over X and subsets $X_5, X_6 \subseteq X$. How to determine minimum value of α for which \mathcal{P} is solvable?
3. For a $\mathcal{P} \in \text{cond-fMOOP}$, if we transform the condition number restriction using bounds, call the new problem \mathcal{P}' . How close are the solutions of \mathcal{P}' and \mathcal{P} ?

Question (1) is essential because not all fMOOP problems have solutions with bounded condition numbers. Having a method to identify such problems will enable us to focus on the right problems and develop methods to solve them accordingly.

Question (2) is crucial because even if a problem has solutions with bounded condition numbers, it may not have solutions with arbitrary small condition numbers. Thus, having a method to find the minimum value of α can determine the optimality of developed methods and how close they are to the optimal value.

Question (3) is also significant because computing the condition number is a costly task. However, we have many easy-to-compute bounds for the condition number. Thus, it is essential to know how close the solutions of \mathcal{P}' and \mathcal{P} are to judge the degree of approximation produced by using the bounds, which resolve the computational problem.

3. METHODOLOGY AND RESULTS

3.1 Theoretical Results

3.1.1 Condition Number over Composition of Function

Say we have a function $f = g(h(x)) = g \circ h(x)$ where $f : X \rightarrow Y$, $h : X \rightarrow Z$, and $g : Z \rightarrow Y$ then we can write

$$cond_{abs}(f, x) = \lim_{\epsilon \rightarrow 0} \sup_{\|\delta x\| \leq \epsilon} \frac{\|\delta f(x)\|}{\|\delta x\|} = \lim_{\epsilon \rightarrow 0} \sup_{\|\delta x\| \leq \epsilon} \frac{\|\delta g(h(x))\|}{\|\delta x\|} \quad (3.1)$$

\implies

$$cond_{abs}(f, x) = \lim_{\epsilon \rightarrow 0} \sup_{\|\delta x\| \leq \epsilon} \frac{\|\delta g(h(x))\|}{\|\delta h(x)\|} \frac{\|\delta h(x)\|}{\|\delta x\|} \quad (3.2)$$

under the assumption that $\forall \epsilon \geq 0$ and $\|\delta x\| \leq \epsilon \exists \delta \geq 0$ such that $\|\delta h(x)\| \leq \delta \forall x$ and $\lim_{\epsilon \rightarrow 0} \delta = 0$. Then we can write

$$cond_{abs}(f, x) = \lim_{\delta \rightarrow 0} \sup_{\|\delta h(x)\| \leq \delta} \frac{\|\delta g(h(x))\|}{\|\delta h(x)\|} \lim_{\epsilon \rightarrow 0} \sup_{\|\delta x\| \leq \epsilon} \frac{\|\delta h(x)\|}{\|\delta x\|} \quad (3.3)$$

let $h(x) = z \implies \delta h(x) = \delta z$, and since z is a function of x , δz is not independent of δx hence we can write

$$cond_{abs}(f, x) \leq \lim_{\delta \rightarrow 0} \sup_{\|\delta z\| \leq \delta} \frac{\|\delta g(z)\|}{\|\delta z\|} \lim_{\epsilon \rightarrow 0} \sup_{\|\delta x\| \leq \epsilon} \frac{\|\delta h(x)\|}{\|\delta x\|} \quad (3.4)$$

\implies

$$cond_{abs}(f, x) \leq cond_{abs}(g, h(x)) \times cond_{abs}(h, x) \quad (3.5)$$

the above 3.5 bound can be used to simplify the fMOOP with constraints on condition number for complex functions which are composition of simpler function whose condition number can be bound analytically, for eg. Neural Networks come under such functions.

3.1.2 L_2 Regularization & Absolute Condition Number

Consider the problem of fitting data with n data points $(X_i, Y_i), i \in [n]$, each column being 1 data point $(X, Y) \in (\mathbb{R}^{d \times n}, \mathbb{R}^{1 \times n})$ using a function $f : \mathbb{R}^{d \times 1} \rightarrow \mathbb{R}^{1 \times 1}$, for case of simplicity assume f is linear transform i.e. $f(X) = AX \approx Y$ where $A \in \mathbb{R}^{1 \times d}$.

Consider the L_2 regularization formulations of this problem as follows

$$\min_{A \in \mathbb{R}^{1 \times d}} \frac{1}{n} \sum_{i \in [n]} \|AX_i - Y_i\|_2 + \lambda \|A\|_2^2 \quad (3.6)$$

now, consider the same problem under fMOOP for minimizing the $\|\cdot\|_F$ frobenius norm of the transform A over the aggregation scheme of mean and $X_3 = \mathbb{R}^{d \times 1}$

$$\min_{A \in \mathbb{R}^{1 \times d}} \frac{1}{n} \sum_{i \in [n]} \|AX_i - Y_i\|_2 \quad (3.7)$$

$$\min_{A \in \mathbb{R}^{1 \times d}} \max_{x \in X_3} \text{cond}_{abs}(f, x) \quad (3.8)$$

by definition

$$\text{cond}_{abs}(f, x) = \lim_{\epsilon \rightarrow 0} \sup_{\|\delta x\|_F \leq \epsilon} \frac{\|A\delta x\|_F}{\|\delta x\|_F} \quad (3.9)$$

note that $\|A\delta x\|_F = \sqrt{(\sum_{i \in [d]} A_i \delta x_i)^2}$ and by Cauchy–Schwarz inequality we can write $(\sum_{i \in [d]} A_i \delta x_i)^2 \leq (\sum_{i \in [d]} A_i^2)(\sum_{i \in [d]} \delta x_i^2) = \|A\|_F^2 \|\delta x\|_F^2$, which implies

$$\text{cond}_{abs}(f, x) = \lim_{\epsilon \rightarrow 0} \sup_{\|\delta x\|_F \leq \epsilon} \frac{\|A\delta x\|_F}{\|\delta x\|_F} \leq \frac{\|A\|_F \|\delta x\|_F}{\|\delta x\|_F} = \|A\|_F \quad (3.10)$$

note that for this case $\|\cdot\|_F$ is equivalent to $\|\cdot\|_2$, and the above upper bound implies that the problem 3.8 when minimized for the worst case using the upper bound 3.10 we get the fMOOP as follows

$$\min_{A \in \mathbb{R}^{1 \times d}} (\|A\|_2, \frac{1}{n} \sum_{i \in [n]} \|AX_i - Y_i\|_2) \quad (3.11)$$

which when scalarized with squaring the norm restriction gives us the standard L_2 regularization.

3.2 Experimental Results

3.2.1 Methodologies

In the following section we will describe the methodologies we developed to solve Linear Factor Model problem with the condition $AA^\top = I$ 1.3.1.

3.2.1.1 Baseline Method

The following method is based on randomized sampling method, which involves generating the matrix M randomly and is orthonormalized to get A then using linear regression we get the value for β . This process is repeated multiple times, and the best result is selected from the attempts. In an alternate implementation of this algorithms we can even do grid search over the entries of A , but because $A \in \mathbb{R}^{d \times k}$, which is a very large space we restrict this method over randomized search. Note that the step 5, 6 together only take $O(k^3 + d^2k)$ time with one time overhead of $O(Tndk)$ as we can compute \mathbf{D}, \mathbf{N} as in 3.48, 3.49, then A compute β with just 4 matrix multiplications and 1 matrix inversion taking $O(k^3)$ operations. Following is the pseudocode for the method:

3. METHODOLOGY AND RESULTS

$\mathcal{A}_0[N]$: Baseline Method

```

1:  $\beta_{optimal}, A_{optimal}, \mathcal{R}_{optimal} \leftarrow \emptyset, \emptyset, -\infty$ 
2: for  $i \leftarrow 1 \dots N$  do
3:    $M_i \leftarrow random(\mathbb{R}^{d \times k})$ 
4:    $A_i \leftarrow gram\_schmidt\_algorithm(M_i)$                                  $\triangleright$  orthonormalize  $M_i$  to  $A_i$ 
5:   Compute  $\{F_{t,i}^{n \times k}\}$  for  $A_i$  over  $t \in [d, T]$ 
6:    $\beta_i \leftarrow regression(\{R_t^{n \times 1}\}, \{F_{t,i}^{n \times k}\}, t \in [d, T])$ 
7:   if  $\mathcal{R}_{\beta_i, A_i} > \mathcal{R}_{optimal}$  then           $\triangleright$  update the optimal if better solution found
8:      $\beta_{optimal}, A_{optimal}, \mathcal{R}_{optimal} \leftarrow \beta_i, A_i, \mathcal{R}_{\beta_i, A_i}$ 
9: return  $\beta_{optimal}, A_{optimal}$ 
10: complexity:  $O(N(dk + d^2k + ndS + k^3)) = O(N(k^3 + ndS))$ 

```

3.2.1.2 Gradient Based Methods

To compute the partial derivatives of the loss function $\mathcal{L}_{\beta^{k \times 1}, A^{d \times k}}$ with respect to $A^{d \times k}$ and $\beta^{k \times 1}$, we will use the chain rule and the derivative of the matrix multiplication.

first, let's compute the derivative of $\mathcal{L}_{\beta^{k \times 1}, A^{d \times k}}$ with respect to $\beta^{k \times 1}$. Lets Define $\Delta_t = R_t^{n \times 1} - R_{t \times d}^{n \times d} \times A^{d \times k} \times \beta^{k \times 1}$

$$\frac{\partial \mathcal{L}_{\beta^{k \times 1}, A^{d \times k}}}{\partial \beta^{k \times 1}} = \frac{1}{2(T-d+1)} \sum_{\forall t \in [d, T]} \frac{\partial}{\partial \beta^{k \times 1}} \|\Delta_t\|_2^2 \quad (3.12)$$

now, let's compute the derivative of $\frac{1}{2} \|\Delta_t\|_2^2$ with respect to $\beta^{k \times 1}$:

$$\frac{\partial}{\partial \beta^{k \times 1}} \|\Delta_t\|_2^2 = \frac{\partial}{\partial \beta^{k \times 1}} \Delta_t^\top \Delta_t = F_t^\top (F_t \beta - R_t) \quad (3.13)$$

which gives

$$\frac{\partial \mathcal{L}_{\beta^{k \times 1}, A^{d \times k}}}{\partial \beta^{k \times 1}} = \frac{1}{T-d+1} \sum_{\forall t \in [d, T]} F_t^\top (F_t \beta - R_t) \quad (3.14)$$

similarly consider

$$\frac{\partial \mathcal{L}_{\beta^{k \times 1}, A^{d \times k}}}{\partial A^{d \times k}} = \frac{1}{2(T-d+1)} \sum_{\forall t \in [d, T]} \frac{\partial}{\partial A^{d \times k}} \|\Delta_t\|_2^2 \quad (3.15)$$

we will use the following identities when a , b and C are not functions of X .

$$\frac{\partial(\mathbf{a}^\top \mathbf{X} \mathbf{b})}{\partial \mathbf{X}} = \mathbf{a} \mathbf{b}^\top \quad (3.16)$$

3.2 Experimental Results

$$\frac{\partial(\mathbf{a}^\top \mathbf{X}^\top \mathbf{b})}{\partial \mathbf{X}} = \mathbf{b} \mathbf{a}^\top \quad (3.17)$$

$$\frac{\partial(\mathbf{X}\mathbf{a})^\top \mathbf{C}(\mathbf{X}\mathbf{b})}{\partial \mathbf{X}} = \mathbf{C}\mathbf{X}\mathbf{b}\mathbf{a}^\top + \mathbf{C}^\top \mathbf{X}\mathbf{a}\mathbf{b}^\top \quad (3.18)$$

for the equations below and onwards we are dropping the dimensions

$$\|\Delta_t\|_2^2 = \Delta_t^\top \Delta_t = (R_t - R_{t \times d} A \beta)^\top (R_t - R_{t \times d} A \beta) \quad (3.19)$$

$$\|\Delta_t\|_2^2 = R_t^\top R_t - R_t^\top R_{t \times d} A \beta - \beta^\top A^\top R_{t \times d}^\top R_t + \beta^\top A^\top R_{t \times d}^\top R_{t \times d} A \beta \quad (3.20)$$

now using the identities 3.16, 3.17, 3.18 we get

$$\frac{\partial}{\partial A} \|\Delta_t\|_2^2 = R_{t \times d}^\top R_{t \times d} A \beta \beta^\top - R_{t \times d}^\top R_t \beta^\top \quad (3.21)$$

which gives

$$\frac{\partial \mathcal{L}_{\beta, A}}{\partial A} = \frac{1}{T-d+1} \sum_{\forall t \in [d, T]} R_{t \times d}^\top R_{t \times d} A \beta \beta^\top - R_{t \times d}^\top R_t \beta^\top \quad (3.22)$$

for vector-valued scalar functions $v(\mathbf{x})$ and $u(\mathbf{x})$ with respect to \mathbf{x} , and where A is not a function of x .

$$\frac{\partial}{\partial \mathbf{x}} (v(\mathbf{x}) u(\mathbf{x})) = v \frac{\partial u}{\partial \mathbf{x}} + \frac{\partial v}{\partial \mathbf{x}} u \quad (3.23)$$

$$\frac{\partial}{\partial \mathbf{x}} (\mathbf{x}^\top \mathbf{A} \mathbf{x}) = (\mathbf{A} + \mathbf{A}^\top) \mathbf{x} \quad (3.24)$$

$$\frac{\partial(\mathbf{A}\mathbf{x})}{\partial \mathbf{x}} = \mathbf{A}^\top \quad (3.25)$$

define $\rho_t = \frac{\langle R_t, R_{t \times d} A \beta \rangle}{\|R_t\|_2 \|R_{t \times d} A \beta\|_2} = \frac{R_t^\top R_{t \times d} A \beta}{\|R_t\|_2 \|R_{t \times d} A \beta\|_2} = \frac{R_t^\top F_t \beta}{\|R_t\|_2 \|F_t \beta\|_2}$ and $\hat{R}_t = \frac{R_t}{\|R_t\|_2}$, now using 3.24, 3.23, 3.25, 3.16 and 3.18 we get

$$\frac{\partial \rho_t}{\partial \beta} = \frac{1}{\|F_t \beta\|_2} F_t^\top \hat{R}_t - \frac{2 F_t^\top F_t \beta}{2 \|F_t \beta\|_2^3} \hat{R}_t^\top F_t \beta \quad (3.26)$$

$$\frac{\partial \rho_t}{\partial A} = \frac{1}{\|F_t \beta\|_2} R_{t \times d}^\top \hat{R}_t \beta^\top - \frac{2 R_{t \times d}^\top R_{t \times d} A \beta \beta^\top}{2 \|F_t \beta\|_2^3} \hat{R}_t^\top F_t \beta \quad (3.27)$$

from that we can get

$$\frac{\partial \mathcal{R}_{\beta, A}}{\partial A} = \frac{1}{S} \sum_{\forall t \in [T, T+S]} \frac{\partial \rho_t}{\partial A} \quad (3.28)$$

3. METHODOLOGY AND RESULTS

$$\frac{\partial \mathcal{R}_{\beta,A}}{\partial \beta} = \frac{1}{S} \sum_{\forall t \in [T, T+S]} \frac{\partial \rho_t}{\partial \beta} \quad (3.29)$$

also we can get derivative w.r.t. A for the frobenius norm square of $AA^\top - I$

$$\frac{\partial \|AA^\top - I\|_F^2}{\partial A} = -4 \cdot (I - A \cdot A^\top) \cdot A \quad (3.30)$$

Observation

Consider the set of matrices $\mathcal{O}_d^k = \{A | AA^\top = I_d, A \in \mathbb{R}^{d \times k}\}$, observe that \mathcal{O}_d^k is set of all orthonormal square matrix of size $k \times k$ and for a $Q \in \mathcal{O}_d^k$ we have $QQ^\top = I$, which implies $Q^{-1} = Q^\top$ and hence $QQ^\top = Q^\top Q = I$. Using the aforementioned fact we observe that $AQ \in \mathcal{O}_d^k$ as

$$(AQ)(AQ)^\top = AQQ^\top A^\top = AA^\top \quad (3.31)$$

and for $A \in \mathcal{O}_d^k$ and $Q \in \mathcal{O}_d^d$ we observe that $QA \in \mathcal{O}_d^k$

$$(QA)(QA)^\top = QAA^\top Q^\top = QQ^\top = I \quad (3.32)$$

now consider and iterative scheme for finding A, β which generates $\{A_i, \beta_i\}_{i=0}^N$ with updates at i 'th iteration denoted as $\{\Delta A_i, \Delta \beta_i\}$

$$\begin{aligned} A_{i+1} &= A_i + \Delta A_i \\ \beta_{i+1} &= \beta_i + \Delta \beta_i \end{aligned} \quad (3.33)$$

but since in our problem we have added restriction on $A \in \mathcal{O}_d^k$, which we can solve by one of the following 2 approaches

General approaches to satisfy orthonormality condition

1. Design the iteration scheme such that $A_i \in \mathcal{O}_d^k, \forall i \in \{0, 1, 2, \dots, N\}$
2. Relax the Condition on $A \in \mathcal{O}_d^k$ to $A \in \mathbb{R}^{d \times k}$ and solve using an iterative scheme; then project A_N to \mathcal{O}_d^k i.e. solve the problem $\min_{A \in \mathcal{O}_d^k} \|A_N - A\|$ with $\mathcal{R}_{\beta,A}, \mathcal{L}_{\beta,A}$ being reasonably close to $\mathcal{R}_{\beta_N, A_N}, \mathcal{L}_{\beta_N, A_N}$ respectively.

first we will use the observation 3.31 to design an iterative scheme such that $A_i \in \mathcal{O}_d^k, \forall i \in \{0, 1, 2, \dots, N\}$

lets say via some iterative method we get ΔA_i update for A_i but since $A_{i+1} = A_i + \Delta A_i$ need not belong to \mathcal{O}_d^k we need some way to project ΔA_i in a space such that $A_{i+1} \in \mathcal{O}_d^k$ we define \mathcal{O}_d^k

$$\delta \mathcal{O}_d^k = \{\delta | \delta = A - B, A, B \in \mathcal{O}_d^k\} \quad (3.34)$$

3.2 Experimental Results

which implies, we require to project ΔA_i in $\delta\mathcal{O}_d^k$ or to say $\Delta A_i \rightarrow \text{proj}_{\delta\mathcal{O}_d^k}(\Delta A_i)$

Properties of \mathcal{O}_d^k

1. $A \in \mathcal{O}_d^k \implies -A \in \mathcal{O}_d^k$
2. $A \in \mathcal{O}_d^k$ and $Q \in \mathcal{O}_d^k$ we have $AQ \in \mathcal{O}_d^k$, by 3.31
3. $A \in \mathcal{O}_d^k$ and $Q \in \mathcal{O}_d^d$ we have $QA \in \mathcal{O}_d^k$, by 3.32

Properties of $\delta\mathcal{O}_d^k$

1. $\bar{\mathbf{0}} \in \delta\mathcal{O}_d^k$
2. $A, B \in \mathcal{O}_d^k$ we have $A - B, A + B \in \delta\mathcal{O}_d^k$ as by property 1 of \mathcal{O}_d^k
3. $\delta \in \delta\mathcal{O}_d^k$ and $Q \in \mathcal{O}_d^k$ we have $\delta Q \in \delta\mathcal{O}_d^k$ by 3.31
4. $\delta \in \delta\mathcal{O}_d^k$ and $Q \in \mathcal{O}_d^d$ we have $Q\delta \in \delta\mathcal{O}_d^k$ by 3.32

So if we could get an associated $Q_i \in \mathcal{O}_d^k$ with ΔA_i such that we can replace $A_{i+1} = A_i + \text{proj}_{\delta\mathcal{O}_d^k}(\Delta A_i)$ with $\hat{A}_{i+1} = A_i Q_i$, such that $\hat{A}_{i+1} \approx A_{i+1}$.

now recall Cayley Transformation for a Q which doesn't have -1 as one of its eigenvalues then there is a skew-symmetric matrix $T = -T^\top$ satisfying the following properties

$$\begin{aligned} Q &= (I + T)^{-1}(I - T) = (I - T)(I + T)^{-1} \\ T &= (I - Q)(I + Q)^{-1} \end{aligned} \tag{3.35}$$

let $2T_i = \Delta A_i^\top A_i - A_i^\top \Delta A_i$ and we generate associated $Q_i = (I - T_i)(I + T_i)^{-1}$ by using Cayley Transformation over T .

then observe that $X = A_i Q_i - A_i = A_i(Q_i - I)$, multiplying by $(1 + T_i)$ on right side gives

$$\begin{aligned} X(1 + T_i) &= A_i((1 - T_i) - (1 + T_i)) \\ &= -A_i \cdot 2T_i \\ &= -A_i(\Delta A_i^\top A_i - A_i^\top \Delta A_i) \\ &= -A_i \Delta A_i^\top A_i + \Delta A_i \\ &= \Delta A_i - A_i \Delta A_i^\top A_i \end{aligned} \tag{3.36}$$

for T_i with maximum magnitude of eigenvalues < 1 and we know that

$$(1 + T_i)^{-1} = 1 - T_i + T_i^2 - \dots \tag{3.37}$$

3. METHODOLOGY AND RESULTS

using which we can write

$$\begin{aligned} X &= A_i Q_i - A_i \approx \Delta A_i \\ A_i Q_i &\approx A_i + \Delta A_i \end{aligned} \tag{3.38}$$

based on the above observation we can write the following iterative scheme

$\mathcal{A}_1[N, \mathcal{U}_\mathcal{A}, \mathcal{U}_\beta]$: Orthogonal Property Iterative Scheme (OPI Scheme)

- 1: $A_0 \leftarrow \text{random}(\mathbb{R}^{d \times k})$
 - 2: $A_0 \leftarrow \text{gram_schmidt_algorithm}(A_0)$ ▷ orthonormalize A_0 , as $A_0 \in \mathcal{O}_d^k$
 - 3: $\beta_0 \leftarrow \text{regression}(\{R_t^{n \times 1}\}, \{F_t^{n \times k}[A_0]\}, t \in [d, T])$
 - 4: **for** $i \leftarrow 0$ to $N - 1$ **do**
 - 5: $\Delta A_i \leftarrow \mathcal{U}_\mathcal{A}[\vec{\alpha}](\mathcal{R}_{\beta, A}, \mathcal{L}_{\beta, A}, \beta_i, A_i) - A_i$
 - 6: $2T_i \leftarrow \Delta A_i^\top A_i - A_i^\top \Delta A_i$
 - 7: $Q_i \leftarrow (I - T_i)(I + T_i)^{-1}$
 - 8: $A_{i+1} \leftarrow A_i Q_i$
 - 9: $\beta_{i+1} \leftarrow \mathcal{U}_\beta[\vec{\phi}](\mathcal{R}_{\beta, A}, \mathcal{L}_{\beta, A}, \beta_i, A_i, A_{i+1})$
 - 10: **return** β_N, A_N
 - 11: **complexity:** $O(N(|\mathcal{U}_\mathcal{A}[\vec{\alpha}]| + k^2 d + k^3 + k^2 d + |\mathcal{U}_\beta[\vec{\phi}]|)) = O(N(|\mathcal{U}_\mathcal{A}[\vec{\alpha}]| + k^3 + |\mathcal{U}_\beta[\vec{\phi}]|))$
-

for the second case were we relax the Condition on $A \in \mathcal{O}_d^k$ to $A \in \mathbb{R}^{d \times k}$ we can write a general iterative scheme as follows

$\mathcal{A}_2[N, \mathcal{U}_\mathcal{A}, \mathcal{U}_\beta, \mathcal{U}_\mathcal{O}]$: Delayed Orthogonalization Iterative Scheme (DOI Scheme)

- 1: $A_0 \leftarrow \text{random}(\mathbb{R}^{d \times k})$
 - 2: $\beta_0 \leftarrow \text{regression}(\{R_t^{n \times 1}\}, \{F_t^{n \times k}[A_0]\}, t \in [d, T])$
 - 3: **for** $i \leftarrow 0$ to $N - 1$ **do**
 - 4: $A_{i+1} \leftarrow \mathcal{U}_\mathcal{A}[\vec{\alpha}](\mathcal{R}_{\beta, A}, \mathcal{L}_{\beta, A}, \beta_i, A_i)$
 - 5: $\beta_{i+1} \leftarrow \mathcal{U}_\beta[\vec{\phi}](\mathcal{R}_{\beta, A}, \mathcal{L}_{\beta, A}, \beta_i, A_i, A_{i+1})$
 - 6: $\beta^*, A^* \leftarrow \mathcal{U}_\mathcal{O}[\vec{\lambda}](\mathcal{R}_{\beta, A}, \mathcal{L}_{\beta, A}, \beta_N, A_N)$
 - 7: **return** β^*, A^*
 - 8: **complexity:** $O(N(|\mathcal{U}_\mathcal{A}[\vec{\alpha}]| + |\mathcal{U}_\beta[\vec{\phi}]|) + |\mathcal{U}_\mathcal{O}[\vec{\lambda}]|)$
-

The above defined iterative scheme requires internal methods for getting A_{i+1} and β_{i+1} which are $\mathcal{U}_\mathcal{A}[\vec{\alpha}](\mathcal{R}_{\beta, A}, \mathcal{L}_{\beta, A}, \beta_i, A_i)$ and $\mathcal{U}_\beta[\vec{\phi}](\mathcal{R}_{\beta, A}, \mathcal{L}_{\beta, A}, \beta_i, A_i, A_{i+1})$ respectively. And the $\mathcal{U}_\mathcal{O}[\vec{\lambda}](\mathcal{R}_{\beta, A}, \mathcal{L}_{\beta, A}, \beta_i, A_i)$ solves for the following problem $\min_{A \in \mathcal{O}_d^k} \|A_N - A\|$ and corresponding β which minimizes the objectives $(-\mathcal{R}_{\beta, A}, \mathcal{L}_{\beta, A})$

3.2 Experimental Results

For $\mathcal{U}_A[\vec{\alpha}]$ ($\mathcal{R}_{\beta,A}, \mathcal{L}_{\beta,A}, \beta_i, A_i$)

Gradient Based Update $\mathcal{U}_A^1[\vec{\alpha}]$

Here $\vec{\alpha} = (\alpha_0, \alpha_1, \alpha_2, \alpha_3, \alpha_4)$ where α_0 is the number of iterations, α_1 is learning rate, α_2 is the weight of gradient of $-\mathcal{R}_{\beta_i, A}$, α_3 is the weight of gradient of $\mathcal{L}_{\beta_i, A}$, and α_4 is the weight of gradient of $\|AA^\top - I\|_F^2$.

we get $A_{i+1} = \mathcal{U}_A[\vec{\alpha}](\mathcal{R}_{\beta,A}, \mathcal{L}_{\beta,A}, \beta_i, A_i)$

$$A_{i+1} = A_i - \alpha_1 \left(-\alpha_2 \frac{\partial \mathcal{R}_{\beta_i, A_i}}{\partial A} + \alpha_3 \frac{\partial \mathcal{L}_{\beta_i, A_i}}{\partial A} + \alpha_4 \frac{\partial \|A_i A_i^\top - I\|_F^2}{\partial A} \right) \quad (3.39)$$

$$\Delta A_i = \alpha_1 \left(\alpha_2 \frac{\partial \mathcal{R}_{\beta_i, A_i}}{\partial A} - \alpha_3 \frac{\partial \mathcal{L}_{\beta_i, A_i}}{\partial A} - \alpha_4 \frac{\partial \|A_i A_i^\top - I\|_F^2}{\partial A} \right) \quad (3.40)$$

For $\mathcal{U}_\beta[\vec{\phi}]$ ($\mathcal{R}_{\beta,A}, \mathcal{L}_{\beta,A}, \beta_i, A_i, A_{i+1}$)

Gradient Based Update $\mathcal{U}_\beta^1[\vec{\phi}]$

Here $\vec{\phi} = (\phi_0, \phi_1, \phi_2, \phi_3)$ where ϕ_0 is the number of iterations, ϕ_1 is learning rate, ϕ_2 is the weight of gradient of $-\mathcal{R}_{\beta, A_{i+1}}$, ϕ_3 is the weight of gradient of $\mathcal{L}_{\beta, A_{i+1}}$.

we get $\beta_{i+1} = \mathcal{U}_\beta[\vec{\phi}](\mathcal{R}_{\beta,A}, \mathcal{L}_{\beta,A}, \beta_i, A_i, A_{i+1})$

$$\beta_{i+1} = \beta_i - \phi_1 \left(-\phi_2 \frac{\partial \mathcal{R}_{\beta_i, A_i}}{\partial \beta} + \phi_3 \frac{\partial \mathcal{L}_{\beta_i, A_i}}{\partial \beta} \right) \quad (3.41)$$

$$\Delta \beta_i = \phi_1 \left(\phi_2 \frac{\partial \mathcal{R}_{\beta_i, A_i}}{\partial \beta} - \phi_3 \frac{\partial \mathcal{L}_{\beta_i, A_i}}{\partial \beta} \right) \quad (3.42)$$

Gradient Based Update $\mathcal{U}_\beta^2[\vec{\phi}]$ with Regression

Here $\vec{\phi} = (\phi_0, \phi_1, \phi_2, \phi_3, \phi_4)$ where ϕ_0 is the number of iterations, ϕ_1 is learning rate, ϕ_2 is the weight of gradient of $-\mathcal{R}_{\beta, A_{i+1}}$, ϕ_3 is the weight of gradient of $\mathcal{L}_{\beta, A_{i+1}}$, and ϕ_4 is the weight for the change in β_i w.r.t. the regressed value of $\hat{\beta}_{i+1} = \text{regression}(\{R_t^{n \times 1}\}, \{F_{t,i+1}^{n \times k}\}, t \in [d, T])$ where $\{F_{t,i+1}^{n \times k}\}$ are derived from A_{i+1} .

we get $\beta_{i+1} = \mathcal{U}_\beta[\vec{\phi}](\mathcal{R}_{\beta,A}, \mathcal{L}_{\beta,A}, \beta_i, A_i, A_{i+1})$

$$\beta_{i+1} = (1 - \phi_4)(\beta_i - \phi_1 \left(-\phi_2 \frac{\partial \mathcal{R}_{\beta_i, A_i}}{\partial \beta} + \phi_3 \frac{\partial \mathcal{L}_{\beta_i, A_i}}{\partial \beta} \right)) + \phi_4(\hat{\beta}_{i+1} - \beta_i) \quad (3.43)$$

$$\Delta \beta_i = (1 - \phi_4)\phi_1 \left(\phi_2 \frac{\partial \mathcal{R}_{\beta_i, A_i}}{\partial \beta} - \phi_3 \frac{\partial \mathcal{L}_{\beta_i, A_i}}{\partial \beta} \right) + \phi_4(\hat{\beta}_{i+1} - 2\beta_i) \quad (3.44)$$

For $\mathcal{U}_O[\vec{\lambda}]$ ($\mathcal{R}_{\beta,A}, \mathcal{L}_{\beta,A}, \beta^*, A^*$)

Here $\vec{\lambda} = (\lambda_0, \lambda_1, \lambda_2, \lambda_3)$ where λ_0 is the number of iterations, λ_1 is the step size for update

3. METHODOLOGY AND RESULTS

in A , λ_2 is the step size for update in β , λ_3 is the weight of regression change, and let $N = \lambda_0$.

$\mathcal{U}_O^1[\vec{\lambda}](\mathcal{R}_{\beta,A}, \mathcal{L}_{\beta,A}, \beta^*, A^*)$: Iterative Closing Method

```

1:  $A \leftarrow \text{random}(\mathbb{R}^{d \times k})$ 
2:  $A \leftarrow \text{gram\_schmidt\_algorithm}(A)$   $\triangleright$  orthonormalize  $A$ 
3:  $\beta \leftarrow \text{regression}(\{R_t^{n \times 1}\}, \{F_t^{n \times k}[A]\}, t \in [d, T])$ 
4:  $\beta_{optimal}, A_{optimal}, \mathcal{R}_{optimal} \leftarrow \beta, A, \mathcal{R}_{\beta,A}$ 
5: for  $i \leftarrow 1$  to  $N$  do
6:    $\Delta A \leftarrow A^* - A$ 
7:    $Q \leftarrow \text{ortho\_cayley\_transformation}(\frac{\lambda_1}{2} \Delta A)$ 
8:    $A \leftarrow AQ$ 
9:    $\beta_{reg} \leftarrow \text{regression}(\{R_t^{n \times 1}\}, \{F_t^{n \times k}[A]\}, t \in [d, T])$ 
10:   $\Delta \beta \leftarrow (1 - \lambda_3) \frac{\partial \mathcal{R}_{\beta,A}}{\partial \beta} + \lambda_3 (\beta_{reg} - \text{beta})$ 
11:   $\beta \leftarrow \beta + \lambda_2 \Delta \beta$ 
12:  if  $\mathcal{R}_{\beta,A} > \mathcal{R}_{optimal}$  then  $\triangleright$  update the optimal if better solution found
13:     $\beta_{optimal}, A_{optimal}, \mathcal{R}_{optimal} \leftarrow \beta, A, \mathcal{R}_{\beta,A}$ 
14: return  $\beta_{optimal}, A_{optimal}$ 
15: complexity:  $O(k^3 + N(k^3 + ndS)) = O(N(k^3 + ndS))$ 

```

Observation: For a given A and for $\{F_t^{n \times k}\}$ computed w.r.t. A over $t \in [d, T]$ and $\beta \leftarrow \text{regression}(\{R_t^{n \times 1}\}, \{F_t^{n \times k}\}, t \in [d, T])$. Here we are using L_2 norm for regression which gives us the formula for β , as we know that β is the solution of the equation 3.14 and using the fact that $F_t^{n \times k} = R_t^{n \times d} A$, we can write a formula for β as follows

$$\frac{\partial \mathcal{L}_{\beta,A}}{\partial \beta} = 0 \quad (3.45)$$

$$\sum_{\forall t \in [d, T]} (F_t^{n \times k})^\top F_t^{n \times k} \beta - \sum_{\forall t \in [d, T]} (F_t^{n \times k})^\top R_t^{n \times 1} = 0 \quad (3.46)$$

we can write $(F_t^{n \times k})^\top F_t^{n \times k} = A^\top (R_t^{n \times d})^\top R_t^{n \times d} A$ and $(F_t^{n \times k})^\top R_t^{n \times 1} = A^\top (R_t^{n \times d})^\top R_t^{n \times 1}$

$$\sum_{\forall t \in [d, T]} A^\top (R_t^{n \times d})^\top R_t^{n \times d} A \beta - \sum_{\forall t \in [d, T]} A^\top (R_t^{n \times d})^\top R_t^{n \times 1} = 0 \quad (3.47)$$

$$\mathbf{D} = \sum_{\forall t \in [d, T]} (R_t^{n \times d})^\top R_t^{n \times d} \quad (3.48)$$

$$\mathbf{N} = \sum_{\forall t \in [d, T]} (R_t^{n \times d})^\top R_t^{n \times 1} \quad (3.49)$$

3.2 Experimental Results

$$A^\top \mathbf{D} A \beta = A^\top \mathbf{N} \quad (3.50)$$

$$\beta = (A^\top \mathbf{D} A)^{-1} A^\top \mathbf{N} \quad (3.51)$$

so if $\bar{A} = AQ$ where $QQ^\top = I$ then we can write using 3.51 for \bar{A}

$$\begin{aligned} \bar{\beta} &= (\bar{A}^\top \mathbf{D} \bar{A})^{-1} \bar{A}^\top \mathbf{N} \\ \bar{\beta} &= (Q^\top A^\top \mathbf{D} A Q)^{-1} Q^\top A^\top \mathbf{N} \\ \bar{\beta} &= Q^\top (A^\top \mathbf{D} A)^{-1} Q Q^\top A^\top \mathbf{N} \\ \bar{\beta} &= Q^\top \beta \end{aligned} \quad (3.52)$$

Hence, if $\bar{A} = AQ$ then if $\beta, \bar{\beta}$ solve regression problem generated by A, \bar{A} respectively then we get $\bar{\beta} = Q^\top \beta$.

3.2.1.3 No Preference Method

Under MOOP methods we decided to use No Preference method, because of its no dependence on of use defined parameters and easy to formulate and compute.

First we compute the **utopia point** as defined in 2.1.1 for this problem, here we have 2 dimensions for the objective to minimize namely $-\mathcal{R}_{\beta,A}$ and $\mathcal{L}_{\beta,A}$ and the utopia point values denoted by $-\mathcal{R}_{\beta,A}^* = \min -\mathcal{R}_{\beta,A} = -\max \mathcal{R}_{\beta,A} = -1$ as its inner-product between vectors divided by the norms induced by that inner-product in and $\mathcal{L}_{\beta,A}^* = \min \mathcal{L}_{\beta,A} = 0$. And finally we have the condition that $AA^\top = I$, which either we can view as constraint or consider I as the utopia point value of AA^\top . Finally we can consider one of the 2 optimization problem framed under *No preference method*

No Preference Method Optimization Problem 1

$$\min_{\beta \in \mathbb{R}^{k \times 1}, A \in \mathcal{O}_d^k} \|[\mathcal{L}_{\beta,A} - \mathcal{L}_{\beta,A}^*, -\mathcal{R}_{\beta,A} + \mathcal{R}_{\beta,A}^*]\| \quad (3.53)$$

in this case we can use any norm $\|\cdot\|$ defiend on \mathbb{R}^2 for example euclidean norm.

No Preference Method Optimization Problem 2

$$\min_{\beta \in \mathbb{R}^{k \times 1}, A \in \mathbb{R}^{d \times k}} \|[\mathcal{L}_{\beta,A} - \mathcal{L}_{\beta,A}^*, -\mathcal{R}_{\beta,A} + \mathcal{R}_{\beta,A}^*, AA^\top - I]\| \quad (3.54)$$

in this case we can use any norm $\|\cdot\|$ defiend on \mathbb{R}^{2+d^2} as $AA^\top - I$ is a $d \times d$ matrix which can vectorize to get a $d \times d$ vector for the purposes of computing the norm.

3. METHODOLOGY AND RESULTS

$NPM_1[\vec{\mu}]$: No Preference Method 1

```

1:  $A \leftarrow \text{random}(\mathbb{R}^{d \times k})$ 
2:  $A \leftarrow \text{gram\_schmidt\_algorithm}(A)$  ▷ orthonormalize  $A$ 
3:  $\beta \leftarrow \text{regression}(\{R_t^{n \times 1}\}, \{F_t^{n \times k}[A]\}, t \in [d, T])$ 
4:  $\beta_{\text{optimal}}, A_{\text{optimal}}, \mathcal{R}_{\text{optimal}} \leftarrow \beta, A, \mathcal{R}_{\beta, A}$ 
5: for  $i \leftarrow 1$  to  $N$  do
6:    $\Delta A \leftarrow (1 - \mathcal{R}_{\beta, A}) \frac{\partial \mathcal{R}_{\beta, A}}{\partial A} - \mathcal{L}_{\beta, A} \frac{\partial \mathcal{L}_{\beta, A}}{\partial A} - \frac{1}{2} \frac{\partial \|AA^\top - I\|_F^2}{\partial A}$ 
7:    $\Delta \beta \leftarrow (1 - \mathcal{R}_{\beta, A}) \frac{\partial \mathcal{R}_{\beta, A}}{\partial \beta} - \mathcal{L}_{\beta, A} \frac{\partial \mathcal{L}_{\beta, A}}{\partial \beta}$ 
8:    $Q \leftarrow \text{ortho\_cayley\_transformation}(\frac{\mu_1}{2} \Delta A)$ 
9:    $A \leftarrow AQ$ 
10:   $\beta \leftarrow \beta + \mu_2 \Delta \beta$ 
11:  if  $\mathcal{R}_{\beta, A} > \mathcal{R}_{\text{optimal}}$  then ▷ update the optimal if better solution found
12:     $\beta_{\text{optimal}}, A_{\text{optimal}}, \mathcal{R}_{\text{optimal}} \leftarrow \beta, A, \mathcal{R}_{\beta, A}$ 
13: return  $\beta_{\text{optimal}}, A_{\text{optimal}}$ 
14: complexity:  $O(N(ndS + k^3))$ 

```

$NPM_2[\vec{\mu}]$: No Preference Method 2

```

1:  $A \leftarrow \text{random}(\mathbb{R}^{d \times k})$ 
2:  $A \leftarrow \text{gram\_schmidt\_algorithm}(A)$  ▷ orthonormalize  $A$ 
3:  $\beta \leftarrow \text{regression}(\{R_t^{n \times 1}\}, \{F_t^{n \times k}[A]\}, t \in [d, T])$ 
4:  $\beta_{\text{optimal}}, A_{\text{optimal}}, \mathcal{R}_{\text{optimal}} \leftarrow \beta, A, \mathcal{R}_{\beta, A}$ 
5: for  $i \leftarrow 1$  to  $N$  do
6:    $\Delta A \leftarrow (1 - \mathcal{R}_{\beta, A}) \frac{\partial \mathcal{R}_{\beta, A}}{\partial A} - \mathcal{L}_{\beta, A} \frac{\partial \mathcal{L}_{\beta, A}}{\partial A} - \frac{1}{2} \frac{\partial \|AA^\top - I\|_F^2}{\partial A}$ 
7:    $\Delta \beta \leftarrow (1 - \mathcal{R}_{\beta, A}) \frac{\partial \mathcal{R}_{\beta, A}}{\partial \beta} - \mathcal{L}_{\beta, A} \frac{\partial \mathcal{L}_{\beta, A}}{\partial \beta}$ 
8:    $A \leftarrow A + \mu_1 \Delta A$ 
9:    $\beta \leftarrow \beta + \mu_2 \Delta \beta$ 
10:  if  $\mathcal{R}_{\beta, A} > \mathcal{R}_{\text{optimal}}$  then ▷ update the optimal if better solution found
11:     $\beta_{\text{optimal}}, A_{\text{optimal}}, \mathcal{R}_{\text{optimal}} \leftarrow \beta, A, \mathcal{R}_{\beta, A}$ 
12: return  $\beta_{\text{optimal}}, A_{\text{optimal}}$ 
13: complexity:  $O(N(ndS + k^2))$ 

```

3.2.1.4 Evolutionary Algorithms

The main problem with applying Evolutionary Algorithms for this problem is that we need to search for A, β but $A \in \mathcal{O}_d^k$ which is not closed under addition of any random

matrix in $\mathbb{R}^{d \times k}$ so instead for perturbing A for purposes of mutation and crossover we will use the Cayley Transformation 3.35 over the perturbation to get the Q , which will be used to multiplicatively perturb A .

3.2.1.4.1 NSGA-II

Below is the pseudocode for NSGA-II for Population size N , maximum number of generations G , crossover probability p_c and mutation probability p_m , crossover function \mathcal{C} , mutation function \mathcal{M}

NSGA-II($N, G, p_c, p_m, \mathcal{C}, \mathcal{M}$)

```

1:  $P \leftarrow \text{random\_candidate\_solutions}(N)$ 
2: Evaluate the fitness of each candidate solution in  $P$             $\triangleright$  fitness : objective function
3: for  $t \leftarrow 1$  to  $G$  do
4:    $C \leftarrow \text{get\_offspring\_population}(P, N, \mathcal{C}, \mathcal{M})$ 
5:    $F_1, F_2, \dots, F_t \leftarrow \text{non\_dominated\_sorting}(P \cup C)$             $\triangleright F_i$  is the  $i$ 'th front
6:    $P, i \leftarrow \emptyset, 1$ 
7:   while  $|P| + |F_i| \leq N$  do
8:     Calculate crowding distance for each solution in  $F_i$ 
9:      $P \leftarrow P \cup F_i$ 
10:     $i \leftarrow i + 1$ 
11:   if  $|P| < N$  then
12:     Sort  $F_i$  in descending order of crowding distance
13:      $P \leftarrow P \cup F_i[1 : N - |P|]$ 
14: return  $P$ 

```

`non_dominated_sorting(P)` : Sorts the population P into different non-dominated fronts F_1, F_2, \dots based on the Pareto-dominance relationship i.e. we say $x \preceq y$ or x dominates y if x is no worst when compared to y in all the objectives but is better at least some of the objective. The first front F_1 contains the non-dominated solutions, the subset of P which contains all the solutions which are not dominated by any other solution in P . For $i > 1$, the i 'th front F_i contains the solutions that are only dominated by the solutions in the first $i - 1$ sets namely F_1, F_2, \dots, F_{i-1} , for more details and algorithm to compute the F_i 's refer [10].

Crowding distance and crowding comparator: The crowding distance is a measure of crowdedness around a solution in its front F_i . Larger the value of crowding distances

3. METHODOLOGY AND RESULTS

implies that there aren't many closeby solution in the front, hence solutions with larger crowding distances are preferred, as they give more diversity to the population. For exact definition of the Crowding distance refer [10].

Since $S = (\beta, A)$ and $A \in \mathcal{O}_d^k$, we need to use $crossover[\mathcal{C}](S_1, S_2)$ and $mutate[\mathcal{M}](S_1, S_2)$ which are customized for our problem to ensure that new updated solution after crossover and mutation have A' s $\in \mathcal{O}_d^k$ and also need to modify $random_candidate_solutions(N)$ so that the initial solutions have A' s $\in \mathcal{O}_d^k$. For initial population we can simply orthonormalize A 's in the initial population.

$\mathcal{C}[Case] : crossover(S_1, S_2)$

Require: $A_1, A_2 \in \mathcal{O}_d^k$

- 1: $(\beta_1, A_1), (\beta_2, A_2) \leftarrow S_1, S_2$
 - 2: $\Delta A_1, \Delta A_2 \leftarrow (A_2 - A_1), (A_1 - A_2)$
 - 3: $Q_1, Q_2 \leftarrow \text{ortho_cayley_transformation}(\frac{\lambda}{2}\Delta A_1), \text{ortho_cayley_transformation}(\frac{\lambda}{2}\Delta A_2)$
 - 4: $A'_1, A'_2 \leftarrow A_1 Q_1, A_2 Q_2$
 - 5: **if** $Case$ is Use Regression **then**
 - 6: $\beta'_i \leftarrow regression(\{R_t^{n \times 1}\}, \{F_t^{n \times k}[A'_i]\}, t \in [d, T])$ for $i \in \{1, 2\}$
 - 7: **else**
 - 8: $\beta'_1, \beta'_2 \leftarrow \beta_1 + \lambda(\beta_2 - \beta_1), \beta_2 + \lambda(\beta_1 - \beta_2)$
 - 9: **return** $(\beta'_1, A'_1), (\beta'_2, A'_2)$
 - 10: **complexity:** $O(k^3)$
-

$\text{get_offspring_population}(P, N, \mathcal{C}, \mathcal{M})$

Require: N to be even

- 1: $Q \leftarrow \emptyset$
 - 2: **for** $i \leftarrow 1 \dots N$ with step size of 2 **do**
 - 3: $S_1, S_2 \leftarrow \text{binary_tournament_selection}(P)$ $\triangleright 2$ candidate solutions from P
 - 4: **if** True with probability p_c **then**
 - 5: $S_1, S_2 \leftarrow crossover[\mathcal{C}](S_1, S_2)$
 - 6: **if** True with probability p_m **then**
 - 7: $S_1, S_2 \leftarrow mutate[\mathcal{M}](S_1), mutate[\mathcal{M}](S_2)$
 - 8: Evaluate the fitness of S_1 and S_2
 - 9: $Q \leftarrow Q \cup \{S_1, S_2\}$
 - 10: **return** Q
-

3.2 Experimental Results

\mathcal{M} : mutate(S)

Require: $A \in \mathcal{O}_d^k$, $\lambda \in \mathbb{R}^+$

- 1: $(\beta, A) \leftarrow S$
 - 2: $\Delta A, \Delta \beta \leftarrow Random(\mathbb{R}^{d \times k}), Random(\mathbb{R}^{k \times 1})$
 - 3: $Q \leftarrow \text{ortho_cayley_transformation}(\frac{\lambda}{2} \Delta A)$
 - 4: **return** $(\beta + \lambda \Delta \beta, A Q)$
 - 5: **complexity:** $O(k^3)$
-

3.2.1.4.2 Multi-Objective Particle Swarm Optimization

Several multiobjective optimization algorithms are based on PSO, including Multiobjective Particle Swarm Optimization (MOPSO) [12], Nondominated Sorting Particle Swarm Optimization (NSPSO) [Li] and others. In this method we will be using the variant described in the paper 'An Effective Use of Crowding Distance in Multiobjective Particle Swarm Optimization' [26]

The pseudocode for the described method is as follows:

update_position[ϵ](\vec{x}_i, \vec{v}_i)

- 1: $(\beta_i, A_i) \leftarrow x_i$
 - 2: $(\Delta \beta_i, \Delta A_i) \leftarrow v_i$
 - 3: $Q_i \leftarrow \text{ortho_cayley_transformation}(\frac{\epsilon}{2} \Delta A_i)$
 - 4: **return** $(\beta + \epsilon \Delta \beta_i, A_i Q_i)$
-

nondominated_merge[M]($A, \{\vec{x}_i\}$)

- 1: $F \leftarrow \text{nondominated}(\{\vec{x}_i\})$
 - 2: $F_1, F_2, \dots, F_t \leftarrow \text{non_dominated_sorting}(F \cup A)$
 - 3: $A, i \leftarrow \emptyset, 1$
 - 4: **if** $|F_1| > M$ **then**
 - 5: Sort F_1 in descending order of crowding distance
 - 6: $F_1 \leftarrow F_1[1 : M]$
 - 7: **return** F_1
-

update_velocity[ω, c_1, c_2]($\vec{x}_i, \vec{v}_i, \vec{p}_i, \vec{p}_g$)

- 1: $r_1, r_2 \sim U(0, 1)$
 - 2: $v_i \leftarrow \omega v_i + c_1 r_1 (\vec{p}_i - \vec{x}_i) + c_2 r_2 (\vec{p}_g - \vec{x}_i)$
 - 3: **return** v_i
-

3. METHODOLOGY AND RESULTS

MOPSO($N, T_{max}, \theta, \omega, c_1, c_2, p_m, \epsilon$) Multi-Objective Particle Swarm Optimization

- 1: Initialize the population of particles randomly:
- 2: $\{\vec{x}_i\}, \{\vec{v}_i\} \leftarrow \text{initialize_position_and_velocity}(N)$
- 3: $\{\vec{f}(\vec{x}_i)\} \leftarrow \text{objective_function}(\{\vec{x}_i\})$
- 4: $\{\vec{p}_i\}, \{\vec{f}(\vec{p}_i)\} \leftarrow \{\vec{x}_i\}, \{\vec{f}(\vec{x}_i)\}$ $\triangleright \vec{p}_i, \vec{f}(\vec{p}_i)$: best positions and objective for each \vec{x}_i
- 5: $A \leftarrow \text{nondominated}(\{\vec{x}_i\})$ $\triangleright A$ is the archive of non-dominated solutions
- 6: $\{\vec{d}(\vec{a}_i)\} \leftarrow \text{crowding_distance}(A)$ $\triangleright d(\vec{a}_i)$: average distance to \vec{a}_i 's neighbors in A
- 7: **for** $t \leftarrow 1$ to T_{max} **do**
- 8: **for** $i \leftarrow 1$ to N **do**
- 9: $\vec{p}_g \leftarrow \text{get_global_best}[\theta](A)$ \triangleright global best w.p. θ else random point from A
- 10: $\vec{v}_i \leftarrow \text{update_velocity}[\omega, c_1, c_2](\vec{x}_i, \vec{v}_i, \vec{p}_i, \vec{p}_g)$
- 11: $\vec{x}_i \leftarrow \text{update_position}[\epsilon](\vec{x}_i, \vec{v}_i)$
- 12: **if** True with probability p_m **then**
- 13: $\vec{x}_i, \vec{v}_i \leftarrow \text{mutate}(\vec{x}_i, \vec{v}_i)$ \triangleright Mutate the solutions particle \vec{x}_i, \vec{v}_i
- 14: $\vec{f}(\vec{x}_i) \leftarrow \text{objective_function}(\vec{x}_i)$ \triangleright Update the objective of particle \vec{x}_i
- 15: $A \leftarrow \text{nondominated_merge}(A, \{\vec{x}_i\})$ \triangleright Update the nondominated solutions
- 16: $\{\vec{d}(\vec{a}_i)\} \leftarrow \text{crowding_distance}[\text{True}](A)$
- 17: $\{\vec{d}(\vec{x}_i)\} \leftarrow \text{crowding_distance}[\text{False}](\{\vec{x}_i\})$
- 18: $\{\vec{p}_i\}, \{\vec{f}(\vec{p}_i)\} \leftarrow \text{update_personal_best}(\{\vec{p}_i\}, \{\vec{f}(\vec{p}_i)\}, \{\vec{x}_i\}, \{\vec{f}(\vec{x}_i)\}, \{\vec{d}(\vec{x}_i)\})$
- 19: **return** A

`initialize_position_and_velocity(N)`

- 1: $P, V \leftarrow \emptyset, \emptyset$
- 2: **for** $i \leftarrow 1$ to N **do**
- 3: $A_i \leftarrow \text{random}(\mathbb{R}^{d \times k})$
- 4: $A_i \leftarrow \text{gram_schmidt_algorithm}(A_i)$
- 5: $\beta_i \leftarrow \text{regression}(\{R_t^{n \times 1}\}, \{F_{t,i}^{n \times k}[A_i]\}, t \in [d, T])$
- 6: $(\Delta\beta_i, \Delta A_i) \leftarrow (\frac{\partial \mathcal{R}_{\beta_i, A_i}}{\partial \beta}, \frac{\partial \mathcal{R}_{\beta_i, A_i}}{\partial A})$
- 7: $x_i, v_i \leftarrow (\beta_i, A_i), (\Delta\beta_i, \Delta A_i)$
- 8: $P \leftarrow P \cup \{x_i\}$
- 9: $V \leftarrow V \cup \{v_i\}$
- 10: **return** P, V

3.2 Experimental Results

mutate[$\vec{\lambda}$](\vec{x}_i, \vec{v}_i)

Require: $A_i \in \mathcal{O}_d^k$

- 1: $(\beta_i, A_i) \leftarrow \vec{x}_i$
 - 2: $\Delta A_i, \Delta \beta_i \leftarrow \text{Random}(\mathbb{R}^{d \times k}), \text{Random}(\mathbb{R}^{k \times 1})$
 - 3: $Q_i \leftarrow \text{ortho_cayley_transformation}(\frac{\lambda_1}{2} \Delta A_i)$
 - 4: $\vec{x}'_i \leftarrow (\beta_i + \lambda_1 \Delta \beta_i, A_i Q_i)$
 - 5: $\vec{v}'_i \leftarrow (\vec{v}_i \cdot \beta_i + \lambda_2 \text{Random}(\mathbb{R}^{k \times 1}), \vec{v}_i \cdot A_i + \lambda_2 \text{Random}(\mathbb{R}^{d \times k}))$
 - 6: **return** \vec{x}'_i, \vec{v}'_i
-

crowding_distance[Case](A)

Require: Case to be either **True** or **False**

- 1: **if** Case is **True** **then** ▷ Already a nondominated set
 - 2: $t, F_1 \leftarrow 1, A$
 - 3: **if** Case is **False** **then** ▷ Not a nondominated set
 - 4: $F_1, F_2, \dots, F_t \leftarrow \text{non_dominated_sorting}(A)$
 - 5: **for** $i \leftarrow 1$ to t **do**
 - 6: **for** a in F_i **do**
 - 7: $d(a) \leftarrow 0$
 - 8: **for** $j \leftarrow 1$ to D **do** ▷ D is the number of objective
 - 9: $F_i \leftarrow \text{sort_by_objective}(F_i, j)$ ▷ Sort by j'th objective value
 - 10: **for** $k \leftarrow 2$ to $n_i - 1$ **do** ▷ F_i has n_i solutions in it
 - 11: $d(F_i[k]) \leftarrow d(F_i[k]) + (F_i[k+1][j] - F_i[k-1][j])$
 - 12: $d(F_i[1]), d(F_i[n_i]) \leftarrow \text{Max distance among all points in } F_i$
 - 13: **return** d
-

3. METHODOLOGY AND RESULTS

3.2.2 Data

We will be using the data provided by *Qube Research and Technologies (QRT)* on the *Challenge Data* site, under the challenge: *Learning factors for stock market returns prediction* [18].

The data contains (cleaned) daily returns of 50 stocks over a time period of 755 days. For our purposes of experiments on this data we will use values of $(n, k, d, T, S) = (50, 21, 20, 600, 155)$.

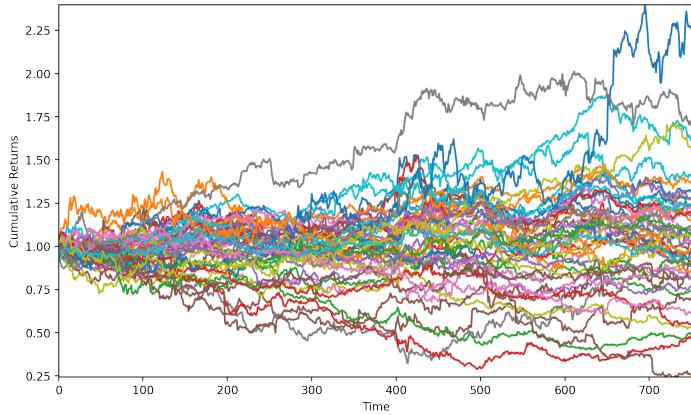


Figure 3.1: Cumulative Returns for Daily Returns Time Series

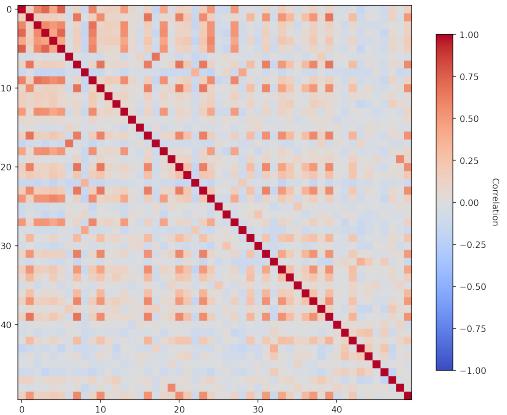


Figure 3.2: Correlation Matrix of Daily Returns

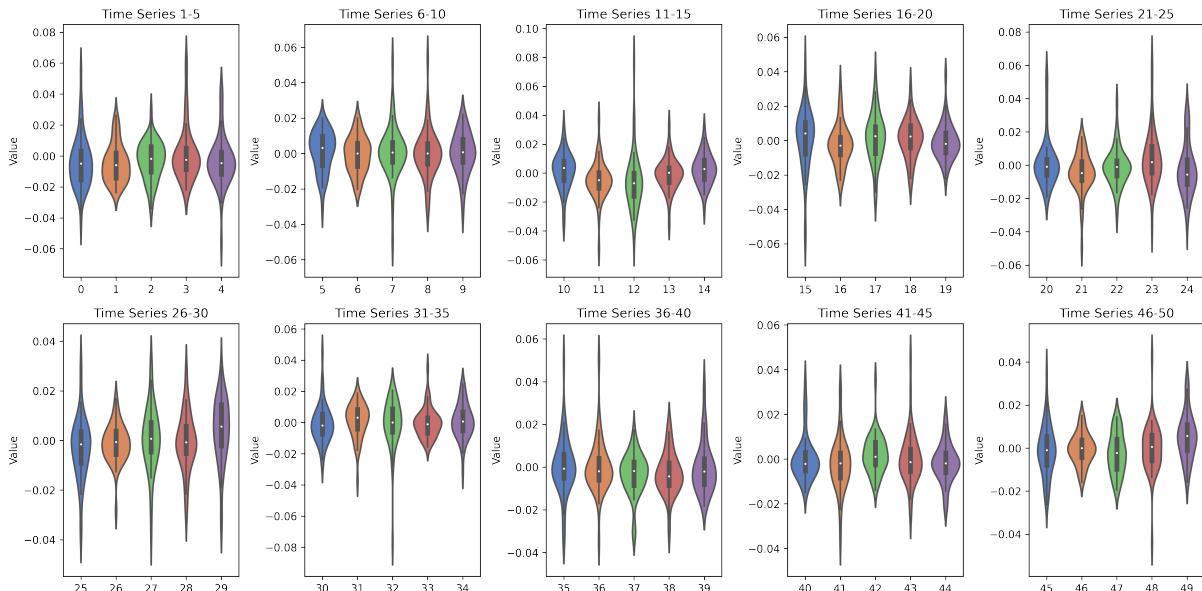


Figure 3.3: Violin Plots for 50 Time Series

3.2 Experimental Results

3.2.3 Results

Since we have two objectives and one important condition, namely $R = \mathcal{R}_{\beta,A}$, $L = \mathcal{L}_{\beta,A}$, and $O = \frac{1}{2}||AA^\top - I||_F^2$, we will use their negative logarithms for the purpose of comparing the methods and deriving the statistics, as L and O are losses. Note that the higher the value of R , $-\log(L)$, and $-\log(O)$, the better the solution/method.

3.2.3.1 Baseline Method

We implemented the baseline method and executed it for $N = 10000$ (i.e., $\mathcal{A}_0[10000]$). For the plots below color of the point indicates its $-\log(L)$ value. We observe that all solutions have a high value of $-\log(L)$, which appears to be bounded due to natural variation in the data. Additionally, the solutions are symmetrically distributed over R , but exhibit a skewed distribution over the values of $-\log(O)$.

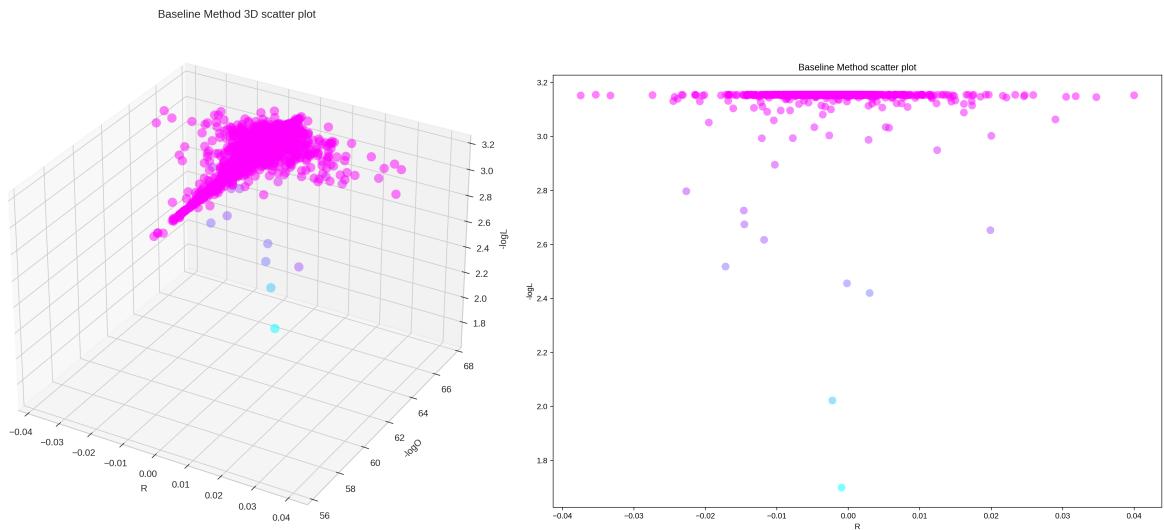


Figure 3.4: 3D Scatter Plot : Baseline Method

Figure 3.5: 2D $(R, -\log(L))$ Scatter Plot : Baseline Method

3.2.3.2 OPI Method : Type 1

To get better statistical result we ran the method multiple times with value of N set to 1000. $\mathcal{A}_1[1000, \mathcal{U}_A^1, \mathcal{U}_\beta^1]$, we are using $\mathcal{U}_A^1[(100, 0.01, 0.3, 0.3, 0.3)]$ to update A and $\mathcal{U}_\beta^1[(100, 0.01, 0.5, 0.5)]$ to update β . We observe that all solution attempts converge to the same point in $(R, -\log(L))$, with initial updates being large but decreasing over time. Additionally, this method is able to overcome the problem of local minima in one dimension through the presence of better minima in another dimension.

3. METHODOLOGY AND RESULTS

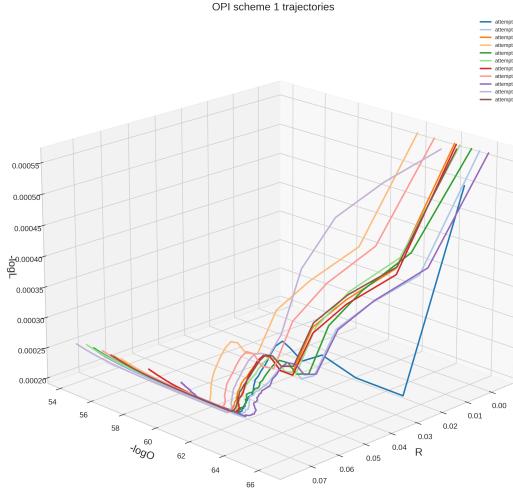


Figure 3.6: 3D Trajectory Line Plot : OPI Method Type 1

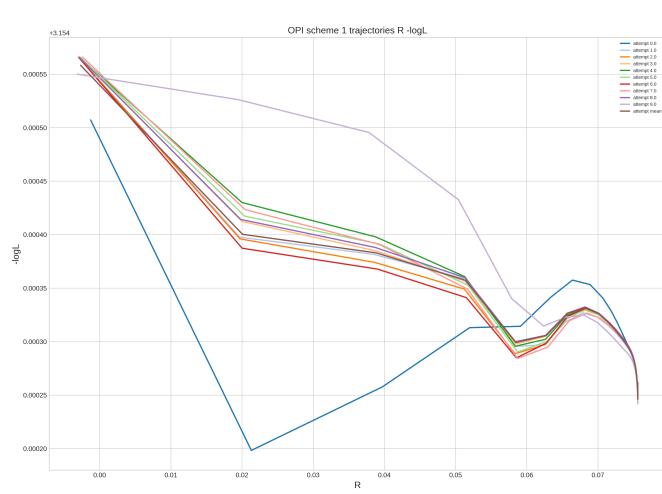


Figure 3.7: 2D ($R, -\log(L)$) Trajectory Line Plot : OPI Method Type 1

3.2.3.3 OPI Method : Type 2

To get better statistical result we ran the method multiple times with $N = 1000$. This method uses \mathcal{U}_A^1 to update A but $\mathcal{U}_\beta^2[(100, 0.01, 0.5, 0.5, 0.01)]$ to update β . We observe that this method updates the solutions non-smoothly, as it relies on regression to update β , which may result in the next update taking the value of $(R, -\log(L), -\log(O))$ to a distant point.

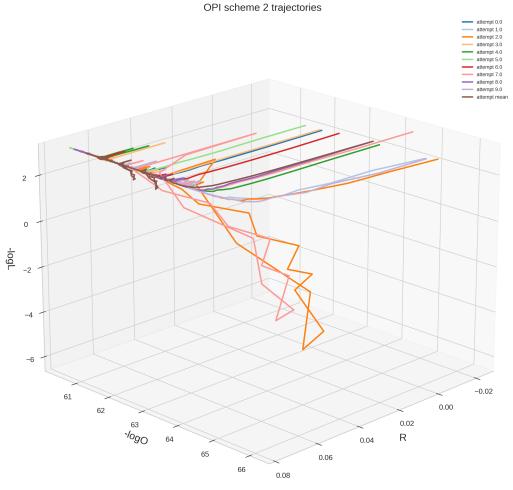


Figure 3.8: 3D Trajectory Line Plot : OPI Method Type 2

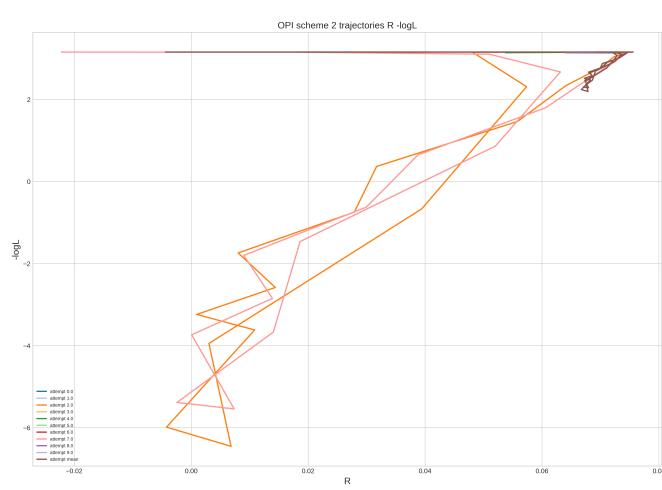


Figure 3.9: 2D ($R, -\log(L)$) Trajectory Line Plot : OPI Method Type 2

3.2.3.4 DOI Method : Type 1

Here we use same internal methods as of OPI Type 1, but since this method relaxes $A \in \mathcal{O}_d^k$ to $A \in \mathbb{R}^{d \times k}$ we use $\mathcal{U}_O^1[100, 0.01, 0.3, 0.3, 0.3]$ to bring it back in \mathcal{O}_d^k . Same as

3.2 Experimental Results

before to get better statistical result we ran the method multiple times with $N = 1000$. This method converges to a point with decreasing update size, but the generated solutions do not consider the condition $A \in \mathcal{O}_d^k$, resulting in non-smooth updates.

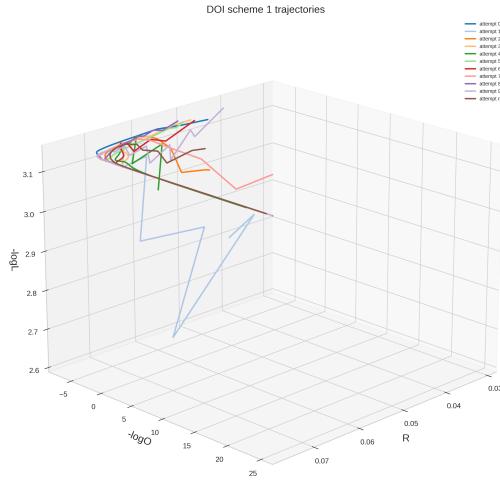


Figure 3.10: 3D Trajectory Line Plot : DOI Method Type 1

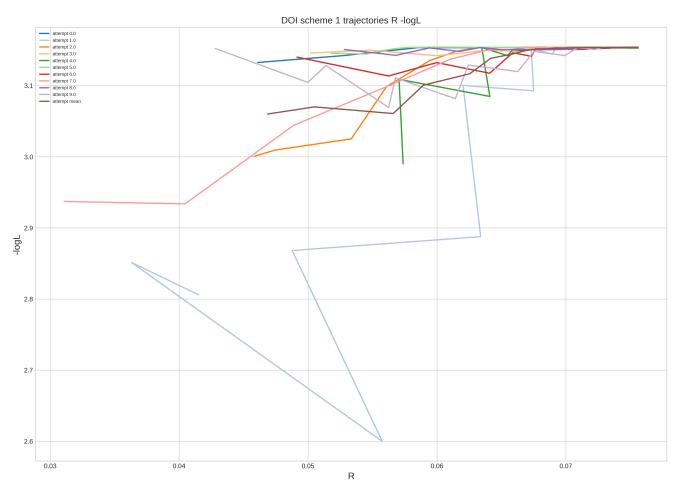


Figure 3.11: 2D ($R, -\log(L)$) Trajectory Line Plot : DOI Method Type 1

3.2.3.5 DOI Method : Type 2

It is same as DOI Type 1, but uses OPI Type 2 in place of OPI Type 1 and here too to get better statistical result we ran the method multiple times with $N = 1000$. This method has issues similar to DOI type 1 and OPI type 2, as it is not very smooth and the next update may not be nearby. But in $(R, -\log(O))$ space, it is somewhat smooth, with little influence from regression to obtain β , which minimizes L .

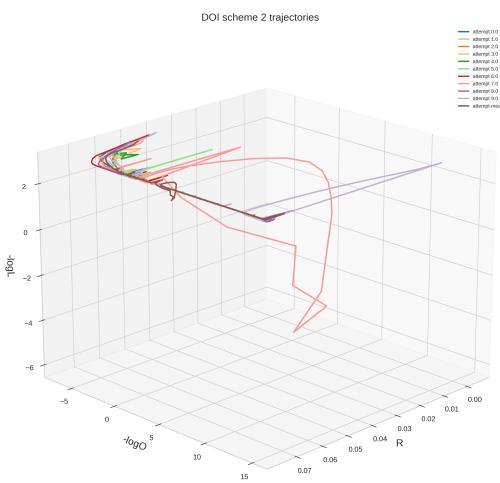


Figure 3.12: 3D Trajectory Line Plot : DOI Method Type 2

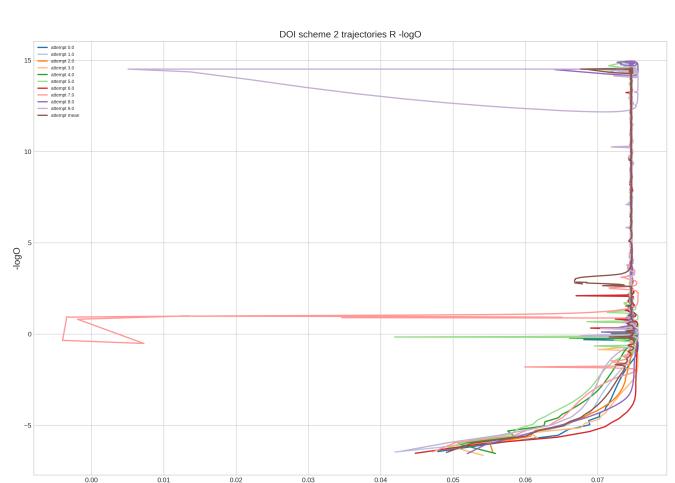


Figure 3.13: 2D ($R, -\log(O)$) Trajectory Line Plot : DOI Method Type 2

3. METHODOLOGY AND RESULTS

3.2.3.6 No Preference Method (NPM) : Type 1

A learning rate of 0.01 is used for A and β , with 1000 steps and $NPM_1[(0.01, 0.01)]$. This method is observed to be very smooth, with all attempts converging to a point, and the update size decreasing over time.

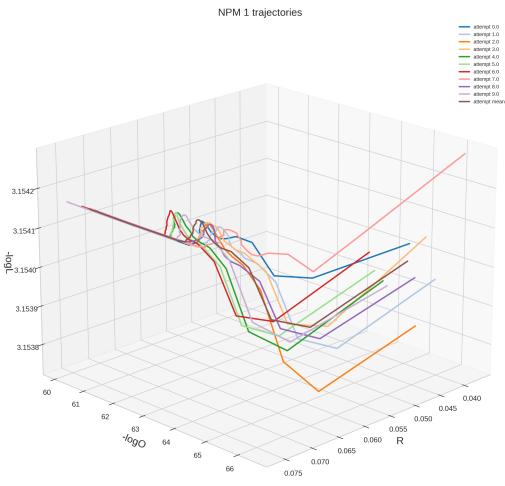


Figure 3.14: 3D Trajectory Line Plot : NPM Type 1

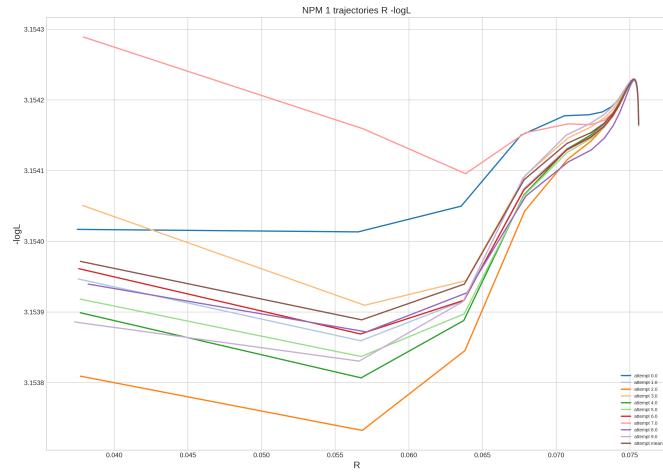


Figure 3.15: 2D (R , $-\log(L)$) Trajectory Line Plot : NPM Type 1

3.2.3.7 No Preference Method (NPM) : Type 2

The method uses a learning rate of 0.01 for both A and β , with 1000 steps and $NPM_2[(0.01, 0.01)]$. However, it doesn't utilize the orthogonal property and Cayley transform to update A . As seen in the figures, the method is smooth and converges with decreasing update size. However, there is some variation in the convergence point.

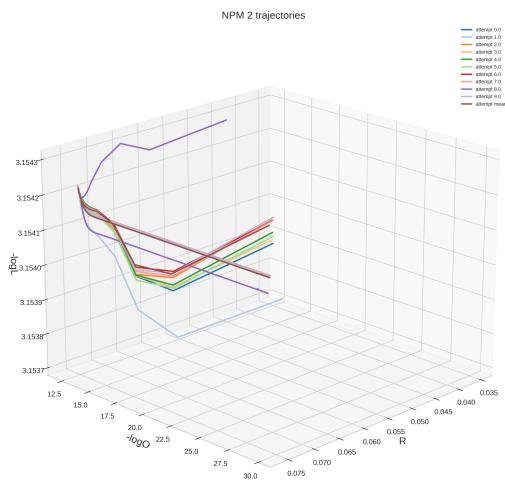


Figure 3.16: 3D Trajectory Line Plot : NPM Type 2

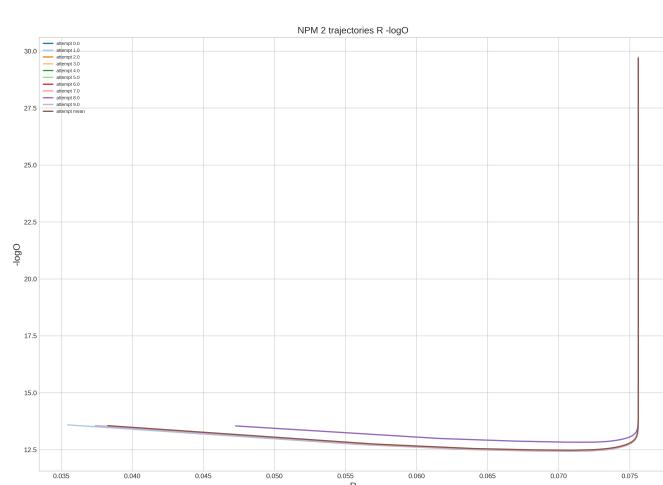


Figure 3.17: 2D (R , $-\log(O)$) Trajectory Line Plot :NPM Type 2

3.2 Experimental Results

3.2.3.8 NSGA-II

Type 1 : In this NSGA-II type we dont use regression for updating β and the parameters of this are $\text{NSGA-II}(100, 500, 1, 0.2, \mathcal{C}[\text{No Regression}, 0.01], \mathcal{M}[0.01])$, that is with population size of 500 and number of generations 100. We observe that, except for the first generation, all the points lie on a non-dominant front and the performance appears to be improving steadily with each generation.

Type 2: This uses regression for updating β . We can see that there are multiple non-dominant fronts and as we reach convergence we have only 1 non-dominant front.

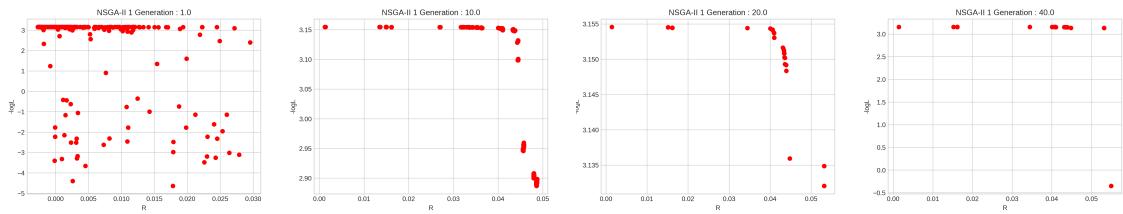


Figure 3.18: Solutions scatter plot of NSGA-II Type 1 for generations 1,10,20,40

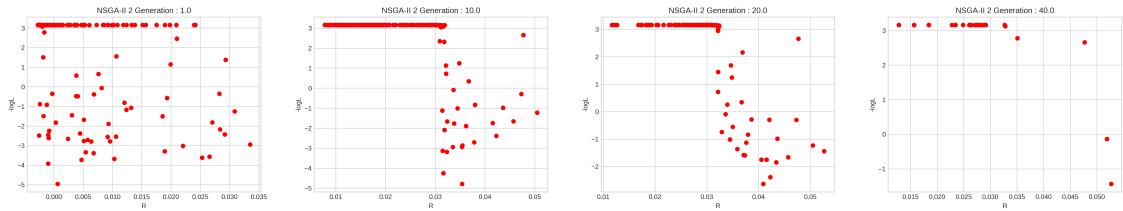


Figure 3.19: Solutions scatter plot of NSGA-II Type 2 for generations 1,10,20,40

3.2.3.9 MOPSO

We executed MOPSO(500, 100, 0.95, 0.9, 0.5, 0.2, 0.1, 0.006), population size of 500 and 100 time iterations. We can observe that the solutions has swarm behavior, with a drift towards higher values of R , as a tradeoff, their $-\log(L)$ value decreases.

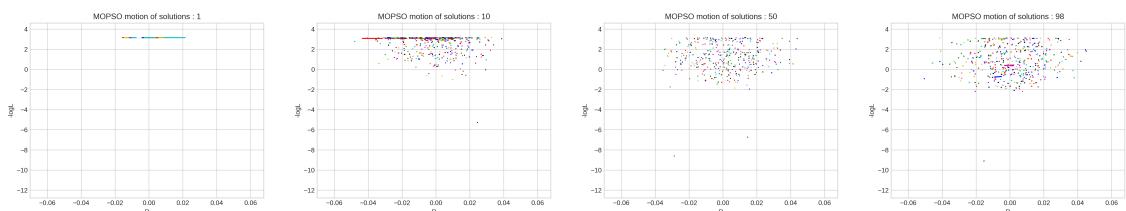


Figure 3.20: Solutions plot of MOPSO for time values of 1,10,50,98

3. METHODOLOGY AND RESULTS

3.2.3.10 Comparison Table

Objective	Statistic	Baseline	OPI		DOI		NPM		NSGA-II		MOPSO
			Type 1	Type 2	Type 1	Type 2	Type 1	Type 2	Type 1	Type 2	
R	mean (1e-2)	-0.272	7.53	7.43	7.53	7.43	7.55	7.55	1.90	2.63	0.00266
	std (1e-3)	2.36	3.43	4.48	1.85	3.23	1.42	1.41	9.38	7.60	16.2
	min (1e-2)	-3.74	-0.31	-2.22	3.11	-0.40	3.73	3.55	-2.35	-1.55	-5.86
	50% (1e-2)	-0.275	7.56	7.47	7.56	7.47	7.56	7.56	1.62	2.91	-8.96
	max (1e-2)	4.00	7.56	7.56	7.56	7.56	7.56	7.56	5.51	5.93	5.63
$-\log(L)$	mean	3.15	3.15	3.14	3.15	3.15	3.15	3.15	3.12	3.03	1.42
	std (1e-2)	2.51	0.00147	27.6	9.56	19	0.00155	0.00216	35.9	74.6	146
	min	1.70	3.15	-6.45	2.60	-6.37	3.15	3.15	-4.66	-4.96	-11.3
	50%	3.15	3.15	3.15	3.15	3.15	3.1	3.15	3.15	3.15	1.55
	max	3.15	3.15	3.15	3.15	3.15	3.15	3.15	3.15	3.15	3.15
$-\log(O)$	mean	64.46	58.34	61.80	13.82	10.06	61.58	27.22	63.60	64.15	63.38
	std	1.36	2.51	0.829	9.54	5.98	0.75	4.19	0.52	0.37	0.95
	min	56.41	54.09	60.5	-6.56	-6.65	60.06	12.43	56.32	58.47	59.65
	50%	64.68	57.84	61.6	16.21	14.4	61.46	29.01	63.54	64.09	63.42
	max	67.50	66.32	66.61	24.46	14.96	66.42	29.74	66.98	66.88	67.29

Table 3.1: Statistics on solutions generated by various method

3.2.3.11 Discussion

We note that the baseline method provides a basis for comparison with other methods. Methods that have high mean values and low variance values are considered good, as they converge more quickly towards the optimum. When considering the distribution of $-\log(L)$, we observe that all methods, except for MOPSO, have a distribution that is very close to optimal. Additionally, we note that most of the standard deviation in the values of $-\log(L)$ is found in the latter decimals of the values. As a result, there is no significant difference in the first two digits after the decimal point. However, we do observe that MOPSO performs particularly poorly, with a high standard deviation. Additionally, we note that both DOI types, OPI type 2, and both NSGA-II types have slightly higher standard deviations. Regarding R , we note that all methods except NSGA-II and MOPSO achieve the maximum value. However, among the methods that achieve the maximum value, OPI type 1, DOI type 1, and both NPM types achieve it more efficiently, as their 50% value is the same. Moreover, the NPM methods have the lowest variation among them. Regarding $-\log(O)$, we observe that the baseline method has the highest values. We also note that the methods that perform well with higher R generally have lower values of $-\log(O)$.

Chapter 4

Conclusion and Future Work

4.0.0.0.1 In conclusion, we have presented a comprehensive study of solving cond-fMOOP with restrictions on condition number, which generalizes several important problems. We have reviewed various methods to solve MOOPs and discussed important results on the condition number of matrices and functions. We also observed that bounding the condition number of neural networks is a viable defensive mechanism against adversarial attacks. We have formulated some critical questions about cond-fMOOP and derived two theoretical results on the condition number of composition of functions and the relationship between L_2 regularization and absolute condition number. We have then focused on the original problem which motivated the definition of fMOOP i.e. the problem of Linear Factor Model with the condition $AA^\top = I$ and proposed a method that leverages the property $AA^\top = I$ to keep A satisfying this condition while updating it based on a designed update scheme. Our experiments have shown that this property can help the methods to converge smoothly towards a solution.

It would be interesting to conduct further studies focusing on determining what other problems fall into cond-fMOOP category. Developing a class of methods to address this problem will be a significant step forward. Furthermore, addressing any of the three critical questions that we have mentioned can further improve the space of methods for solving the general problem and the problems contained in it. Additionally, future works can also explore other properties formulations of the loss and reward functions to optimize the computation of the solution. Finally, exploring other potential applications of the methods presented in this project beyond the Linear Factor Model problem is another avenue for future research.

References

- [1] Turing, Alan M(1948): *Rounding-off errors in matrix processes*, 1: 287–308.
- [2] Rice, John R(1966): *A theory of condition*, 2: 287–310.
- [3] Pareto, Vilfredo / others u.a.(1971): *Manual of political economy*.
- [4] Haimes, Yacov(1971): *On a bicriterion formulation of the problems of integrated system identification and system optimization*, 3: 296–297.
- [5] Zeleny, Milan(1973): *Compromise programming*.
- [6] Vincent, Thomas L / Grantham, Walter Jervis(1981): *Optimality in parametric systems*(Book).
- [7] Vincent, Thomas L(1983): *Game theory as a design tool*.
- [8] Datta, Biswa Nath(1995): *Numerical linear algebra and applications*. Brooks.
- [9] Athan, Timothy Ward / Papalambros, Panos Y(1996): *A note on weighted criteria methods for compromise solutions in multi-objective optimization*, 2: 155–176.
- [10] Deb, Kalyanmoy / Pratap, Amrit / Agarwal, Sameer / Meyarivan, TAMT(2002): *A fast and elitist multiobjective genetic algorithm: NSGA-II*, 2: 182–197.
- [11] Messac, Achille / Mattson, Christopher A(2002): *Generating well-distributed sets of Pareto points for engineering design using physical programming*, 4: 431–450.
- [12] Coello, Carlos A Coello / Pulido, Gregorio Toscano / Lechuga, Maximino Salazar(2004): *Handling multiple objectives with particle swarm optimization*, 3: 256–279.
- [13] Poli, Riccardo / Kennedy, James / Blackwell, Tim(2007): *Particle swarm optimization*, 1: 33–57.

REFERENCES

- [14] Chehab, Jean Paul / Raydan, Marcos(2008): *Geometrical properties of the Frobenius condition number for positive definite matrices*, 8-9: 2089–2097.
- [15] Maréchal, Pierre / Ye, Jane J(2009): *Optimizing condition numbers*, 2: 935–947.
- [16] Deadman, Edvin / Relton, Samuel D(2016): *Taylor’s theorem for matrix functions with applications to condition number estimation*354–371.
- [17] Gutman, David H / Pena, Javier F(2021): *The condition number of a function relative to a set*, 1: 255–294.
- [18] (QRT), Qube Research Technologies(2022): *Learning factors for stock market returns prediction*, (<https://challengedata.ens.fr/participants/challenges/72/>).
- [19] Higham, Nicholas J(2022): *The Matrix Function Toolbox* (<https://www.mathworks.com/matlabcentral/fileexchange/20820-the-matrix-function-toolbox>).
- [20] Higham, Nicholas J (2008): *Functions of matrices: theory and computation.* , SIAM.
- [21] Hwang, Ching Lai / Masud, Abu Syed Md (1979): *Methods for multiple objective decision making*. Multiple objective decision making—methods and applications. Springer: 21–283.
- [22] Hwang, C L / Masud, Abu Syed Md (2012): *Multiple objective decision making—methods and applications: a state-of-the-art survey.* , Springer Science & Business Media.
- [Li)Li] Li, Xiaodong *A non-dominated sorting particle swarm optimizer for multiobjective optimization.* .
- [24] Miettinen, Kaisa (1998): *No-Preference Methods*. Boston, MA, Springer US: 67–76.
- [25] Miettinen, Kaisa (2012): *Nonlinear multiobjective optimization.* , Springer Science & Business Media.
- [26] Raquel, Carlo R / Naval Jr, Prospero C (2005): *An effective use of crowding distance in multiobjective particle swarm optimization.* .