INCOMPLETE PREFERENCES AND ROBUST SATISFICING IN MULTI-ARMED BANDITS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF ENGINEERING AND SCIENCE
OF BILKENT UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR
THE DEGREE OF
MASTER OF SCIENCE
IN
ELECTRICAL AND ELECTRONICS ENGINEERING

By
Yaşar Cahit Yıldırım
September 2025

Incomplete Preferences and Robust Satisficing in Multi-Armed Bandits
By Yaşar Cahit Yıldırım
September 2025

We certify that we have read this thesis and that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Cem Tekin(Advisor)

Çağın Ararat

Emre Özkan

Approved for the Graduate School of Engineering and Science:

Orhan Arıkan
Director of the Graduate School

ABSTRACT

INCOMPLETE PREFERENCES AND ROBUST SATISFICING IN MULTI-ARMED BANDITS

Yaşar Cahit Yıldırım
M.S. in Electrical and Electronics Engineering
Advisor: Cem Tekin
September 2025

This thesis studies sequential decision-making in multi-armed bandits (MAB) in two settings: when there are multiple objectives to consider and preferences among actions are non-total, and when actions are subject to adversarial perturbations. In the first part, we introduce PaVeBa, a pure-exploration algorithm for vector-valued rewards under incomplete preferences, using cone-induced comparisons instead of scalarizations. We analyze its behavior and validate it empirically against strong baselines. To support research and reproducibility in this area, we also develop VOPy, a Python library that implements PaVeBa and a broader suite of vector-optimization algorithms and tools. In the second part, we investigate robust satisficing under adversarial attacks. Building on existing robust satisficing formulations (*e.g.*, stability radius and fragility-based), we design a Thompson-sampling variant of an algorithm family and demonstrate that it achieves reliable satisficing performance under targeted perturbations.

Keywords: multi-armed bandits, vector optimization, robust satisficing.

ÖZET

ÇOK KOLLU HAYDUTLARDA EKSİK TERCİHLER VE GÜRBÜZ YETERLİLİK

Yaşar Cahit Yıldırım
Elektrik ve Elektronik Mühendisliği, Yüksek Lisans
Tez Danışmanı: Cem Tekin
Eylül 2025

Bu tez, çok kollu haydut (MAB) problemlerinde sıralı karar vermeyi iki farklı bağlamda incelemektedir: birden fazla amacın bulunduğu ve eylemler arasındaki tercihlerin tam olmadığı durumlarda ve eylemlerin hasım saldırılarına maruz kaldığı durumlarda. İlk bölümde, skalerleştirmeler yerine koniler ile temellendirilen karşılaştırmalar kullanan, eksik tercihler altında vektör değerli ödüller için saf keşif algoritması olan PaVeBa'yı sunuyoruz. Algoritmanın davranışını analiz ediyor ve güçlü temel metotlara karşı deneysel olarak doğruluyoruz. Bu alandaki araştırmaları ve tekrar üretilebilirliği desteklemek için, PaVeBa ve yanı sıra daha geniş bir vektör optimizasyonu algoritmaları ve araçları yelpazesini içeren VOPy isimli bir Python kütüphanesi de geliştiriyoruz. İkinci kısımda ise, hasım saldırıları altında gürbüz yeterlilik problemini inceliyoruz. Mevcut gürbüz yeterlilik formülasyonlarını (örneğin, kararlılık yarıçapı ve kırılganlık tabanlı) temel alarak, bir algoritma ailesinin Thompson-örnekleme varyantını tasarlıyor ve hedeflenmiş bozucu saldırılar altında güvenilir bir yeterlilik performansı elde ettiğini gösteriyoruz.

Anahtar sözcükler: çok kollu haydutlar, vektör eniyileme, gürbüz yeterlilik.

Acknowledgement

I am grateful to my advisor, Assoc. Prof. Cem Tekin, for his invaluable guidance and support, and most importantly for his trust throughout my studies, which helped me grow both as a researcher and as a person. His confidence in me played a crucial role in the completion of this dissertation. I would also like to sincerely thank Assoc. Prof. Çağın Ararat, whom I hold in the utmost respect as a scholar, whose rigor, clarity, and dedication have been an inspiration to me. I would also like to express my gratitude to Assoc. Prof. Emre Özkan for serving on my jury and giving feedback on my thesis.

I am deeply thankful to my family–my father, my mother, and my brother– for their understanding, and unconditional love and support. Through the loud and the quiet, they have been a constant source of "truth" in my life.

I thank Artun, Efe, and Onat for our collaborations. Furthermore, I would like to thank them and Enes for their delightful personalities and our companionship throughout this journey. They all left their traces on my thinking, and our discourses on (very) diverse subjects have been the source of expedition to brilliant minds for me. Special thanks to Onat and Artun, whose empathy and encouragement were invaluable at times. I'd also like to thank Ali and the other lab members for creating a warm work environment.

I want to thank Dr. Murat Kumru for his encouragement and kindness.

I am forever indebted to my partner, Zeynep, for the purpose and joy she brings to my life.

This work was supported by Türk Telekom as part of the 5G and Beyond Joint Graduate Support Programme, coordinated by the Information and Communication Technologies Authority, and in part by the Scientific and Technological Research Council of Türkiye under Grants 121E159 and 124E065.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

An important subset of machine learning involves what we refer to as sequential decision-making under uncertainty. This concept covers the scenarios where a learner repeatedly interacts with an environment: it acts within it, observes some uncertain phenomena, and updates its strategy before repeating the process. If the environment itself does not evolve dependent on the learner's actions, and the set of actions the learner can take is fixed, we call this distilled problem a *multi-armed bandit* (MAB) problem. The available actions the learner can take are the *arms*.

The name "multi-armed bandit" comes from imagining a row of slot machines, each with its own unknown payout, or *reward*, distribution. The central challenge is the *exploration–exploitation* dilemma: on the one hand, the learner must explore to reduce uncertainty about the mean rewards of different arms (sampling arms whose estimates have high variance), while on the other hand, it wants to exploit what it has learned so far by favoring the arm with the highest estimated mean. This tension defines the core difficulty of MAB problems, and almost every algorithm in this space can be seen as a way of balancing these two objectives.

At each round $t \in \mathbb{N}$, the learner chooses a single arm $x_t \in \mathcal{X}$ (or a batch of arms)

from the action set $\mathcal{X}$, observes reward $y(x_t) = f(x_t) + \eta_t$ where η_t is the observation noise, and gradually learns reward statistics of arms. In MAB literature, learners aim either for *best-arm identification* or for *regret minimization*. Regret at round t is defined as

$$r_t = f(x^*) - f(x_t),$$

i.e., the missed reward between the action taken, $x_t \in \mathcal{X}$, and the optimal action, $x^* \in \mathcal{X}$.

Many real-world problems, however, involve richer structures than vanilla MABs with simple noisy scalar rewards. Some settings require vector-valued rewards, as in *multi-objective optimization*. In others, those vector-valued outcomes come with only partial comparability, as in *vector optimization*. Another line of work considers settings where the learner has only limited control over which arms can be played, as in adversarial perturbations.

This thesis focuses on two such extensions to the bandit framework: vector optimization with incomplete preferences, and *robust satisficing* under adversarial attacks. Both extensions share the core concept of sequential learning, but they address different complexities that arise during real-world decision-making.

The first part of the thesis will be on vector optimization in MABs, where each observation yields a reward vector of multiple objectives, and user preferences are encoded through a polyhedral cone. In this context, we propose Pareto Vector Bandits (PaVeBa), an adaptive elimination algorithm for pure exploration of arms with noisy vector-valued rewards. We develop Gaussian process (GP) based heuristics for this algorithm and include extensive experiments for all versions. Furthermore, to facilitate the advancement of the vector optimization field by providing practical solutions to practitioners and easing algorithm development for researchers, we develop VOPy, an open-source Python framework for black-box vector optimization. VOpy provides preference comparisons through cones, various modeling options, and modular implementations of several algorithms, including PaVeBa and VOGP [1].

The second part of the thesis addresses robust satisficing (RS) in bandits, specifically in Gaussian process (GP) bandits and in the context of adversarial attacks. The adversarial setting is concerned with cases where learned policies may fail due to adversarial perturbations of selected arms, causing input uncertainty. In this context, we investigated the adversarial perturbations of arms following [2], and proposed Thompson sampling based family of algorithms. We empirically compare this algorithm family with robust optimization (RO) methods on both synthetic and real-world problems.

The remainder of this thesis is organized as follows. Chapter 2 provides the background and preliminary discussions on vector optimization, Bayesian optimization and Gaussian processes, and robust satisficing. Chapter 3 covers our contributions to vector optimization in MABs, including the PaVeBa algorithm and its experimental evaluation. Chapter 4 introduces the VOPy library, designed to foster research in vector optimization. Finally, Chapter 5 focuses on our contributions to robust satisficing bandits in adversarial settings, presenting an empirical algorithm family and analyzing its performance.

Chapter 2

Background and Preliminaries

2.1 Vector Optimization

Consider the MAB problems where each arm $x \in \mathcal{X}$ has a corresponding mean reward vector $\boldsymbol{f}(x) \in \mathbb{R}^D$, where $D \in \mathbb{N} \coloneqq \{1, 2, \ldots\}$. Notice that in this setting, as there are multiple objectives, comparing two arms is not straightforward. One could scalarize the reward vector, for example, by taking the dot product $\boldsymbol{w}^T \cdot \boldsymbol{f}(x)$ with a weight vector $\boldsymbol{w} \in \mathbb{R}^D$, which would give a total order, but it's not an easy task choosing $\boldsymbol{w}$.

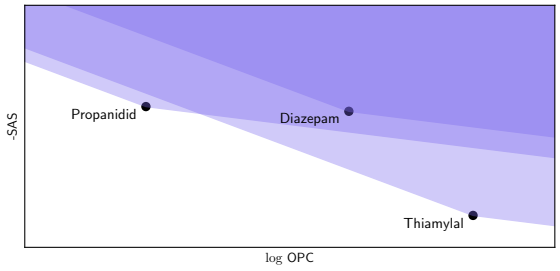


Figure 2.1: Three anesthetics ordered by a polyhedral cone. Among them, Diazepam is the only Pareto optimal drug.

Instead, one can use the strict partial order $<$ for comparing arms, an approach called *multi-objective optimization* (MOO). In this case, an arm is said to dominate another if it is at least as good in all objectives and strictly better in one [3]. However, note that this procedure would not necessarily yield a single arm, as some vectors are not comparable. To see this, consider two reward vectors $\mu = [1\ 0]^T$ and $\nu = [0\ 1]^T$ and notice that neither $\mu < \nu$ nor $\nu < \mu$. This leads to the identification of the *Pareto set*, where instead of a single best arm, we obtain a set of best arms. To that end, some work on bandit literature focused on the identification of the Pareto set of arms. Noteworthy contributions include algorithms for (ϵ, δ)-*PAC Pareto set* identification [4] defined similar to its single-objective case, exploration using Gaussian processes in large datasets [5, 6], feasible arm identification [7], and algorithms using information-theoretic heuristics [8–10]. Additionally, regret minimization in multi-objective bandits is explored by [11] and [12], while multi-objective reinforcement learning has been delved into by [13] and [14].

While multi-objective methods address vector-valued rewards, they employ a specific domination concept, as we mentioned, which we will refer to as the standard componentwise order hereafter. The field of *vector optimization* [15, 16] generalizes this by offering a more flexible dominance notion, a partial order $\preceq_C$ induced by a cone C defined as $\boldsymbol{\mu} \preceq_C \boldsymbol{\nu}$ if and only if (iff) $\boldsymbol{\nu} - \boldsymbol{\mu} \in C$. The flexibility of the framework under different cones has led to diverse real-world applications, as the choice of C allows users to reflect their incomplete (non-total) preferences, where choosing C as the positive orthant yields the MOO approach outlined above. To exemplify this flexibility, consider the anesthetic design problem, where both the *octanol-water partition coefficient* (OPC) and the *synthetic accessibility score* (SAS) are essential metrics. The former has been shown to correlate well with anesthetic efficacy [17, 18], while the latter predicts the difficulty of synthesizing a molecule [19]. To illustrate what's missing if standard componentwise order is used, Figure 1 shows the negative of SAS (ease of synthesis) and $\log \mathrm{OPC}$ of three different anesthetics [20, 21]. Note that all three drugs are Pareto optimal in the standard sense. However, Propanidid is at an absolute disadvantage in practice compared to Diazepam. That's because ease of synthesis does not mean

much unless the drug is actually useful, and Diazepam is only slightly harder to synthesize but vastly more effective as an anesthetic. By using a larger cone than the standard one, as shown in the Figure 2.1, the optimal drug among the three can be selected as Diazepam, ruling out other drugs that rely on extreme trade-offs between accessibility and effectiveness.

2.2 Bayesian Optimization

Bayesian optimization (BO) [22, 23] is a sequential approach to optimizing an objective function $f(x)$ defined over a feasible set $\mathcal{X}$. It is used in cases where f is expensive to evaluate and its analytic form is unknown. In such cases, one cannot afford to take a large number of samples as required by evolutionary or random search methods, nor can one rely on gradient-based algorithms since derivatives or structural properties like concavity are unavailable.

In most of the BO literature, the domain $\mathcal{X}$ is taken to be continuous, but in the multi-armed bandit setting, we often assume it is finite, corresponding to a fixed set of arms. The central idea of BO is to build a probabilistic *surrogate model* of the unknown function that is updated as new observations are collected, and to guide decisions through an *acquisition function*. The surrogate captures both what is currently known about f and the uncertainty that remains, while the acquisition function uses this information to balance the exploration of uncertain regions with the exploitation of regions that already look promising. Among different surrogate models, Gaussian processes (GPs) are ubiquitous, to the point where BO became almost synonymous with GPs. While we focus here in this preliminary chapter on the single-objective case $f : \mathcal{X} \to \mathbb{R}$, the same principles extend to the multi-objective setting $\boldsymbol{f} : \mathcal{X} \to \mathbb{R}^D$, where multi-output GPs and suitably adapted acquisition functions are employed.

2.2.1 Gaussian Processes

A standard surrogate model for the unknown function is to use a Gaussian process (GP) prior.

Definition 1. *[24, Definition 2.1] A* Gaussian process *is a collection of random variables, such that any finite subset of which follows a joint Gaussian distribution.*

Similar to how a Gaussian random variable is completely specified by its mean and variance, a GP prior is completely specified by its mean function $\mu(x)$ and covariance function $k(x, x')$, where

$$\begin{aligned} \mu(x) &= \mathbb{E}\left[f(x)\right], \\ k(x, x') &= \mathbb{E}\left[(f(x) - \mu(x))(f(x') - \mu(x'))\right]. \end{aligned}$$

When we condition a GP on observed data points $\{(x_i, y_i)\}_{i=1}^{t}$ with Gaussian noise, the posterior at any new point x remains Gaussian with predictive mean and variance

$$\begin{aligned} \mu_t(x) &= k_t(x)^\top \left(\boldsymbol{\Sigma}_t + \sigma^2 \boldsymbol{I}\right)^{-1} y_{1:t}, \\ \sigma_t^2(x) &= k(x, x) - k_t(x)^\top \left(\boldsymbol{\Sigma}_t + \sigma^2 \boldsymbol{I}\right)^{-1} k_t(x), \end{aligned} \tag{2.1}$$

where $\boldsymbol{\Sigma}_t$ is the covariance matrix on observed points and $k_t(x)$ is the covariance vector between x and the observed points. Notice the $\sigma^2 \boldsymbol{I}$ term in the equations, which accounts for the observation noise $\eta \sim \mathcal{N}(0, \sigma^2)$ from $y(x) = f(x) + \eta$.

Covariance functions. The choice of covariance function (or kernel) $k(x, x')$ plays a central role in Gaussian processes, as it encodes prior assumptions about the smoothness and structure of the unknown function. Different kernels can also be combined by addition or multiplication to represent more complex behaviors [25, Chapter 2]. In practice, kernel hyperparameters are usually estimated from data by maximizing the marginal likelihood of the GP.

The posterior mean and variance allow us to define acquisition functions that guide the selection of the next query point. Two widely used examples are GP-UCB [26] and Thompson sampling [27, 28].

GP-UCB. The upper confidence bound (UCB) rule encourages optimism in the face of uncertainty by augmenting the mean with a scaled standard deviation [26]:

$$x_t \in \arg\max_{x \in \mathcal{X}} \ \mu_{t-1}(x) + \sqrt{\beta_t}\, \sigma_{t-1}(x),$$

where β_t controls the exploration–exploitation trade-off.

Thompson Sampling. Thompson sampling [27, 28] selects the next point by sampling a function $\tilde{f}$ from the GP posterior and then choosing

$$x_t \in \arg\max_{x \in \mathcal{X}} \ \tilde{f}(x).$$

This randomized policy implicitly balances exploration and exploitation by occasionally sampling from regions of high uncertainty.

2.3 Robust Satisficing

Decision making often involves dealing with ambiguity, or *uncertainty* in the Knightian sense [29]. Robust optimization (RO) is a widely accepted framework handling such uncertainty: rather than assuming exact knowledge of model parameters, the goal is to find solutions that remain effective across a range of scenarios. In this framework, the agent maximizes utility under the worst case within a defined uncertainty set, *i.e.*, a set of values that model parameters may take [30]. So, in RO, the goal is different than the standard formulation of optimization which is

$$\arg\max_{x} f(x),$$

and it is instead

$$\arg\max_{x} \min_{z \in \Delta} f(x, z)$$

where $x \in \mathcal{X}, z \in \mathcal{Z}$ refer to different model parameters, x is a decision variable, z is an uncertain parameter, and $\Delta \subseteq \mathcal{Z}$ is an uncertainty set. In general, uncertainty sets are specified around a nominal parameter value $\hat{z} \in \mathcal{Z}$. This nominal does not necessarily represent the "true" parameter, but serves as a

baseline from which the deviations are measured. Formally, we can write an uncertainty set with radius ϵ as

$$\Delta_\epsilon(\hat{z}) = \{z \in \mathcal{Z} : d(z, \hat{z}) \leq \epsilon\}$$

where d is a distance function on $\mathcal{Z}$. We take d to be a pseudometric in this thesis, though in other settings it may be specialized to a metric (*e.g.*, Euclidean distance) or generalized to a divergence (*e.g.*, KL divergence). In [31], the uncertain parameter z is a contextual variable where they only have an uncertain distribution on it, and they work using expectations. Different from this distributional setting, [2] work in an adversarial setting, where the uncertainty is on x and z represent the adversarial perturbation on x.

Satisficing. Decision theory often assumes that an agent's goal is to maximize expected utility. However, *satisficing*, a concept introduced and a term coined by Nobel Laureate[1] in Economics Herbert Simon ([32, 33]), offers a more general perspective: instead of seeking the best possible outcome, an agent may aim at a solution that is "good enough", *i.e.*, one that meets or exceeds a specified threshold τ. This principle reflects decision-making under bounded rationality, where information and/or computation may be limited. Satisficing has also been considered in MAB problems and regret minimization ([34–37]).

Then, when considered in the scope of robustness, the satisficing paradigm becomes a convenient tool for handling uncertainties. The focus in *robust satisficing* (RS) shifts from choosing an action that maximizes a worst-case utility as in RO, to an action that achieves at least a threshold τ of utility across a range of possible model parameters [38].

2.3.1 Robust Satisficing Formulations

Since RS is a paradigm of decision-making, one can formulate different approaches to achieve it. Here, we will introduce two known formulations of RS, and a novel

[1]Trivia: So far, only two people have won both the Nobel Prize and the A.M. Turing Award. One is Herbert Simon, and other is Geoffrey Hinton.

approach proposed in [2] that unifies them, to find the robust satisficing action $x^{\mathrm{RS}} \in \mathcal{X}$.

The first formulation we consider stems from the notion of *stability radius*, and it is used in RS literature [39] as well as in information-gap decision theory [40]. We try to find a solution that maximizes a stability radius. Stability radius is defined by the maximal radius for the uncertainty set around a nominal $\hat{z}$ so that all elements of this set achieve the predefined utility threshold τ, hence the RS solution is given by solving

$$x^{\mathrm{RS}} = \arg\max_{x} \epsilon_\tau(x, \hat{z}) \tag{2.2}$$

where

$$\epsilon_\tau(x, \hat{z}) \coloneqq \max \epsilon \;\; \text{s.t. } f(x, z) \geq \tau, \; \forall z \in \Delta_\epsilon(\hat{z}), \; \epsilon \geq 0 \tag{2.3}$$

and where ϵ is denoted the stability radius.

The second formulation we will examine is from [38], which forms RS as a *fragility* minimization problem. Concretely, the RS solution is given by solving

$$x^{\mathrm{RS}} \coloneqq \arg\min_{x} \kappa_\tau(x, \hat{z}) \tag{2.4}$$

where

$$\kappa_\tau(x, \hat{z}) \coloneqq \min k \;\; \text{s.t. } f(x, z) \geq \tau - k d(z, \hat{z}), \; \forall z \in \Delta_\infty(\hat{z}), \; k \geq 0, \tag{2.5}$$

and where the fragility k of an action is defined as the minimum amount of suboptimality that can be incurred per unit of shift in the uncertain variable, for all possible shifts $\Delta_\infty(\hat{z})$. Note that $\Delta_\infty(\hat{z})$ is quite an unusual notation since it is precisely equal to $\mathcal{Z}$. We gave the formulation in this notation just to make the two formulations look similar, and the next couple of paragraphs, as well as Chapter 5 would clarify why.

We note that, since the stability radius-based formulation in Equation 2.2 only guarantees satisficing within a bounded neighborhood of $\Delta_\epsilon(\hat{z})$ if the stability radius around the RS action is ϵ, it can be seen as a local robustness model. In contrast, the fragility based formulation in Equation 2.4 considers all possible values of $z \in \mathcal{Z}$, making it a global robustness model.

In [2], a new formulation for RS, called RS-G (for RS-General), is proposed to unify the stability radius and fragility formulations. RS-G is formulated similarly to the fragility formulation, where instead of fragility, they define a p-fragility. The RS solution is found by solving

$$x^{\mathrm{RS}} \coloneqq \arg\min_{x} \kappa_{\tau,p}(x, \hat{z}) \tag{2.6}$$

where p-fragility $\kappa_{\tau,p}(x, \hat{z})$ is defined as

$$\kappa_{\tau,p}(x, \hat{z}) \coloneqq \min k \;\; \text{s.t. } f(x, z) \geq \tau - [kd(z, \hat{z})]^p \,, \;\; \forall z \in \Delta_\infty(\hat{z}), \;\; k \geq 0. \tag{2.7}$$

It is trivial to see that setting $p = 1$ gives us the fragility based formulation. In [2, Proposition 4.1], they prove that having $p = \infty$ also gives the same RS solution as stability radius formulation. Then, the RS-G formulation provides a family of formulations that governs previous attempts at formulizing robust satisficing approach, parameterized with a value p.

Chapter 3

Pure Exploration in Vector Bandits

Pure exploration aims to find the optimal arms through repeated interaction with the environment. In this setting, one typically fixes an error upper bound for identification and tries to minimize the number of samples required. This setup is known as the *fixed confidence setting* [41]. Notable methods here include Exponential Gap Elimination [41] and the Track-and-Stop strategy [42]. A related concept is the (ϵ, δ)-probably approximately correct (PAC) best-arm identification proposed by [43], where the goal is to identify an ϵ-optimal arm with confidence $1 - \delta$. When $\epsilon = 0$, this reduces to the fixed confidence case.

Most existing pure exploration work has concentrated on scalar rewards, but in practice, many problems involve multiple objectives that must be considered simultaneously. This naturally motivates the use of vector optimization. Recently, [44] examined this idea in a bandit setting with sequential noisy observations. They provided important insights and lower bounds for (ϵ, δ)-PAC exploration but left open the design of an algorithm that could achieve these bounds efficiently.

To fill this gap, we study bandits with vector-valued rewards under cone-based orderings, following the vector optimization approach adopted by [15] and [16]. We propose the *Pareto Vector Bandits* (PaVeBa) method. PaVeBa, as shown in [45],

nearly matches the lower bounds established by [44] while using a stricter success condition for (ϵ, δ)-PAC Pareto identification. We present convex optimization formulations to handle certain computations required for Pareto identification using ellipsoidal confidence regions from multi-objective Gaussian processes with correlated outputs.

We then compare PaVeBa against the Naïve Elimination algorithm of [44] and state-of-the-art multi-objective optimization methods. Our results show that PaVeBa and its heuristic variants outperform other methods in vector optimization, while delivering a multi-objective optimization performance on par with established algorithms such as ϵ-PAL [5], MESMO [9], and JES [10].

3.1 Problem Definition

Consider a finite set of arms $\mathcal{X} \subset \mathbb{R}^K$. For each arm $x \in \mathcal{X}$, the corresponding mean reward is given by the vector $\boldsymbol{f}(x) \in \mathbb{R}^D$, where $D \in \mathbb{N} \coloneqq \{1, 2, \ldots\}$. We use $\|\cdot\|_2$ to denote the Euclidean norm on $\mathbb{R}^D$ and $B(\boldsymbol{\mu}, r)$ to denote the closed ball in $\mathbb{R}^D$ whose center is $\boldsymbol{\mu} \in \mathbb{R}^D$ and radius is $r \geq 0$ with respect to $\|\cdot\|_2$. For $\boldsymbol{\mu} \in \mathbb{R}^D$, the distance to a nonempty set $A \subseteq \mathbb{R}^D$ is defined as $d(\boldsymbol{\mu}, A) \coloneqq \inf_{\boldsymbol{\nu} \in A} \|\boldsymbol{\mu} - \boldsymbol{\nu}\|_2$. Let $t \in \mathbb{N}$ denote some arbitrary index for rounds of sequential evaluations. Sampling arm $x \in \mathcal{X}$ at round t, gives the random vector $\boldsymbol{y}_t(x) = \boldsymbol{f}(x) + \boldsymbol{\eta}_t(x)$ as observation, where $\boldsymbol{\eta}_t(x)$ is the random noise vector at round t for arm x. Note that the notation is designed to accommodate batch observations. We assume that the noise family $(\boldsymbol{\eta}_t(x))_{x \in \mathcal{X}, t \in \mathbb{N}}$ consists of independent norm-subgaussian random vectors with a common parameter $\sigma > 0$. Specifically,

$$\mathbb{P}\{\|\boldsymbol{\eta}_t(x)\|_2 \geq u\} \leq 2e^{-\frac{u^2}{2\sigma^2}}$$

for each $u \geq 0$ [46, Definition 3]. For each $x \in \mathcal{X}$, let $N_t(x)$ denote the set of rounds at which arm x is sampled by round t, and let $n_t(x) = |N_t(x)|$.

Since we work with multi-dimensional rewards, we need a notion of ranking between them. Following [44], we use a convex polyhedral ordering cone C to

partially order the reward vectors. We define $C \coloneqq \{\boldsymbol{\mu} \in \mathbb{R}^D \mid \boldsymbol{W}\boldsymbol{\mu} \geq \mathbf{0}\}$, where $\boldsymbol{W}$ is an $N \times D$ real matrix for some $N \in \mathbb{N}$ with rows $\boldsymbol{w}_1^\mathsf{T}, \ldots, \boldsymbol{w}_N^\mathsf{T}$. We assume that $\boldsymbol{W}$ has full row rank and $\|\boldsymbol{w}_n\|_2 = 1$ for each $n \in [N] \coloneqq \{1, \ldots, N\}$. For each $n \in [N]$, we also define the constant $\alpha_n \coloneqq \sup_{\boldsymbol{u} \in B(\mathbf{0},1) \cap C} \boldsymbol{w}_n^\mathsf{T}\boldsymbol{u} \in (0, 1]$. The interior of C is $\operatorname{int}(C) = \{\boldsymbol{\mu} \in \mathbb{R}^D \mid \boldsymbol{W}\boldsymbol{\mu} > \mathbf{0}\}$ and the boundary of C is $\operatorname{bd}(C) = \{\boldsymbol{\mu} \in C \mid \exists n \in [N] \colon \boldsymbol{w}_n^\mathsf{T}\boldsymbol{\mu} = 0\}$. We then define a partial order $\preceq_C$ via $\boldsymbol{\mu} \preceq_C \boldsymbol{\nu}$ iff $\boldsymbol{\nu} - \boldsymbol{\mu} \in C$ for each $\boldsymbol{\mu}, \boldsymbol{\nu} \in \mathbb{R}^D$. In this case, we say that $\boldsymbol{\mu}$ is *weakly dominated* by $\boldsymbol{\nu}$. Similarly, we say that $\boldsymbol{\mu}$ is *strongly dominated* by $\boldsymbol{\nu}$, denoted using the strict partial order $\prec_C$ as $\boldsymbol{\mu} \prec_C \boldsymbol{\nu}$, iff $\boldsymbol{\mu} - \boldsymbol{\nu} \in \operatorname{int}(C)$.

We focus on pure exploration, where the goal is to find the set of "optimal" arms. In the vector bandits problem, this corresponds to maximizing $\boldsymbol{f}$ over $\mathcal{X}$ with respect to the partial order $\preceq_C$, and we want to do it with as few samples as possible. In other words, we want to find the set P^* of all arms containing the maximal elements from $\{\boldsymbol{f}(x) \mid x \in \mathcal{X}\}$ with respect to $\preceq_C$, *i.e.*,

$$P^* = \{x \in \mathcal{X} \mid \nexists y \in \mathcal{X} \setminus \{x\} \colon \boldsymbol{f}(x) \preceq_C \boldsymbol{f}(y)\},$$

which is the *Pareto set*.

To use while identifying the Pareto set P^*, we employ two functions defined by [44], which extend the multi-objective definitions from [4]. We further generalize the Definitions in [44], where these functions are defined on the set of arms; we define them on arbitrary vectors $\boldsymbol{\mu}, \boldsymbol{\nu} \in \mathbb{R}^D$. First, we define $M(\boldsymbol{\mu}, \boldsymbol{\nu})$ as the minimum step size along an arbitrary direction in C needed for $\boldsymbol{\nu}$ to dominate $\boldsymbol{\mu}$, or formally,

$$M(\boldsymbol{\mu}, \boldsymbol{\nu}) := \inf\{s \geq 0 \mid \exists \boldsymbol{u} \in B(\mathbf{0}, 1) \cap C \colon \ \boldsymbol{\nu} + s\boldsymbol{u} \in \boldsymbol{\mu} + C\}. \tag{3.1}$$

Second, we define $m(\boldsymbol{\mu}, \boldsymbol{\nu})$ as the minimum step size along an arbitrary direction in C needed for $\boldsymbol{\mu}$ to not be dominated by $\boldsymbol{\nu}$, or formally,

$$m(\boldsymbol{\mu}, \boldsymbol{\nu}) := \inf\{s \geq 0 \mid \exists \boldsymbol{u} \in B(\mathbf{0}, 1) \cap C \colon \ \boldsymbol{\mu} + s\boldsymbol{u} \notin \boldsymbol{\nu} - \operatorname{int}(C)\}. \tag{3.2}$$

We note that it is sufficient to keep track of these two functions to identify the Pareto set because their signs reveal the dominance status of any two arms [44, Propositions 4.2, 4.3; Corollary 4.5].

Further, in their work, [44] introduced two parameters related with functions $M(\cdot,\cdot)$ (3.1), $m(\cdot,\cdot)$ (3.2), and the difficulty of comparing arms using the cone C, given as

$$\beta_1 := \sup_{\boldsymbol{\mu}\notin C} \frac{d(\boldsymbol{\mu}, C \cap (\boldsymbol{\mu}+C))}{d(\boldsymbol{\mu}, C)} \ , \tag{3.3}$$

$$\beta_2 := \sup_{\boldsymbol{\mu}\in \mathrm{int}(C)} \frac{d(\boldsymbol{\mu}, (\mathrm{int}(C))^c \cap (\boldsymbol{\mu}-C))}{d(\boldsymbol{\mu}, (\mathrm{int}(C))^c)} \ . \tag{3.4}$$

[44, Theorem 2.4] states that both of these constants, *i.e.*, β_1 and β_2, are finite, and they then define

$$\beta \coloneqq \max\{\beta_1, \beta_2\} \tag{3.5}$$

as the *ordering complexity* of cone C, which they associate with the difficulty of the vector optimization problem.

As mentioned at the beginning of this chapter, pure exploration can be studied under different objectives, such as fixed confidence and (ϵ,δ)-PAC identification. In this work, we use Definition 2 as an objective that governs both.

Definition 2. *(Success condition) Let $\epsilon \geq 0, \delta \in (0,1)$. A random set $P \subseteq \mathcal{X}$ is called an (ϵ,δ)-PAC Pareto set if the following conditions hold at least with probability $1-\delta$: (i) $P^* \subseteq P$; (ii) for every $x \in P \backslash P^*$, it holds $\Delta^*(x) \leq \epsilon$, where $\Delta^*(x) \coloneqq \max_{y\in P^*} m(\boldsymbol{f}(x), \boldsymbol{f}(y))$.*

Our success condition is a modified version of the one in [44], and is more similar in spirit to [4, Success condition 1]. Specifically, we allow $\epsilon = 0$, and we changed the first condition of [44] to a stricter one. Due to the latter modification, every (ϵ,δ)-PAC Pareto set according to Definition 2 is also a one according to their Definition 4.6.

The *sample complexity* of any algorithm for (ϵ,δ)-PAC Pareto set identification depends on the relative state of the reward vectors with respect to each other, or *gaps*. To quantify the gaps between arms and the Pareto set, we use the following definitions in [4, 44]: $\Delta^+(x) = \min_{y\in P^*\backslash\{x\}} M(\boldsymbol{f}(x), \boldsymbol{f}(y))$ for $x \in P^*$ and $\Delta^+(x) = \max_{y\in P^*} m(\boldsymbol{f}(x), \boldsymbol{f}(y))$ for $x \in \mathcal{X} \setminus P^*$. So, as the product of the

functions $M(\cdot,\cdot)$ and $m(\cdot,\cdot)$ is always zero by [44, Corollary 4.5], we can combine these together as

$$\Delta^+(x) = \max\left\{ \min_{y\in P^*\setminus\{x\}} M(\boldsymbol{f}(x),\boldsymbol{f}(y)),\ \max_{y\in P^*\setminus\{x\}} m(\boldsymbol{f}(x),\boldsymbol{f}(y)) \right\}.$$

3.2 Pareto Vector Bandits

We will now present Pareto Vector Bandits (PaVeBa), an algorithm for (ϵ,δ)-PAC Pareto identification problem on vector bandits (see Algorithm 1).

At each round $t\in\mathbb{N}$, the algorithm maintains three sets: the undecided arms $\mathcal{S}_t$ (also called "secret"), the identified Pareto arms $\mathcal{P}_t$, and the useful Pareto arms $\mathcal{U}_t\subseteq\mathcal{P}_t$. The set of "active" arms is defined as $\mathcal{A}_t \coloneqq \mathcal{S}_t\cup\mathcal{U}_t$. As long as there are undecided arms, *i.e.*, $\mathcal{S}_t\neq\emptyset$, the algorithm samples each active arm to increase its confidence in its status. It then computes the set of arms to discard, arms that are unlikely to satisfy condition (i) in Definition 2, as

$$\mathcal{D}_t = \{x\in\mathcal{S}_t \,|\, \exists y\in\mathcal{A}_t\setminus\{x\}\colon \sup_{\substack{\boldsymbol{\mu}\in\mathcal{E}_t(x),\\ \boldsymbol{\nu}\in\mathcal{E}_t(y)}} M(\boldsymbol{\mu},\boldsymbol{\nu})=0\}, \tag{3.6}$$

where $\mathcal{E}_t(x)$ is a high-probability confidence region for arm x at round t. The set $\mathcal{D}_t$ is then removed from $\mathcal{S}_t$ and $\mathcal{A}_t$ to find the intermediate sets $\overline{\mathcal{S}}_t$ and $\overline{\mathcal{A}}_t$. Next, it identifies the set of arms to deem Pareto optimal, arms that appear to satisfy condition (ii) in Definition 2, as

$$\overline{\mathcal{P}}_t = \{x\in\overline{\mathcal{S}}_t \,|\, \forall y\in\overline{\mathcal{A}}_t\setminus\{x\}\colon \sup_{\substack{\boldsymbol{\mu}\in\mathcal{E}_t(x),\\ \boldsymbol{\nu}\in\mathcal{E}_t(y)}} m(\boldsymbol{\mu},\boldsymbol{\nu})<\epsilon\}. \tag{3.7}$$

$\overline{\mathcal{P}}_t$ is then added to $\mathcal{P}_t$ to obtain $\mathcal{P}_{t+1}$, and removed from $\mathcal{S}_t$ to obtain $\mathcal{S}_{t+1}$. The algorithm also computes the set of useful Pareto arms for the next round as

$$\mathcal{U}_{t+1} = \{y\in\mathcal{P}_{t+1} \,|\, \exists x\in\mathcal{S}_{t+1}\colon \sup_{\substack{\boldsymbol{\mu}\in\mathcal{E}_t(x),\\ \boldsymbol{\nu}\in\mathcal{E}_t(y)}} m(\boldsymbol{\mu},\boldsymbol{\nu})\geq\epsilon\}, \tag{3.8}$$

which includes the arms in $\mathcal{P}_{t+1}$ that may help decide the optimality of still-undecided arms $\mathcal{S}_{t+1}$.

To make these decisions, PaVeBa needs confidence regions for the rewards of each arm. We define $\mathcal{B}_t(x) \coloneqq B(\boldsymbol{\mu}_t(x), r_t(x))$, where

$$r_t(x) \coloneqq \sqrt{\frac{8t\sigma^2}{n_t(x)^2} \log\left(\frac{\pi^2(D+1)|\mathcal{X}|t^2}{6\delta}\right)} \tag{3.9}$$

and $\boldsymbol{\mu}_t(x)$ is the sample mean of arm x at round t, *i.e.*, $\boldsymbol{\mu}_t(x) = \frac{1}{n_t(x)} \sum_{\tau \in N_t(x)} \boldsymbol{y}_\tau(x)$. Finally, we define $\mathcal{E}_t$, the confidence region of x at round t, recursively as $\mathcal{E}_t(x) \coloneqq \mathcal{E}_{t-1}(x) \cap \mathcal{B}_t(x)$ where $\mathcal{E}_1(x) \coloneqq \mathcal{B}_1(x)$.

Algorithm 1 Pareto Vector Bandits (PaVeBa)

Input: $\mathcal{X}$, C, ϵ, δ
Initialize: $\mathcal{S}_1 = \mathcal{X}$, $\mathcal{P}_1 = \emptyset$, $\mathcal{U}_1 = \emptyset$, $t = 1$;
while $\mathcal{S}_t \neq \emptyset$ **do**
    $\mathcal{A}_t = \mathcal{S}_t \cup \mathcal{U}_t$;
    for $x \in \mathcal{A}_t$ **do**
        Observe $\boldsymbol{y}_t(x)$, calculate $\mathcal{E}_t(x)$ using (3.9);
    end for
    Compute $\mathcal{D}_t$ (3.6);
    $\overline{\mathcal{S}}_t = \mathcal{S}_t \setminus \mathcal{D}_t$, $\overline{\mathcal{A}}_t = \mathcal{A}_t \setminus \mathcal{D}_t$;
    Compute $\overline{\mathcal{P}}_t$ (3.7);
    $\mathcal{P}_{t+1} = \mathcal{P}_t \cup \overline{\mathcal{P}}_t$, $\mathcal{S}_{t+1} = \overline{\mathcal{S}}_t \setminus \overline{\mathcal{P}}_t$;
    Compute $\mathcal{U}_{t+1}$ (3.8);
    $t = t + 1$;
end while
return $\hat{P} = \mathcal{P}_t$

PaVeBa can be seen as a generalization of Algorithm 1 of [4] to arbitrary ordering cones. However, it still offers some distinct features even under the standard componentwise order. First, we assume norm-subgaussian noise, which gives spherical confidence regions. In contrast, [4] assumes subgaussian noise in each objective dimension, leading to hyperrectangular confidence regions. Because of this and their restriction to the standard order, their discarding and Pareto identification operations can be held out by direct numerical comparisons. In PaVeBa, these checks require comparing all vectors inside spherical confidence regions of arms using the ordering cone, which involves solving convex optimization problems. Second, our method avoids some redundant comparisons present in their work. For instance, they search for arms to discard within $\mathcal{A}_t$ (the set A in their notation), whereas we only search within $\mathcal{S}_t$. The remaining arms in $\mathcal{A}_t$ are already

deemed likely to be Pareto optimal, so they are not considered for discarding. Third, in their algorithm, Pareto arms are decided as Pareto optimal arms (in the set P using their notation), not when they are identified as Pareto, but when they are no longer useful for determining the status of other arms. This might not seem like a significant difference, but it is using this knowledge that we can cut down redundant comparisons, as exemplified by not seeking arms to discard among useful Pareto arms.

3.3 Technical Analysis

Efficient Implementation of PaVeBa via Convex Programming. To compute the sets $\mathcal{D}_t$, $\overline{\mathcal{P}}_t$, $\mathcal{U}_{t+1}$, we need to check the validity of certain conditions that involve all possible choices of vectors in the confidence regions. For consistency with our heuristic and future use of Gaussian processes (in Section 3.4), we assume that the confidence region of an arm $x \in \mathcal{X}$ at round $t \in \mathbb{N}$ has the form of the form $\mathcal{E}_t(x) = \bigcap_{\tau=1}^{t} \mathcal{B}_\tau(x)$, where $\mathcal{B}_\tau(x) = \{\boldsymbol{\nu} \in \mathbb{R}^D \mid (\boldsymbol{\nu} - \boldsymbol{\mu}_\tau(x))^\mathsf{T} \boldsymbol{\Sigma}_\tau^{-1}(x)(\boldsymbol{\nu} - \boldsymbol{\mu}_\tau(x)) \leq \alpha_\tau(x)\}$ is an ellipsoid with parameters $\boldsymbol{\Sigma}_\tau(x)$, a symmetric positive definite matrix, and $\alpha_\tau(x) > 0$ for each $\tau \in [t]$. Note that by choosing $\boldsymbol{\Sigma}_\tau(x)$ as the identity matrix and $\alpha_\tau(x) = r_\tau(x)$, we recover $\mathcal{B}_\tau(x) = B(\boldsymbol{\mu}_\tau(x), r_\tau(x))$ as in (3.9).

In the general case, Propositions 1 and 2, proved in Appendix A.1, show how these sets can be computed using convex optimization.

Proposition 1. *Let $x, y \in \mathcal{X}$ and $t \in \mathbb{N}$. For each $n \in [N]$, consider the convex optimization problem*

$$\begin{aligned} &\textit{minimize} \quad \boldsymbol{w}_n^\mathsf{T}(\boldsymbol{\nu} - \boldsymbol{\mu}) \quad \textit{subject to:} \quad \boldsymbol{\mu}, \boldsymbol{\nu} \in \mathbb{R}^D, \\ &\quad (\boldsymbol{\mu} - \boldsymbol{\mu}_\tau(x))^\mathsf{T} \boldsymbol{\Sigma}_\tau^{-1}(x)(\boldsymbol{\mu} - \boldsymbol{\mu}_\tau(x)) \leq \alpha_\tau(x), \ \forall \tau \in [t], \\ &\quad (\boldsymbol{\nu} - \boldsymbol{\mu}_\tau(y))^\mathsf{T} \boldsymbol{\Sigma}_t^{-1}(y)(\boldsymbol{\nu} - \boldsymbol{\mu}_\tau(y)) \leq \alpha_\tau(y), \ \forall \tau \in [t]. \end{aligned}$$

Then, the optimal value of this problem is strictly negative for at least one $n \in [N]$ iff $\sup_{\boldsymbol{\mu} \in \mathcal{E}_t(x), \boldsymbol{\nu} \in \mathcal{E}_t(y)} M(\boldsymbol{\mu}, \boldsymbol{\nu}) > 0$.

In view of Proposition 1, we can practically compute $\mathcal{D}_t$. To that end, for all $x \in \mathcal{S}_t$, we can iterate over all $y \in \mathcal{A}_t \setminus \{x\}$ and solve the convex optimization problem for each $n \in [N]$. If we cannot detect at least one y and n for which the solution to this convex optimization problem is strictly negative, then we add x to $\mathcal{D}_t$.

Proposition 2. *Let $x, y \in \mathcal{X}$, $t \in \mathbb{N}$, and $\epsilon > 0$. Consider the convex feasibility problem*

$$\begin{aligned}
&\textit{minimize} \quad 0 \quad \textit{subject to:} \quad \boldsymbol{\mu}, \boldsymbol{\nu} \in \mathbb{R}^D, \\
&\boldsymbol{w}_n^{\mathsf{T}}(\boldsymbol{\nu} - \boldsymbol{\mu}) \geq \epsilon \alpha_n, \ \forall n \in [N] \\
&(\boldsymbol{\mu} - \boldsymbol{\mu}_\tau(x))^{\mathsf{T}} \boldsymbol{\Sigma}_\tau^{-1}(x)(\boldsymbol{\mu} - \boldsymbol{\mu}_\tau(x)) \leq \alpha_\tau(x), \ \forall \tau \in [t], \\
&(\boldsymbol{\nu} - \boldsymbol{\mu}_\tau(y))^{\mathsf{T}} \boldsymbol{\Sigma}_\tau^{-1}(y)(\boldsymbol{\nu} - \boldsymbol{\mu}_\tau(y)) \leq \alpha_\tau(y), \ \forall \tau \in [t].
\end{aligned}$$

Then, this problem has at least one feasible solution iff $\sup_{\boldsymbol{\mu} \in \mathcal{E}_t(x), \boldsymbol{\nu} \in \mathcal{E}_t(y)} m(\boldsymbol{\mu}, \boldsymbol{\nu}) \geq \epsilon$.

Then, in view of Proposition 2, we can practically compute $\overline{\mathcal{P}}_t$. For all $x \in \overline{\mathcal{S}}_t$, iterate over $y \in \overline{\mathcal{A}}_t \setminus \{x\}$ and solve the convex feasibility problem. If the problem is infeasible for all y, then we add x to $\overline{\mathcal{P}}_t$. Similarly, we can also calculate $\mathcal{U}_{t+1}$ using Proposition 2. In this case, for all $y \in \mathcal{P}_{t+1}$, we iterate over $x \in \mathcal{S}_{t+1}$ and solve the convex feasibility problem. If the problem is feasible at least for one x, then we add y to $\mathcal{U}_{t+1}$.

Remark 1. *When $\epsilon = 0$, Proposition 2 does not work. In this case, we propose a slightly modified algorithm with minor adjustments to the definitions of $\overline{\mathcal{P}}_t$ and $\mathcal{U}_{t+1}$. We provide another convex optimization formulation to calculate mentioned sets. For sample complexity bounds for this version, which are similar to those mentioned here, check [45].*

Sample Complexity of PaVeBa. For the sake of completeness, even though these are not the product of this thesis, we provide upper bounds for the sample complexity of the PaVeBa algorithm in this part. For further details and discussions on the sample complexities, see [45].

Theorem 1 ([45], Theorem 1)**.** *When PaVeBa is run on a finite set of arms $\mathcal{X}$, the maximum number of samples required for it to output an (ϵ, δ)-PAC Pareto set $\hat{P}$ is*

$$\frac{|\mathcal{X}|512\beta_2^2\sigma^2}{\epsilon^2}\log^+\left(\frac{256\beta_2^2\sigma^2}{\epsilon^2}\sqrt{\frac{\pi^2(D+1)|\mathcal{X}|}{6\delta}}\right)+|\mathcal{X}|\ .$$

The sample complexity bound in Theorem 1 does not depend on gaps between arms, and it nearly matches the worst-case lower bound provided in [44, Theorem 5.3].

For developing a gap-dependent bound on the sample complexity of PaVeBa, [45] further employs an improved gap definition for Pareto arms, inspired by the multi-objective framework of [4]. This is required due to the design of our algorithm in Algorithm 1. To be specific, in some cases, our algorithm will continue to sample some arms even if they almost surely belong to the Pareto set, *i.e.*, the set $\mathcal{U}_t$. To quantify the contribution of such samplings to the sample complexity of our algorithm, for each $x \in P^*$, [45] define

$$\Delta(x) \coloneqq \min\Bigg\{\min_{y\in P^*\setminus\{x\}}\{M(\boldsymbol{f}(x),\boldsymbol{f}(y)), M(\boldsymbol{f}(y),\boldsymbol{f}(x))\},\\ \min_{y\notin P^*}\left(M(\boldsymbol{f}(y),\boldsymbol{f}(x))+2\Delta^+(y)\right)\Bigg\}. \quad (3.10)$$

Finally, they let $\widetilde{\Delta}_\epsilon^+(x) \coloneqq \max\{\Delta^+(x),\epsilon\}$, $\widetilde{\Delta}_\epsilon(x) \coloneqq \max\{\Delta(x),\epsilon\}$. These ϵ-dependent gaps enable a tighter sample complexity analysis in cases where ϵ already dominates the gaps between arms, and we don't need to improve the confidence intervals for deciding the (ϵ,δ)-Pareto status. Next, an improved result based on the individual gaps of arms is presented.

Theorem 2 ([45], Theorem 2)**.** *When PaVeBa is run on a finite set of arms $\mathcal{X}$, the maximum number of samples required for it to output an (ϵ, δ)-PAC Pareto*

set $\hat{P}$ *is*

$$|\mathcal{X}| + \sum_{x \in P^*} \frac{4608\beta^2\sigma^2}{\widetilde{\Delta}_{3\epsilon}(x)^2} \log^+\left(\frac{2304\beta^2\sigma^2}{\widetilde{\Delta}_{3\epsilon}(x)^2}\sqrt{\frac{\pi^2(D+1)|\mathcal{X}|}{6\delta}}\right) + \sum_{x \in \mathcal{X} \setminus P^*} \frac{512\beta^2\sigma^2}{\widetilde{\Delta}_{\epsilon}^+(x)^2} \log^+\left(\frac{256\beta^2\sigma^2}{\widetilde{\Delta}_{\epsilon}^+(x)^2}\sqrt{\frac{\pi^2(D+1)|\mathcal{X}|}{6\delta}}\right).$$

Here, the sample complexity bound nearly matches the gap-dependent lower bound provided in [44, Theorem 5.1].

It is also worth noting that since our success condition given in Definition 2 is stricter than that of [44], the bounds it matches are for an even easier problem.

3.4 Experiments

In this section, we evaluate the performance of PaVeBa and compare it with other algorithms. We also present heuristic variants of PaVeBa that are designed to reduce sample complexity in problems with correlated arms. For high-probability correct results, algorithms often use overly conservative confidence parameters. These can hinder sampling efficiency, leading to higher computational costs. To address this, we work with scaled-down versions of these parameters and refer to the scaling term as the *contraction factor*.

Heuristic Variants of PaVeBa

To improve empirical performance in large, correlated experimental design problems, we create heuristic variants of PaVeBa that use Gaussian Processes (GPs). As is standard, each reward dimension is modeled by its own GP. We also consider correlated GP outputs, which give ellipsoidal confidence regions. The relation between the significance level of these regions and the confidence term δ is given in the Appendix A.3. We refer to the two variants as PaVeBa-IH (Independent

and Hyperrectangular) and PaVeBa-DE (Dependent and Ellipsoidal). The latter uses an Intrinsic Model of Coregionalization [47].

PaVeBa-IH: Based on the batch-independent multi-output GP implementation in GPyTorch. Each reward dimension is modeled independently, with independent and identically distributed noise. We then use hyperrectangles that encompass the confidence ball in (A.3), with α_t scaled down by a contraction factor of 64.

PaVeBa-DE: Based on the multitask GP in GPyTorch, equivalent to the Intrinsic Model of Coregionalization with full-rank inter-task covariance. Here, we directly use the hyperellipsoidal region given by the GP posterior in (A.3), also scaling down α_t by a contraction factor of 64.

For batch selection in both heuristic variants: we pick the arm with the largest posterior covariance trace, update covariances of active arms with a *fantasy update* (treating the picked arm as if we sampled it), and repeat until K arms are selected. This procedure is valid since GP variance updates do not depend on observed reward [48].

Real-World Problems

We work in finite-arm settings, with datasets built using Sobol samples (except SNW, which is already finite). For DB and VC, we use Botorch implementations [49], and for PK2 and MAR, we use code from [10].

SNW ($K = 3$, $D = 2$, $|\mathcal{X}| = 206$): Based on computational hardware design, optimizing sorting network configurations. The reward vector trades off the network's throughput (speed) and the physical area of the hardware [50].

DB (Disc Brake, $K = 4$, $D = 2$, $|\mathcal{X}| = 128$): From automotive engineering, focusing on disc brake design. Rewards balance brake mass (affecting efficiency and performance) and stopping time (safety). We use two objectives instead of

the three in [51], where they include constraints as a third objective.

PK2 ($K = 3$, $D = 2$, $|\mathcal{X}| = 500$): From organic chemistry, optimizing the Paal-Knorr synthesis of pyrroles/pyrrolidines. Rewards are the pyrrolidine yield and the space-time yield [52].

VC (Vehicle Crashworthiness, $K = 5$, $D = 3$, $|\mathcal{X}| = 2000$): From automotive safety, optimizing vehicle structure. Rewards include weight, acceleration measures, and toe-board intrusion levels (structural deformation) [51, 53]. **VC1**: A smaller version with $|\mathcal{X}| = 100$, used in compute-heavy experiments.

MAR (Marine, $K = 6$, $D = 4$, $|\mathcal{X}| = 500$): From maritime engineering, optimizing bulk carrier designs. Rewards balance cost, weight, annual cargo capacity, and regulatory constraints [51, 54].

Compared Methods

NE: Naïve elimination, based on the implementation from [44].

PIBF: We implement the [4] since no code is available, and we dub it PIBF. For fairness in comparisons with PaVeBa, we use the theoretical confidence regions of PIBF, as given in [4], rather than the empirical ones.

ϵ-PAL: We implement ϵ-PAL [5] in Python using guidance from the original MATLAB code from [5]. We scale down β_t by 9 for ϵ-PAL, as in their paper.

MESMO and JES: As additional baselines, we include two methods from MOO literature, which is a special case of vector optimization under the standard componentwise order. We adapt the information-theoretic acquisition functions MESMO [9] and JES [10] into fixed-budget algorithms. Both algorithms are originally designed for continuous domains, which puts them at a disadvantage in our finite arm setting. To make the comparison fair, we restrict their acquisition calculations to the available arms and compute the Pareto sets using their posterior means at the end of the simulation. In addition, we modify the original MESMO

implementation to handle noise by incorporating a noise parameter.

Cone ordering for other methods: For experiments with cone C other than the multi-objective cone $C = \mathbb{R}^D_+$, we adapt other baselines, *i.e.*, ϵ-PAL, MESMO and JES. We first run the algorithms as usual, then compute the Pareto set from their final posterior means using the cone ordering.

Performance Metrics

To properly account for the success conditions given in Definition 2, we contribute the following metric to the literature, which is a loose F1 score.

Definition 3 (ϵ**-F1 Score**)**.** *Given an input space $\mathcal{X}$ and parameter ϵ, define the positive arms set Π_ϵ such that*

$$\Pi_\epsilon = \{x \in \mathcal{X} : \Delta^*(x) \leq \epsilon\},$$

where $\Delta^*(x)$ is defined in Definition 1. Let an algorithm for (ϵ, δ)-PAC Pareto set identification return $\hat{P}$ as the predicted Pareto set. Further, define the false negative set of arms $\Pi_{\epsilon\setminus\hat{P}}$ as

$$\Pi_{\epsilon\setminus\hat{P}} = \left\{y \in P^* : (\boldsymbol{f}(y) - C) \not\subseteq \cup_{x\in\hat{P}} \left(\boldsymbol{f}(x) + (B(\mathbf{0}, \epsilon) \cap C) - C\right)\right\}.$$

Specifically, $\Pi_{\epsilon\setminus\hat{P}}$ is the set of Pareto optimal arms that is not covered by $\hat{P}$ where covering is defined in part (i) of [44, Definition 4.6]. Then, the ϵ-F1 Score for that algorithm is defined as

$$\epsilon\text{-}F1 = \frac{2|\Pi_\epsilon \cap \hat{P}|}{2|\Pi_\epsilon \cap \hat{P}| + |\Pi_{\epsilon\setminus\hat{P}}| + |\hat{P} \setminus \Pi_\epsilon|}. \tag{3.11}$$

In the ϵ-F1 Score, the Pareto identification problem is treated as a classification problem where the positive class is the ϵ-accurate Pareto optimal arms. Some things to note about this metric are:

- $P^* \subseteq \Pi_\epsilon$.

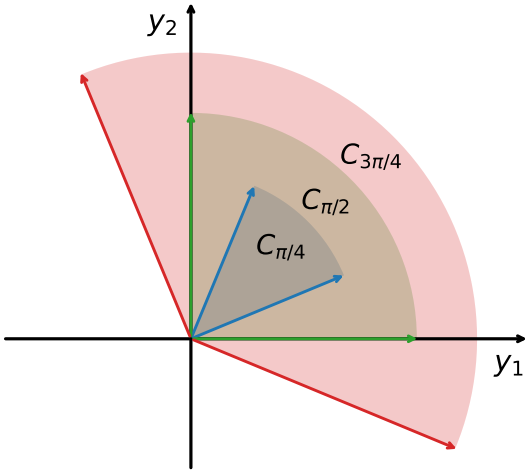


Figure 3.1: Ordering cones for $D = 2$ where $\theta \in \{\pi/4, \pi/2, 3\pi/4\}$ in Definition 4 and $\boldsymbol{y} = [y_1\ y_2] \in \mathbb{R}^D$.

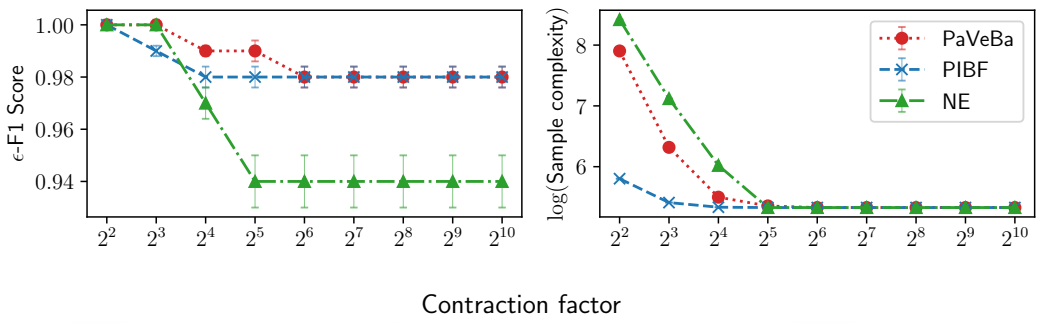
- If an algorithm satisfy condition (i) in [44, Definition 4.6], meaning all $x \in P^*$ are ϵ-covered, then we have $\Pi_{\epsilon\setminus\hat{P}} = \emptyset$. Thus, if an algorithm satisfies success condition (i) in Definition 2 (*i.e.*, $P^* \subseteq \hat{P}$), we have $\Pi_{\epsilon\setminus\hat{P}} = \emptyset$ as well.
- If an algorithm satisfies success condition (ii) in Definition 2, then we have $\hat{P} \setminus \Pi_\epsilon = \emptyset$.
- ϵ-F1 $= 1$ iff an algorithm satisfies [44, Definition 4.6].

We use this as our primary accuracy metric, as it parallels classification measures. For fairness, all algorithms are scored using the same ϵ used in their (ϵ, δ)-PAC Pareto identification setup.

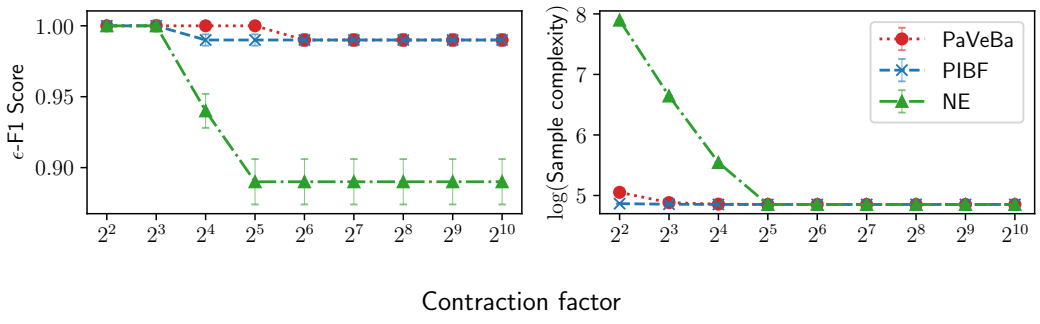
Sample Complexity: This is measured simply as the number of evaluations performed by the algorithm.

Useful Cones

We define a family of cones where $D = 2$ for convenience.



(a)



(b)

Figure 3.2: Comparison with NE and PIBF on SNW (a) and DB (b) with $\sigma_n = \epsilon = 0.01$. Figures on the left show ϵ-F1 scores with varying contraction factors, and figures on the right show the sample complexities.

Definition 4. *Given an angle θ, the cone $C_\theta \subseteq \mathbb{R}^2$ is defined as the convex cone whose two boundary rays make $\frac{\pm\theta}{2}$ radians with the identity line. In other words,*

$$C_\theta := \{\boldsymbol{x} \in \mathbb{R}^2 \mid \pi/4 - \theta/2 \leq \theta_{\boldsymbol{x}} \leq \pi/4 + \theta/2\},$$

where $\theta_{\boldsymbol{x}}$ denotes the angle in the polar coordinates of a point $\boldsymbol{x} \in \mathbb{R}^2$.

Remark 2. *Cone $C_{\pi/2}$ in Definition 4 corresponds to standard componentwise cone with $D = 2$.*

Definition 5. *For experiments where $D = 3$, we define two other cones with the following $\boldsymbol{W}$ matrices and refer to them as acute and obtuse cones, respectively.*

$$\textit{Acute: } \boldsymbol{W} = \begin{bmatrix} 1 & -2 & 4 \\ 4 & 1 & -2 \\ -2 & 4 & 1 \end{bmatrix} / \sqrt{21} \qquad \textit{Obtuse: } \boldsymbol{W} = \begin{bmatrix} 1 & 0.4 & 1.6 \\ 1.6 & 1 & 0.4 \\ 0.4 & 1.6 & 1 \end{bmatrix} / \sqrt{3.72}$$

		ϵ-F1 Score w.r.t. θ				
	ϵ	$\pi/4$	$\pi/3$	$\pi/2$	$2\pi/3$	$3\pi/4$
NE	10^{-1}	**0.99 ± .01**	0.99 ± .01	0.97 ± .03	0.96 ± .04	0.97 ± .04
	10^{-2}	0.93 ± .02	0.91 ± .03	0.87 ± .04	0.82 ± .07	0.78 ± .10
PaVeBa	10^{-1}	**0.99 ± .01**	**1.00 ± .00**	**0.98 ± .01**	**0.99 ± .02**	**0.98 ± .02**
	10^{-2}	**0.94 ± .02**	**0.93 ± .03**	**0.91 ± .04**	**0.88 ± .05**	**0.88 ± .08**

(a)

NE	10^{-1}	**0.98 ± .02**	0.96 ± .04	0.94 ± .06	**0.95 ± .06**	**0.97 ± .06**
	10^{-2}	0.93 ± .04	0.89 ± .05	0.84 ± .09	0.93 ± .07	0.78 ± .17
PaVeBa	10^{-1}	**0.98 ± .02**	**0.97 ± .03**	**0.95 ± .05**	**0.95 ± .07**	**0.97 ± .06**
	10^{-2}	**0.94 ± .03**	**0.94 ± .04**	**0.92 ± .07**	**0.94 ± .08**	**0.91 ± .14**

(b)

Table 3.1: ϵ-F1 scores of NE and PaVeBa for different values of ϵ and θ on SNW (a) and DB (b) datasets. The best results are in bold.

		Sample Complexity w.r.t. θ				
	ϵ	$\pi/4$	$\pi/3$	$\pi/2$	$2\pi/3$	$3\pi/4$
SNW	10^{-1}	654.5 ± 50.9	499.1 ± 42.4	378.7 ± 28.2	306.9 ± 21.7	272.4 ± 17.2
	10^{-2}	10100.9 ± 3315.0	6053.9 ± 1480.0	2594.0 ± 982.5	1162.9 ± 684.0	790.0 ± 442.7
DB	10^{-1}	304.9 ± 38.7	218.8 ± 21.1	167.6 ± 10.4	145.0 ± 9.7	141.8 ± 7.3
	10^{-2}	2045.3 ± 1915.3	780.0 ± 561.3	308.8 ± 163.5	473.8 ± 603.6	196.7 ± 67.2

Table 3.2: Sample complexities of PaVeBa with different values of ϵ and θ on SNW and DB datasets.

Experimental Setup and Results

Before running the experiments, we min-max scale the inputs to $[0, 1]$ and standardize the outputs. Unless otherwise stated, we add Gaussian noise as $\mathcal{N}(\mathbf{0}, \sigma_n^2 \boldsymbol{I}_D)$ with $\sigma_n = 0.1$, set $\epsilon = 0.1$ and $\delta = 0.05$. Note that when $D = 2$, this noise model is σ_n-subgaussian and σ_n-norm-subgaussian. To lower the number of constraints in PaVeBa's convex programs from $O(t)$ to $O(1)$, we replace $\mathcal{E}_t(\cdot)$ with $\mathcal{B}_t(\cdot)$ inside PaVeBa. For GP-based models, we use RBF kernels with automatic relevance determination. Kernel parameters are assumed to be known and are chosen via maximum likelihood estimation on the dataset before optimization. All reported results are averaged over 50 runs.

Dataset	$\lvert\mathcal{X}\rvert$	Algorithm	Sample Complexity	ϵ-F1 Score
PK2	500	ϵ-PAL	178.4 ± 55.9	1.00 ± .01
		JES	57.62 ± 8.12	0.86 ± .06
		MESMO	57.62 ± 8.12	0.86 ± .05
		PaVeBa-IH	57.62 ± 8.12	0.95 ± .08
VC	2000	ϵ-PAL	584.0 ± 61.0	1.00 ± .00
		JES	123.94 ± 14.80	0.87 ± .04
		MESMO	123.94 ± 14.80	0.97 ± .02
		PaVeBa-IH	123.94 ± 14.80	0.97 ± .03
MAR	500	ϵ-PAL	639.04 ± 152.63	0.98 ± .01
		JES	224.38 ± 11.76	0.97 ± .01
		MESMO	224.38 ± 11.76	0.95 ± .03
		PaVeBa-IH	224.38 ± 11.76	0.97 ± .01

Table 3.3: Comparison of PaVeBa with ϵ-PAL, MESMO, and JES under the multi-objective cone.

Experiment 1. We assess the performance of PaVeBa in multi-objective optimization, a special case of vector optimization where the ordering cone is the positive orthant. PaVeBa is compared with Algorithm 1 in [4], denoted here as PIBF, and Naïve Elimination (NE) proposed in [44]. We scale down the confidence regions given by PaVeBa and PIBF, i.e., $r_t(x)$, and the per-arm sampling budget calculated by NE, i.e., L. We use contraction factors chosen as powers of 2 in $[2^2, 2^{10}]$. We set $\sigma_n = \epsilon = 0.01$ for this experiment due to high sample complexities. The results (Figure 3.2) show that PaVeBa remains robust even when the confidence assumptions are not satisfied. With higher contraction factors, its sample complexity is similar to PIBF, while both PaVeBa and PIBF outperform NE.

Experiment 2. We evaluate the sample complexity of PaVeBa under three different polyhedral ordering cones C and examine how sample complexity relates to ordering complexity. Specifically, we take $\theta \in \{\pi/4, \pi/2, 3\pi/4\}$ in Definition 4. For comparison, we also include NE. NE uses a per-arm sampling budget L. So, for fairness, we give it the next multiple of $|\mathcal{X}|$ samples larger than what PaVeBa used, *i.e.*, $L = \lceil T/|\mathcal{X}| \rceil$ where T is the sampling complexity of PaVeBa

Cone	Algorithm	Sample Complexity	ϵ-F1 Score
Acute	ϵ-PAL	88.08 $\pm$ 28.06	0.98 $\pm$ .02
	JES	91.94 $\pm$ 15.33	0.81 $\pm$ .04
	MESMO	91.94 $\pm$ 15.33	0.97 $\pm$ .02
	PaVeBa-DE	91.94 $\pm$ 15.33	0.99 $\pm$ .02
Obtuse	ϵ-PAL	88.08 $\pm$ 28.06	1.00 $\pm$ .00
	JES	27.58 $\pm$ 5.68	0.86 $\pm$ .07
	MESMO	27.58 $\pm$ 5.68	0.85 $\pm$ .11
	PaVeBa-DE	27.58 $\pm$ 5.68	1.00 $\pm$ .03

Table 3.4: Comparison of PaVeBa with ϵ-PAL, MESMO, and JES on the VC1 dataset under acute and obtuse cones.

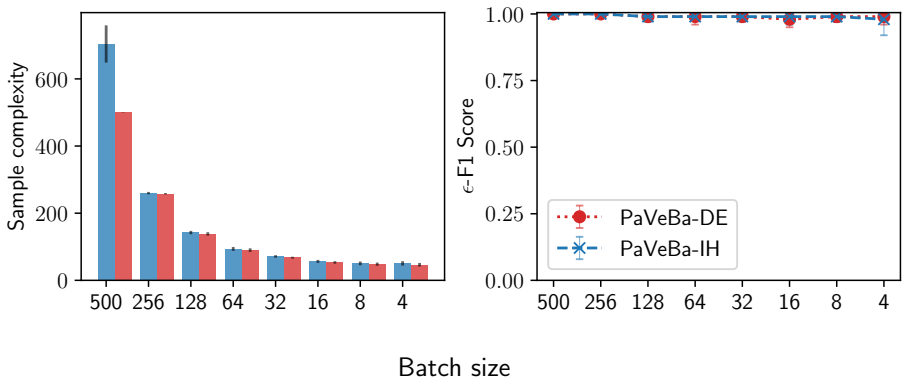


Figure 3.3: The sample complexities (Left) and ϵ-F1 scores (Right) of PaVeBa-IH and PaVeBa-DE on PK2.

for that specific experiment. PaVeBa is run with a contraction factor of. The results in Tables 3.1 and 3.2 indicate that PaVeBa achieves better performance under similar sample complexity.

Experiment 3. We compare the heuristic variants of PaVeBa against ϵ-PAL [5], MESMO [9] and JES [10]. Since these baselines sample only one arm per round, we add a batch size parameter κ to PaVeBa-IH and PaVeBa-DE. PaVeBa and ϵ-PAL work in the fixed confidence setting, while others work with a fixed sampling budget. To ensure fairness, in each run, we allocate a budget to MESMO and

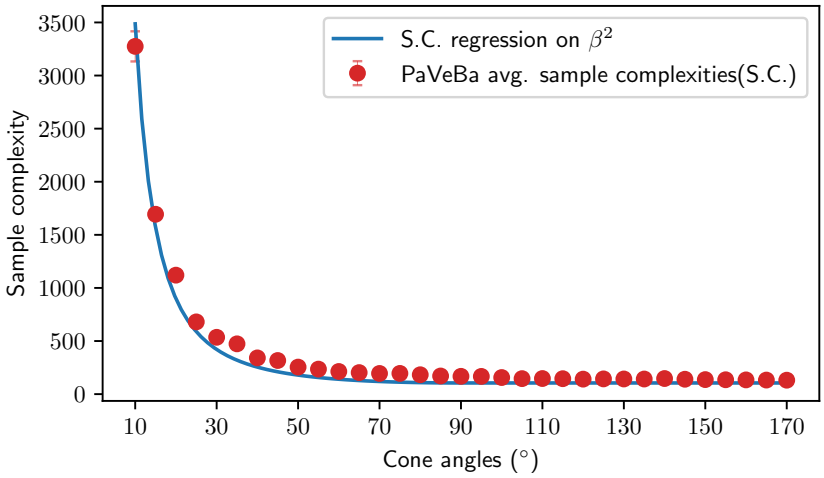


Figure 3.4: Least squares fit of the average sample complexity using β^2 for different cone angles. Angles are sampled with step size 5° from the range $[10°, 170°]$.

JES equal to the number of samples taken by PaVeBa. We scale down $\alpha_\tau(x)$ by 64 for PaVeBa and β_t by 9 for ϵ-PAL, following [5]. In addition, to reduce PaVeBa's sample complexity without compromising ϵ correctness, we relax (3.6) by allowing ϵ-looseness in discarding.

We first set $\kappa = 1$. In the first simulation (Table 3.3), we compare the algorithms using the multi-objective cone. In the second simulation (Table 3.4), we work with $D = 3$ and evaluate the algorithms under the so-called acute and obtuse cones we defined previously in Definition 5. For algorithms other than PaVeBa, cone-ordered Pareto sets are computed directly from the final posterior means. The results show PaVeBa matches state-of-the-art methods across several datasets with different reward dimensions and cones. While ϵ-PAL achieves good ϵ-F1 scores, it generally requires more samples. In the third simulation, we compare PaVeBa-IH to PaVeBa-DE on different batch sizes. From Figure 3.3 (Left), we see that (1) modeling correlations between objectives improves sample efficiency, and (2) larger batch sizes increase sample complexity. Figure 3.3 (Right) shows that the ϵ-F1 scores of two variants of PaVeBa are nearly the same, suggesting that the main benefit of ellipsoidal confidence regions lies in lowering sample

complexities rather than improving accuracy.

Experiment 4. To relate the empirical sample complexity with the ordering complexity β (see Equation 3.5), we do a regression analysis on the DB dataset. We use the same cone definition as in Experiment 2. We regress a multiplier for β^2 using the least squares approach to fit the empirical sample complexities observed. In Figure 3.4, it can be seen that β^2 matches well with the empirical observations. This finding consolidates the theoretical findings related to the sample complexity of PaVeBa [45, Theorems 1 and 2].

Chapter 4

VOPy: A Framework for Black-box Vector Optimization

4.1 Introduction

Black-box optimization is a subfield of optimization where the optimal value of a function is sought without using derivatives, primarily relying on potentially noisy pointwise evaluations of the function. In many real-life problems, gradients are not available, making black-box optimization methods invaluable. These applications include hyperparameter tuning of machine-learning algorithms [55, 56], neural architecture search [57], portfolio allocation [58], and synthetic gene design [59]. Black-box optimization is also widely used in experimental design [60], such as thermal conductivity [61] and crystal-structure prediction [62]. Libraries like `COMBO` [63], `RBFOpt` [64], `Google Vizier` [65], and `OpenBox` [66] have been implemented to perform black-box optimization.

Despite the success of single-objective optimization, many applications require simultaneous optimization of different objectives. Multi-objective optimization (MOO) typically addresses this issue using the componentwise order for objective vectors [3]. But one may want to consider a more general order to reflect more and

Figure 4.1: Left shows a bi-objective Pareto front with respect to MOO, which all the mentioned state-of-the-art libraries aim to find. Middle and right show Pareto fronts with respect to two other preferences, which only `VOPy` can be configured to find.

less conservative ways in which one arm dominates another through preferences (see Figure 4.1). In vector optimization (VO), one defines a vector partial order $\preceq_C$ on $\mathbb{R}^D$ induced by a closed convex cone C with $C \cap -C = \{\mathbf{0}\}$ via $\boldsymbol{\mu} \preceq_C \boldsymbol{\nu}$ if and only if $\boldsymbol{\nu} - \boldsymbol{\mu} \in C$. Here, $D \geq 2$ is the number of objectives and $\boldsymbol{\mu}, \boldsymbol{\nu} \in \mathbb{R}^D$ are arbitrary objective vectors. Then, the goal is to find the *Pareto set* of a vector-valued function $\boldsymbol{f}$ with respect to $\preceq_C$ over a given set of arms through sequential pointwise observations. See Section 2.1 for more details. In different applications, there could be observation noise, limited observation budget, partial observations of $\boldsymbol{f}$, the ability to accommodate batch observations, and so on. Although there is recent work on black-box vector optimization algorithms [1, 44, 45], a unifying framework for working with these cases is missing for algorithm designers and practitioners. We introduce `VOPy` to ease the development and applications of the field.

Related work. While there are no tools specifically designed for black-box VO, several libraries have been developed for MOO. An important example is `PyMOO` [67], which offers a comprehensive suite of ready-to-use MOO algorithms. But it does not include Bayesian methods and falls short in facilitating decision-making processes crucial for end-to-end black-box algorithms. Some tools take advantage of Bayesian optimization (BO), which is widely used in MOO applications. For instance, `PyePAL` [68] implements the ϵ-`PAL` algorithm [5], providing utilities for its application across various problems. However, it lacks the flexibility to adapt to new or varied algorithms. `OpenBox` [66], a comprehensive platform for black-box optimization, and `Optuna` [69], an easy-to-use hyperparameter optimization

framework, support MOO. However, they lack the accommodation of custom algorithm development. Lastly, `BoTorch` [49] is a well-rounded library highly regarded for its extensiveness and modularity in Bayesian optimization. It supports decoupled algorithms, a feature shared with `VOPy`. While these MOO libraries offer various capabilities, crucially, they **all** lack support for the general partial orders essential for vector optimization—a gap `VOPy` is designed to address. To emphasize this point, we include a comparison with `Optuna` under examples in our `documentation website`.

Contributions. **(1)** We design and implement `VOPy`, the first library for developing black-box vector optimization algorithms. **(2)** We provide solutions to the convex optimization problems that arise while performing confidence region comparisons with respect to $\preceq_C$. **(3)** We implement some of the existing VO and MOO algorithms from the literature using the modular design of `VOPy`. **(4)** We provide support for decoupled algorithms, which enable partial evaluations when objective costs may differ, along with novel example algorithms.

4.2 VOPy

`VOPy` is an open-source library where it is possible to use the existing vector optimization methods for applications, develop new algorithms using the efficient framework of `VOPy` for uncertainty and partial-order modeling, and develop methods almost from scratch and benchmark them against known methods. `VOPy` adopts a modular design philosophy, inspired by the Bayesian optimization library `BoTorch`. An overview of `VOPy` with its dependencies, core modules, and built-in algorithms can be seen in Figure 4.2. Source code for `VOPy` resides in `https://github.com/Bilkent-CYBORG/VOPy/`.

Design and Implementation. The architecture of `VOPy` is divided into four main interfaces: `Order`, `Model`, `Algorithm`, and `Problem`. The `Order` interface is central to defining the partial order used across the library. This can be the componentwise order in MOO or any other partial order defined by a polyhedral

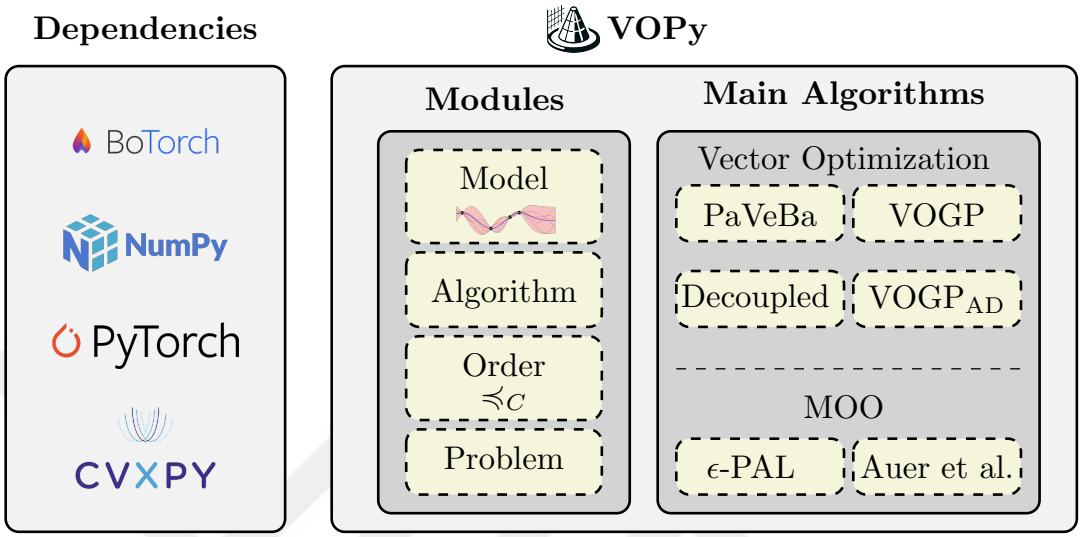


Figure 4.2: Overview of the dependencies, core modules, and built-in algorithms of `VOPy`.

ordering cone. This interface is critical as it allows all components of the library to perform solution comparisons. The `Model` interface facilitates the creation of various modeling components that can use both frequentist or Bayesian methods. `VOPy` comes with some built-in models that should suffice for most quick implementations: the empirical mean and variance model, multi-output Gaussian processes (GP) with independent or coregionalized outputs, and GPs that support decoupled modeling. The `Algorithm` interface is designed to foster algorithms. Implementations of the state-of-the-art VO algorithms, as well as some important MOO algorithms, are presented together with `VOPy`. Lastly, with the `Problem` interface, the user can connect their optimization problem with the other components of `VOPy`. The necessary tools to generate problems from offline datasets or link real-world problems are available. Together, these interfaces promote a flexible yet structured approach for developing and implementing end-to-end black-box VO algorithms, ensuring that `VOPy` can be adapted to a wide range of optimization problems and settings. In noisy black-box optimization, possible solutions have confidence regions containing the actual function value with high probability. In MOO, hyperrectangles are used, which fit well with componentwise order for easy comparison between regions. However, this approach does not

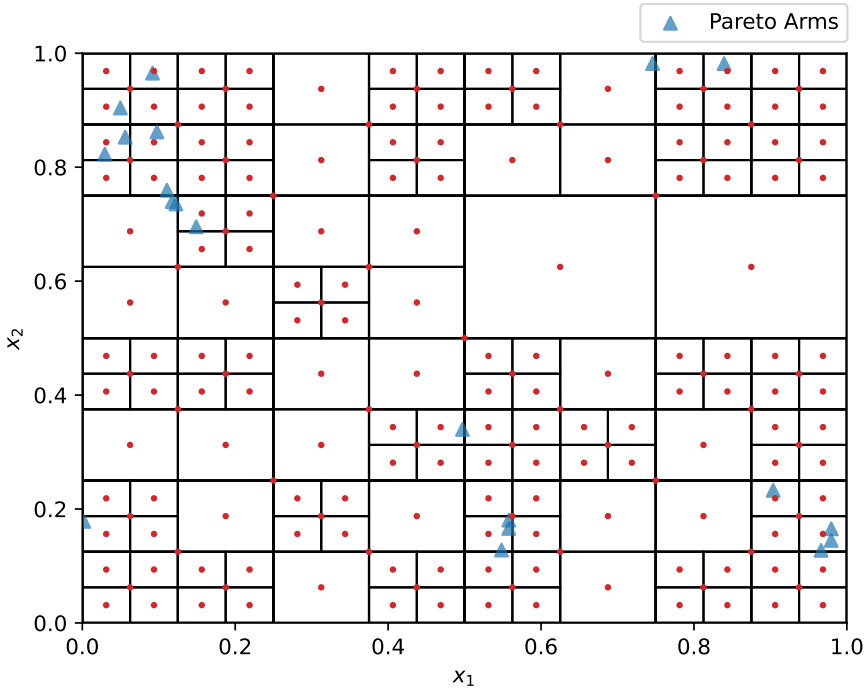


Figure 4.3: An example of an adaptively discretized 2-dimensional domain while using the VOGP [1] algorithm from VOPy. Pareto points, and thus the focus of the algorithm may also be seen in the figure.

extend to arbitrary partial orders, nor can it exactly represent models involving dependent objectives. To address this, we provide a `ConfidenceRegion` abstract class with `RectangularConfidenceRegion` and `EllipsoidalConfidenceRegion` derivatives, both supporting solutions to these comparisons using the `CVXPY` library.

`VOPy` is designed with performance in mind and provides significant speed-ups when possible (see Performance notebook). `VOPy` follows the coding conventions of PEP8 [70] and enforces it through Flake8, Black and μsort. `VOPy` has also been checked for vulnerabilities with the tool Bandit and is secure. These choices align with most of the frequently used libraries in the Python community, such as the PyTorch ecosystem.

`VOPy` is extensively tested with a high code coverage ($\geq 95\%$). Documentations for `VOPy` are accessible at `https://vopy.readthedocs.io/en/stable/`.

4.3 How to use VOPy?

VOPy features various VO algorithms, including Naive Elimination [44], PaVeBa and PaVeBa-GP [45], and VOGP, along with its extension to continuous domains using adaptive discretization [1] (see Figure 4.3). From the MOO literature, VOPy implements Algorithm 1 of [4], which has no official code released, and ϵ-PAL [5]. VOPy also includes a version of PaVeBa that works with unknown hyperparameters, a decoupled version of PaVeBa, and a novel entropy-based decoupled algorithm.

VOPy natively supports componentwise orders, 2D cones, polyhedral approximations of 3D ice-cream cones, and polyhedral cones defined by a coefficient matrix. The library includes various GPyTorch models that handle coupled, decoupled, dependent, and independent objectives. Furthermore, the library includes input/output transformation modules that can be optionally passed to the models to facilitate data normalization within the models. For real-world datasets with vector-valued outputs, VOPy provides the SNW dataset [50] and datasets for the Disk Brake [51], Vehicle Safety [53], and ML Fairness [71] problems, generated using Sobol sequences.

Using existing methods for new problems. Researchers and practitioners can define their specific problem instances and integrate their experimental data to apply VOPy's black-box vector-optimization tools. These ready-to-use components can be directly applied to optimization tasks with minimal setup. Following the documentation's examples, users only need to connect their problem through a dataset or problem structure, set up the appropriate algorithm with an existing model and order, and let the algorithm run.

Benchmarking. Algorithm developers can create new algorithms by inheriting from VOPy's abstract algorithm class, leveraging existing models, orders, and problems to compare their results with existing algorithms. VOPy's benchmarking capabilities include metrics such as ϵ-F1 score [45] and hypervolume discrepancy [10].

Custom uses. Any module in `VOPy` can be replaced with custom methods, for example, users can incorporate a novel uncertainty model, using it together with existing algorithms and orders. Alternatively, if an application requires a non-polyhedral cone, the practitioner can define a custom `Order` derivative and apply existing algorithms and models to the problem.

Chapter 5

Robust Satisficing in Adversarial Setting with Thompson Sampling

In the preliminary Section 2.3, we introduced robust optimization and robust satisficing in a general sense. In this chapter, we will focus on a specific setting, which is the adversarial setting, and build alongside existing work. Specifically, we will implement the Thompson sampling version of the algorithm family in [2].

One form of uncertainty that RO and RS aim at addressing is perturbations to the actions (arms). In many learning systems, even small, targeted changes to the input can lead to significant deviations in the output, rendering the system unreliable. This phenomenon is well-studied in deep neural networks under the name of adversarial examples [72, 73], but it also appears in other areas, such as control and portfolio optimization [74, 75]. In finance, however, worst-case models designed to guard against such sensitivity often become overly conservative, recommending trivial solutions. Handling such cases requires robustness strategies that extend beyond worst-case approaches, especially when the perturbation model or budget is unknown or only partially known.

5.1 Problem Formulation

We will work with the same problem formulation studied in [2]. Consider f to be an unknown function defined on the domain $\mathcal{X} \subset \mathbb{R}^m$, endowed with a pseudometric $d(\cdot, \cdot) : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$. At each round $t \in [T]$, where T represents the time horizon, the learner picks an arm $\tilde{x}_t \in \mathcal{X}$, in response to which the adversary selects a perturbation δ_t. The learner then observes the perturbed arm $x_t = \tilde{x}_t + \delta_t$ along with the noisy sample of the arm $y_t = f(x_t) + \eta_t$, where η_t are independent random variables, taken to be Gaussian for experiments in this chapter. Suppose the adversary is allowed to play any perturbation δ_t no matter what action $\tilde{x}_t$ the learner chooses. In that case, the adversary can always choose a perturbation δ_t such that $\tilde{x}_t + \delta_t = \arg\min_{x \in \mathcal{X}} f(x)$, and the optimization problem would be trivially hopeless. Therefore, at each round t, we assume an *unknown perturbation budget* ϵ_t, that bounds the set of perturbations the adversary can play. We define for each point $x \in \mathcal{X}$, the *uncertainty set*

$$\Delta_\epsilon(x) := \{x' - x : x' \in \mathcal{X} \text{ and } d(x, x') \leq \epsilon\} \ . \tag{5.1}$$

The uncertainty set (5.1) defines the set of all permissible perturbations of the adversary, when the perturbation budget is ϵ. Hence in response to action $\tilde{x}_t$, the adversary plays the perturbation $\delta_t \in \Delta_{\epsilon_t}(\tilde{x}_t)$.

5.2 Robust Satisficing Approach

We directly use the RS-G formulation from [2] in the adversarial setting. The objective of this formulation is to find $x^{\text{RS-G}} \in \mathcal{X}$ that solves the optimization problem $\min_{x \in \mathcal{X}} \kappa_{\tau,p}(x)$ where $\kappa_{\tau,p}(x)$ is the p-fragility of action $x \in \mathcal{X}$ defined as

$$\kappa_{\tau,p}(x) \coloneqq \min k \ \text{ s.t. } \ f(x+\delta) \geq \tau - [kd(x, x+\delta)]^p \,, \quad \forall \delta \in \Delta_\infty(x) \ . \tag{5.2}$$

This is computationally equivalent to solving

$$\kappa_{\tau,p}(x) := \begin{cases} \max_{\delta \in \Delta_\infty(x) \setminus 0} \frac{[\tau - f(x+\delta)]^{1/p}}{d(x, x+\delta)} & \text{if } f(x) \geq \tau \\ +\infty & \text{if } f(x) < \tau. \end{cases} \tag{5.3}$$

Algorithm 2 AdveRS-G-TS

Input: Kernel function k, $\mathcal{X}$, time horizon T
Initialize: $\mathcal{D}_0 = \emptyset$ (empty dataset), and $\mu_0(x) = 0$, $\sigma_0(x) = 1 \quad \forall x \in \mathcal{X}$
for $t = 1$ to T **do**
    Sample $\tilde{f}_t(x) \sim \mathcal{GP}\big(\mu_{t-1}(x), \sigma^2_{t-1}(x)\big)$, $\forall x \in \mathcal{X}$
    Compute $\overline{\kappa}_{\tau,p,t}(x)$ using (5.5), $\forall x \in \mathcal{X}$
    Select $\tilde{x}_t = \arg\min_{x\in\mathcal{X}} \overline{\kappa}_{\tau,t}(x)$
    Adversary selects perturbation $\delta_t \in \Delta_{\epsilon_t}(\tilde{x}_t)$
    Sample $x_t = \tilde{x}_t + \delta_t$, observe $y_t = f(x_t) + \eta_t$
    $\mathcal{D}_t = \mathcal{D}_{t-1} \cup \{(\tilde{x}_t, \delta_t, y_t)\}$
    Update GP posterior using (2.1)
end for

When (5.2) is feasible, the RS-G action is defined as $x^{\text{RS-G}} = \arg\min_{x\in\mathcal{X}} \kappa_{\tau,p}(x)$.

Regret Measure. We use *lenient regret* introduced by [36], as it is a natural metric for satisficing objectives, defined as

$$R^l_T := \sum_{t=1}^{T} \left(\tau - f(x_t)\right)^+ . \tag{5.4}$$

It computes the cumulative loss with respect to the satisficing threshold τ. Note that under unconstrained adversaries, sublinear lenient regret is not attainable, except for trivial τ values.

Algorithm for RS-G using Thompson sampling. In order to perform Bayesian optimization with the objective as RS-G, building on the algorithm structure in [2], we propose *Adversarially Robust Satisficing-General-Thompson Sampling* (AdveRS-G-TS) algorithm (see Algorithm 2).

At the beginning of each round t, AdveRS-G-TS samples an $\tilde{f}_t$ from the posterior distribution of GP. Then, replacing the f in (5.2) instead with $\tilde{f}_t$, it computes the Thompson p-fragility defined as:

$$\overline{\kappa}_{\tau,p,t}(x) := \max_{\delta\in\Delta_\infty(x)\setminus 0} \frac{\left[\tau - \tilde{f}_t(x+\delta)\right]^{1/p}}{d(x, x+\delta)} , \tag{5.5}$$

if $\tilde{f}_t(x) \geq \tau$, and otherwise $\overline{\kappa}_{\tau,p,t}(x) := \infty$. For each action, the Thompson p-fragility $\overline{\kappa}_{\tau,p,t}(x)$ provides a random sample of the true fragility $\kappa_{\tau,p}(x)$.

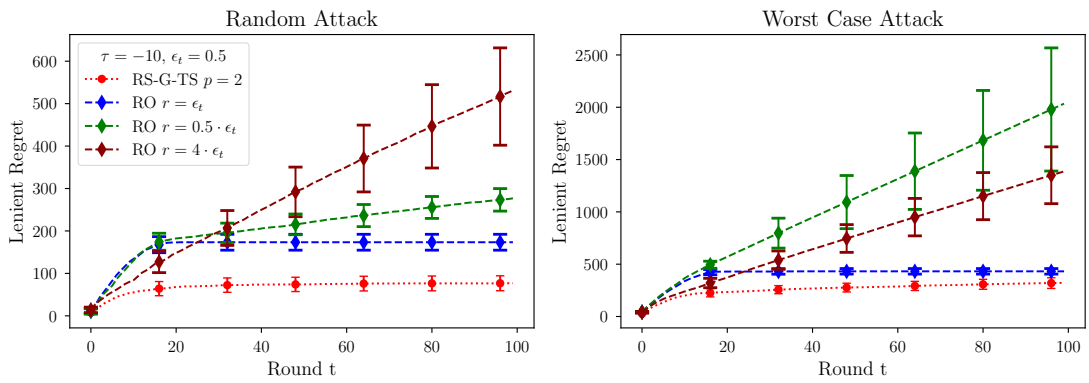


Figure 5.1: Lenient regret on the synthetic problem given in [2], which is originally from [76].

AdveRS-G-TS then selects the action with the smallest random fragility, $\tilde{x}_t = \arg\min_{x \in \mathcal{X}} \overline{\kappa}_{\tau,p,t}(x)$, to guide exploration and exploitation.

5.3 Experiments

Following the experimental design of [2], we retain their setup but replace their algorithms with AdveRS-G-TS, comparing against the robust optimization (RO) baseline of [76]. The RO method is evaluated with ambiguity radii r equal to, smaller than, and larger than the true perturbation budget ϵ_t, using the Euclidean metric to measure perturbations.

We test the algorithms under two adversarial strategies:

$$\textbf{Random: } \delta_t \sim \text{Uni}(\Delta_{\epsilon_t}(\tilde{x}_t)),$$
$$\textbf{Worst Case: } \delta_t = \arg\min \delta \in \Delta_{\epsilon_t}(\tilde{x}_t) f(\tilde{x}_t).$$

We repeat each experiment 50 times, and report results with error bars corresponding to half the standard deviation.

Synthetic experiment. We begin with the illustrative function in [2, Figure 2], originally adapted from [76]. The threshold is set $\tau = -10$, and $\epsilon_t = 0.5$. As a baseline, we run STABLEOPT [76] with radius parameters $r = \epsilon_t$, $r = 4\epsilon_t$, and

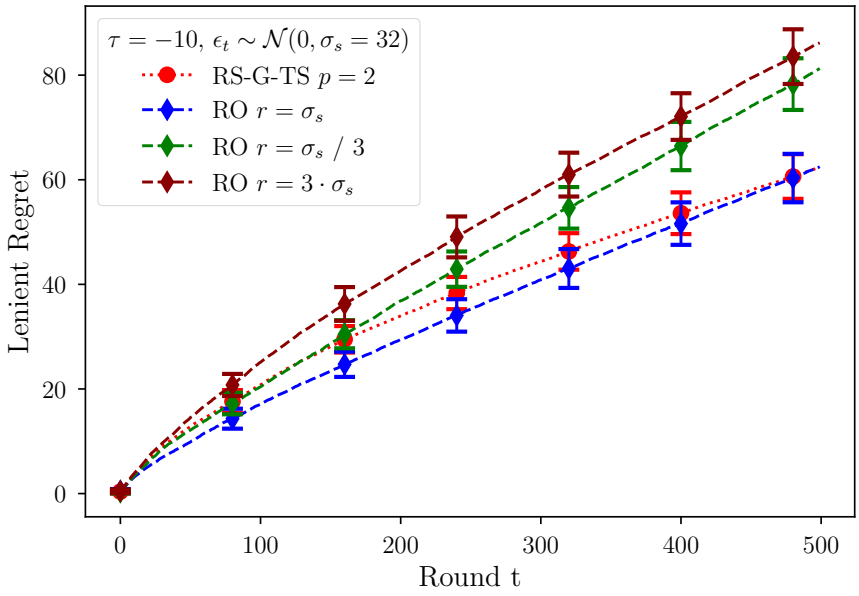


Figure 5.2: Lenient regret of the insulin dosage experiment.

$r = 0.5\epsilon_t$. Observations include Gaussian noise $\eta_t \sim \mathcal{N}(0, 1)$, and the GP model uses a polynomial kernel trained on 500 function samples before experimentation.

Figure 5.1 shows that STABLEOPT suffers linear regret when r is misspecified, while AdveRS-G-TS with $p = 2$ consistently satisfies τ with sublinear lenient regret. STABLEOPT with the correct knowledge of the adversary also achieves sublinear regret.

Insulin dosage. We next apply adversarial contextual GP optimization, where at each step the learner selects $x_t \in \mathcal{X}$ after observing a reference context $c_t^{\text{ref}} \in \mathcal{C}$. The adversary perturbs this context to $c_t = c_t^{\text{ref}} + \delta_t$, and the learner observes $f(x_t, c_t) + \eta_t$. This setup is instantiated in automated insulin dosing for Type 1 Diabetes Mellitus (T1DM), using the UVA/PADOVA simulator [77]. Bolus insulin is typically administered before meals [78], and carbohydrate information improves postprandial blood glucose (PBG) prediction [79]. Since pre-meal announcements often deviate from actual intake, robustness to context shifts is essential. The goal is to keep PBG within a τ-neighborhood of a clinical target K. The action space is insulin dosage $[0, 15]$ units, and the context is carbohydrate

intake perturbed by $\delta_t \sim \mathcal{N}(0, \sigma_s)$ with $\sigma_s = 32$ (corresponding to roughly two slices of bread). As shown in Figure 5.2, none of the algorithms achieve sublinear regret, but AdveRS-G-TS outperforms STABLEOPT when r is not tuned to σ_s. STABLEOPT performs similarly to AdveRS-G-TS if it is run with $r = \sigma_s$.

Conclusion. In these experiments, we painted the general picture that AdveRS-G-TS performs very well without knowing the adversary's budget, while STABLEOPT's performance is good when the adversarial budget is known and poorly otherwise. Moreover, we argue that this parameter is difficult to specify in practice. In contrast, τ can be naturally chosen by domain experts, as in the insulin dosage example.

Chapter 6

Conclusion

This thesis has explored two directions for extending the classical multi-armed bandit (MAB) framework: vector optimization (VO), *i.e.*, vector-valued rewards under incomplete preferences, and robust satisficing (RS) under adversarial perturbations. While distinct in their problem settings and formulations, both directions share a central motivation: in realistic environments, decision-makers must go beyond conventional settings and instead embrace richer structures of preference and uncertainty.

In the first part, we developed *Pareto Vector Bandits* (PaVeBa). This pure-exploration algorithm addresses the challenge of identifying optimal arms when rewards are vector-valued and the partial ordering is user-specified through convex polyhedral cones. We also provided two heuristic variants of PaVeBa that adopt different modeling schemes, both employing Gaussian processes. With good empirical performance, we showed that, as multi-objective optimization (MOO) is a special case of vector optimization (VO), there is only a benefit in focusing on vector optimization. To support future work in this emerging area, we also introduced *VOPy*, a Python library that standardizes implementations of vector optimization algorithms for bandits. VOPy reproduces existing methods and provides a framework for developing and comparing new ones, lowering the barrier to entry for researchers in this field.

In the second part, we shifted focus to the robustness of solutions, specifically under adversarial perturbations. We argued that worst-case formulations, though principled, are often hard to configure in practice. By adopting the notion of robust satisficing and extending it with an empirical algorithm that employs Bayesian optimization using Thompson sampling, we demonstrated that a learner can achieve reliable performance under targeted perturbations through a simpler modeling approach. We believe that the satisficing framework should not be overlooked by optimization scholars, as it provides an alternative perspective on robust optimization and, in specific formulations, can even emerge as a dual problem to it.

Taken together, these contributions advance a common message: the boundaries of the MAB framework can be rethought. Real-world decision problems demand frameworks that account for multiple objectives, incomplete preferences, and uncertainty about the models themselves. This thesis demonstrates that such extensions can be both principled and practical, yielding algorithms better aligned with the realities of complex decision-making.

Bibliography

[1] İlter Onat Korkmaz, Yaşar Cahit Yıldırım, Çağın Ararat, and Cem Tekin. Vector optimization with gaussian process bandits. *arXiv preprint arXiv:2412.02484*, 2024.

[2] Artun Saday, Yaşar Cahit Yıldırım, and Cem Tekin. Robust satisficing gaussian process bandits under adversarial attacks. *arXiv preprint arXiv:2506.01625*, 2025.

[3] Michael TM Emmerich and André H Deutz. A tutorial on multiobjective optimization: fundamentals and evolutionary methods. *Natural Computing*, 17:585–609, 2018.

[4] Peter Auer, Chao-Kai Chiang, Ronald Ortner, and Madalina Drugan. Pareto front identification from stochastic bandit feedback. In *Proceedings of The 19th International Conference on Artificial Intelligence and Statistics*, pages 939–947, 2016.

[5] Marcela Zuluaga, Andreas Krause, and Markus Püschel. e-PAL: An active learning approach to the multi-objective optimization problem. *Journal of Machine Learning Research*, 17(104):1–32, 2016.

[6] Amar Shah and Zoubin Ghahramani. Pareto frontier learning with expensive correlated objectives. In *International Conference on Machine Learning*, pages 1919–1927. PMLR, 2016.

[7] Julian Katz-Samuels and Clay Scott. Feasible arm identification. In *International Conference on Machine Learning*, pages 2535–2543. PMLR, 2018.

[8] Daniel Hernández-Lobato, Jose Hernandez-Lobato, Amar Shah, and Ryan Adams. Predictive entropy search for multi-objective bayesian optimization. In *International Conference on Machine Learning*, pages 1492–1501. PMLR, 2016.

[9] Syrine Belakaria, Aryan Deshwal, and Janardhan Rao Doppa. Max-value entropy search for multi-objective bayesian optimization. *Advances in Neural Information Processing Systems*, 32, 2019.

[10] Ben Tu, Axel Gandy, Nikolas Kantas, and Behrang Shafei. Joint entropy search for multi-objective Bayesian optimization. In *Advances in Neural Information Processing Systems 35 (NeurIPS 2022)*, pages 9922–9938, 2022.

[11] Madalina M Drugan and Ann Nowe. Designing multi-objective multi-armed bandits algorithms: A study. In *International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2013.

[12] Eralp Turgay, Doruk Oner, and Cem Tekin. Multi-objective contextual bandit problem with similarity information. In *International Conference on Artificial Intelligence and Statistics*, pages 1673–1681. PMLR, 2018.

[13] Kristof Van Moffaert and Ann Nowé. Multi-objective reinforcement learning using sets of pareto dominating policies. *Journal of Machine Learning Research*, 15(1):3483–3512, 2014.

[14] Conor F Hayes, Roxana Rădulescu, Eugenio Bargiacchi, Johan Källström, Matthew Macfarlane, Mathieu Reymond, Timothy Verstraeten, Luisa M Zintgraf, Richard Dazeley, Fredrik Heintz, et al. A practical guide to multi-objective reinforcement learning and planning. *Autonomous Agents and Multi-Agent Systems*, 36(1):26, 2022.

[15] Johannes Jahn. *Vector Optimization: Theory, Applications, and Extensions*. Springer, 2nd edition, 2011.

[16] Andreas Löhne. *Vector optimization with infimum and supremum*. Springer Science & Business Media, 2011.

[17] Hans Meyer. Zur theorie der alkoholnarkose: Erste mittheilung. welche eigenschaft der anästhetica bedingt ihre narkotische wirkung? *Archiv für experimentelle Pathologie und Pharmakologie*, 42(2):109–118, 1899.

[18] Charles Ernest Overton. *Studien über die Narkose: zugleich ein Beitrag zur allgemeinen Pharmakologie*. G. Fischer, 1901.

[19] Peter Ertl and Ansgar Schuffenhauer. Estimation of synthetic accessibility score of drug-like molecules based on molecular complexity and fragment contributions. *Journal of Cheminformatics*, 1(1):8, 2009.

[20] David S Wishart, Yannick D Feunang, An C Guo, Elvis J Lo, Ana Marcu, Jason R Grant, Tanvir Sajed, Daniel Johnson, Carin Li, Zinat Sayeeda, et al. Drugbank 5.0: a major update to the drugbank database for 2018. *Nucleic Acids Research*, 46(D1):D1074–D1082, 2018.

[21] Kexin Huang, Tianfan Fu, Wenhao Gao, Yue Zhao, Yusuf Roohani, Jure Leskovec, Connor W Coley, Cao Xiao, Jimeng Sun, and Marinka Zitnik. Artificial intelligence foundation for therapeutic science. *Nature Chemical Biology*, 18(10):1033–1036, 2022.

[22] Harold J Kushner. A new method of locating the maximum point of an arbitrary multipeak curve in the presence of noise. *Journal of Basic Engineering*, 1964.

[23] Peter I Frazier. A tutorial on bayesian optimization. *arXiv preprint arXiv:1807.02811*, 2018.

[24] Christopher KI Williams and Carl Edward Rasmussen. *Gaussian processes for machine learning*, volume 2. MIT press Cambridge, MA, 2006.

[25] David Duvenaud. *Automatic model construction with Gaussian processes*. PhD thesis, 2014.

[26] Niranjan Srinivas, Andreas Krause, Sham M Kakade, and Matthias Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. *arXiv preprint arXiv:0912.3995*, 2009.

[27] William R Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3/4):285–294, 1933.

[28] Daniel J Russo, Benjamin Van Roy, Abbas Kazerouni, Ian Osband, Zheng Wen, et al. A tutorial on thompson sampling. *Foundations and Trends® in Machine Learning*, 11(1):1–96, 2018.

[29] Frank H Knight. Risk, uncertainty and profit. *Hart, Schaffner and Marx*, 1921.

[30] Moshe Sniedovich. Wald's maximin model: a treasure in disguise! *The Journal of Risk Finance*, 9(3):287–291, 2008.

[31] Artun Saday, Y Cahit Yıldırım, and Cem Tekin. Robust bayesian satisficing. *Advances in Neural Information Processing Systems*, 36:69253–69269, 2023.

[32] Herbert A Simon. A behavioral model of rational choice. *The quarterly journal of economics*, pages 99–118, 1955.

[33] Herbert A Simon. Rational choice and the structure of the environment. *Psychological review*, 63(2):129, 1956.

[34] Paul Reverdy, Vaibhav Srivastava, and Naomi Ehrich Leonard. Satisficing in multi-armed bandit problems. *IEEE Transactions on Automatic Control*, 62(8):3788–3803, 2016.

[35] Daniel Russo and Benjamin Van Roy. Satisficing in time-sensitive bandit learning. *Mathematics of Operations Research*, 47(4):2815–2839, 2022.

[36] Nadav Merlis and Shie Mannor. Lenient regret for multi-armed bandits. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 8950–8957, 2021.

[37] Alihan Hüyük and Cem Tekin. Multi-objective multi-armed bandit with lexicographically ordered and satisficing objectives. *Machine Learning*, 110(6):1233–1266, 2021.

[38] Daniel Zhuoyu Long, Melvyn Sim, and Minglong Zhou. Robust satisficing. *Operations Research*, 71(1):61–82, 2023.

[39] Miriam Zacksenhouse, Simona Nemets, Mikhail A Lebedev, and Miguel AL Nicolelis. Robust satisficing linear regression: performance/robustness trade-off and consistency criterion. *Mechanical Systems and Signal Processing*, 23(6):1954–1964, 2009.

[40] Yakov Ben-Haim. *Info-gap decision theory: decisions under severe uncertainty*. Elsevier, 2006.

[41] Zohar Karnin, Tomer Koren, and Oren Somekh. Almost optimal exploration in multi-armed bandits. In *International Conference on Machine Learning*, pages 1238–1246. PMLR, 2013.

[42] Aurélien Garivier and Emilie Kaufmann. Optimal best arm identification with fixed confidence. In *Conference on Learning Theory*, pages 998–1027. PMLR, 2016.

[43] Eyal Even-Dar, Shie Mannor, Yishay Mansour, and Sridhar Mahadevan. Action elimination and stopping conditions for the multi-armed bandit and reinforcement learning problems. *Journal of Machine Learning Research*, 7(6), 2006.

[44] Çağın Ararat and Cem Tekin. Vector optimization with stochastic bandit feedback. In *Proceedings of The 26th International Conference on Artificial Intelligence and Statistics*, pages 2165–2190, 2023.

[45] Efe Mert Karagözlü, Yaşar Cahit Yıldırım, Çağın Ararat, and Cem Tekin. Learning the Pareto set under incomplete preferences: Pure exploration in vector bandits. In *Proceedings of The 27th International Conference on Artificial Intelligence and Statistics*, pages 3070–3078, 2024.

[46] Chi Jin, Praneeth Netrapalli, Rong Ge, Sham M Kakade, and Michael I Jordan. A short note on concentration inequalities for random vectors with subgaussian norm. *arXiv preprint arXiv:1902.03736*, 2019.

[47] Edwin V Bonilla, Kian Chai, and Christopher Williams. Multi-task gaussian process prediction. *Advances in Neural Information Processing Systems*, 20, 2007.

[48] Emile Contal, David Buffoni, Alexandre Robicquet, and Nicolas Vayatis. Parallel gaussian process optimization with upper confidence bound and pure exploration. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 225–240. Springer, 2013.

[49] Maximilian Balandat, Brian Karrer, Daniel R. Jiang, Samuel Daulton, Benjamin Letham, Andrew Gordon Wilson, and Eytan Bakshy. BoTorch: a framework for efficient Monte-Carlo Bayesian optimization. In *Advances in Neural Information Processing Systems 33 (NeurIPS 2020)*, pages 21524–21538, 2020.

[50] Marcela Zuluaga, Peter Milder, and Markus Püschel. Computer generation of streaming sorting networks. In *Proceedings of the 49th Annual Design Automation Conference*, pages 1245–1253, 2012.

[51] Ryoji Tanabe and Hisao Ishibuchi. An easy-to-use real-world multi-objective optimization problem suite. *Applied Soft Computing*, 89:106078, 2020.

[52] Jason S Moore and Klavs F Jensen. Automated multitrajectory method for reaction optimization in a microfluidic system using online ir analysis. *Organic Process Research & Development*, 16(8):1409–1415, 2012.

[53] Xingtao Liao, Qing Li, Xujing Yang, Weigang Zhang, and Wei Li. Multiobjective optimization for crash safety design of vehicles using stepwise regression model. *Structural and Multidisciplinary Optimization*, 35:561–569, 2008.

[54] Michael G Parsons and Randall L Scott. Formulation of multicriterion design optimization problems for solution with scalar numerical optimization methods. *Journal of Ship Research*, 48(01):61–76, 2004.

[55] Jia Wu, Xiu-Yun Chen, Hao Zhang, Li-Dong Xiong, Hang Lei, and Si-Hao Deng. Hyperparameter optimization for machine learning models based

on Bayesian optimization. *Journal of Electronic Science and Technology*, 17(1):26–40, 2019.

[56] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical Bayesian optimization of machine learning algorithms. *Advances in Neural Information Processing Systems 25 (NeurIPS 2012)*, pages 2951–2959, 2012.

[57] Kirthevasan Kandasamy, Willie Neiswanger, Jeff Schneider, Barnabás Póczos, and Eric P. Xing. Neural architecture search with Bayesian optimisation and optimal transport. In *Advances in Neural Information Processing Systems 31 (NeurIPS 2018)*, pages 2020–2029, 2018.

[58] Matthew Hoffman, Eric Brochu, Nando De Freitas, et al. Portfolio allocation for Bayesian optimization. In *Proceedings of the 27th Conference on Uncertainty in Artificial Intelligence*, pages 327–336, 2011.

[59] Javier Gonzalez, Joseph Longworth, David C James, and Neil D Lawrence. Bayesian optimization for synthetic gene design. arXiv preprint 1505.01627, 2015.

[60] Kei Terayama, Masato Sumita, Ryo Tamura, and Koji Tsuda. Black-box optimization for automated discovery. *Accounts of Chemical Research*, 54(6):1334–1346, 2021.

[61] Atsuto Seko, Atsushi Togo, Hiroyuki Hayashi, Koji Tsuda, Laurent Chaput, and Isao Tanaka. Prediction of low-thermal-conductivity compounds with first-principles anharmonic lattice-dynamics calculations and Bayesian optimization. *Physical Review Letters*, 115(20):205901, 2015.

[62] Artem R Oganov and Colin W Glass. Crystal structure prediction using ab initio evolutionary techniques: principles and applications. *The Journal of Chemical Physics*, 124(24), 2006.

[63] Tsuyoshi Ueno, Trevor David Rhone, Zhufeng Hou, Teruyasu Mizoguchi, and Koji Tsuda. COMBO: An efficient Bayesian optimization library for materials science. *Materials Discovery*, 4:18–21, 2016.

[64] Alberto Costa and Giacomo Nannicini. RBFOpt: an open-source library for black-box optimization with costly function evaluations. *Mathematical Programming Computation*, 10:597–629, 2018.

[65] Daniel Golovin, Benjamin Solnik, Subhodeep Moitra, Greg Kochanski, John Karro, and David Sculley. Google Vizier: a service for black-box optimization. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1487–1495, 2017.

[66] Huaijun Jiang, Yu Shen, Yang Li, Beicheng Xu, Sixian Du, Wentao Zhang, Ce Zhang, and Bin Cui. Openbox: a Python toolkit for generalized black-box optimization. *Journal of Machine Learning Research*, 25(120):1–11, 2024.

[67] Julian Blank and Kalyanmoy Deb. Pymoo: Multi-objective optimization in Python. *IEEE Access*, 8:89497–89509, 2020.

[68] Kevin Maik Jablonka, Giriprasad Melpatti Jothiappan, Shefang Wang, Berend Smit, and Brian Yoo. Bias free multiobjective active learning for materials design and discovery. *Nature Communications*, 12(1):2312, 2021.

[69] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 2623–2631, 2019.

[70] Guido van Rossum, Barry Warsaw, and Nick Coghlan. Style guide for Python code. PEP 8, Python Software Foundation, 2001.

[71] Vincent Grari, Thibault Laugel, Tatsunori Hashimoto, sylvain lamprier, and Marcin Detyniecki. On the fairness ROAD: Robust optimization for adversarial debiasing. In *The Twelfth International Conference on Learning Representations*, 2024.

[72] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

[73] Andy Zou, Zifan Wang, Nicholas Carlini, Milad Nasr, J Zico Kolter, and Matt Fredrikson. Universal and transferable adversarial attacks on aligned language models. *arXiv preprint arXiv:2307.15043*, 2023.

[74] Lee H Keel, SP Bhattacharyya, and Jo W Howze. Robust control with structure perturbations. *IEEE Transactions on Automatic Control*, 33(1):68–78, 2002.

[75] William J Hurley and Jack Brimberg. A note on the sensitivity of the strategic asset allocation problem. *Operations Research Perspectives*, 2:133–136, 2015.

[76] Ilija Bogunovic, Jonathan Scarlett, Stefanie Jegelka, and Volkan Cevher. Adversarially robust optimization with gaussian processes. *Advances in Neural Information Processing Systems*, 31, 2018.

[77] Boris P Kovatchev, Marc Breton, Chiara Dalla Man, and Claudio Cobelli. In silico preclinical trials: A proof of concept in closed-loop control of type 1 diabetes. *Journal of Diabetes Science and Technology*, 2009.

[78] Thomas Danne, Moshe Phillip, Bruce A Buckingham, Przemyslawa Jarosz-Chobot, Banshi Saboo, Tatsuhiko Urakami, Tadej Battelino, Ragnar Hanas, and Ethel Codner. Ispad clinical practice consensus guidelines 2018: Insulin treatment in children and adolescents with diabetes. *Pediatric Diabetes*, 19:115–135, 2018.

[79] Chiara Zecchin, Andrea Facchinetti, Giovanni Sparacino, and Claudio Cobelli. How much is short-term glucose prediction in type 1 diabetes improved by adding insulin delivery and meal content information to cgm data? a proof-of-concept study. *Journal of Diabetes Science and Technology*, 10(5):1149–1160, 2016.

[80] Tor Lattimore and Csaba Szepesvári. *Bandit algorithms*. Cambridge University Press, 2020.

Appendix A

This is the appendix to the chapter related to the works in "Learning the Pareto Set Under Incomplete Preferences: Pure Exploration in Vector Bandits", *i.e.*, Chapter 3.

A.1 Proofs for the Convex Programming Formulations

In this section, we provide the proofs of Propositions 1 and 2.

A.1.1 Proof of Proposition 1

Note that $\mathcal{E}_t(x)$, $\mathcal{E}_t(y)$ are compact sets and $M(\cdot,\cdot)$ is a continuous function. Hence, the supremum under consideration is indeed a maximum. Then, $\sup_{\boldsymbol{\mu}\in\mathcal{E}_t(x),\boldsymbol{\nu}\in\mathcal{E}_t(y)} M(\boldsymbol{\mu},\boldsymbol{\nu}) > 0$ iff there exist $\boldsymbol{\mu} \in \mathcal{E}_t(x)$, $\boldsymbol{\nu} \in \mathcal{E}_t(y)$ such that $\boldsymbol{\nu}-\boldsymbol{\mu} \in C^c$ by [44, Corollary 4.5]. By the definition of cone C, this is possible iff $\boldsymbol{w}_n^\mathsf{T}(\boldsymbol{\nu}-\boldsymbol{\mu}) < 0$ for some $n \in [N]$, $\boldsymbol{\mu} \in \mathcal{E}_t(x)$, and $\boldsymbol{\nu} \in \mathcal{E}_t(y)$. In other words, for at least one $n \in [N]$, we have $\min_{\boldsymbol{\mu}\in\mathcal{E}_t(x),\boldsymbol{\nu}\in\mathcal{E}_t(y)} \boldsymbol{w}_n^\mathsf{T}(\boldsymbol{\nu}-\boldsymbol{\mu}) < 0$, which corresponds precisely to the problem defined in Proposition 1. □

A.1.2 Proof of Proposition 2

Note that $\mathcal{E}_t(x)$, $\mathcal{E}_t(y)$ are compact sets and $m(\cdot,\cdot)$ is a continuous function. Hence, the supremum under consideration is indeed a maximum. Then, $\sup_{\boldsymbol{\mu}\in\mathcal{E}_t(x),\boldsymbol{\nu}\in\mathcal{E}_t(y)} m(\boldsymbol{\mu},\boldsymbol{\nu}) \geq \epsilon$ iff there exist $\boldsymbol{\mu} \in \mathcal{E}_t(x)$, $\boldsymbol{\nu} \in \mathcal{E}_t(y)$ such that $m(\boldsymbol{\mu},\boldsymbol{\nu}) \geq \epsilon$. By [44, Proposition 4.2(iv)], we have

$$m(\boldsymbol{\mu},\boldsymbol{\nu}) = \min_{n\in[N]} \frac{(\boldsymbol{w}_n^\mathsf{T}(\boldsymbol{\nu}-\boldsymbol{\mu}))^+}{\alpha_n}.$$

Then,

$$m(\boldsymbol{\mu},\boldsymbol{\nu}) \geq \epsilon \quad \Leftrightarrow \quad \forall n \in [N]\colon (\boldsymbol{w}_n^\mathsf{T}(\boldsymbol{\nu}-\boldsymbol{\mu}))^+ \geq \epsilon\alpha_n.$$

Since $\epsilon\alpha_n > 0$ for each $n \in [N]$, we can drop the positive part function and obtain

$$m(\boldsymbol{\mu},\boldsymbol{\nu}) \geq \epsilon \quad \Leftrightarrow \quad \forall n \in [N]\colon \boldsymbol{w}_n^\mathsf{T}(\boldsymbol{\nu}-\boldsymbol{\mu}) \geq \epsilon\alpha_n.$$

Therefore, $\sup_{\boldsymbol{\mu}\in\mathcal{E}_t(x),\boldsymbol{\nu}\in\mathcal{E}_t(y)} m(\boldsymbol{\mu},\boldsymbol{\nu}) \geq \epsilon$ iff there exist $\boldsymbol{\mu} \in \mathcal{E}_t(x)$, $\boldsymbol{\nu} \in \mathcal{E}_t(y)$ such that $\boldsymbol{w}_n^\mathsf{T}(\boldsymbol{\nu}-\boldsymbol{\mu}) \geq \epsilon\alpha_n$ holds for each $n \in [N]$. These conditions are precisely the constraints of the feasibility problem in the proposition. Hence, the result follows. □

A.2 Implementation of PaVeBa When $\epsilon = 0$

The problem with PaVeBa when we have $\epsilon = 0$ is twofold: (i) As pointed out in Remark 1, Proposition 2 does not work when $\epsilon = 0$ due to the positive part for the computation of $m(\cdot,\cdot)$ in [44, Proposition 4.2] making all $m(\cdot,\cdot)$ nonnegative, and (ii) from the definitions in (3.7) and (3.8), we would have

$$\overline{\mathcal{P}}_t = \{x \in \overline{\mathcal{S}}_t \,|\, \forall y \in \overline{\mathcal{A}}_t \setminus \{x\}\colon \sup_{\boldsymbol{\mu}\in\mathcal{E}_t(x),\boldsymbol{\nu}\in\mathcal{E}_t(y)} m(\boldsymbol{\mu},\boldsymbol{\nu}) < 0\} = \emptyset$$
$$\implies \mathcal{P}_{t+1} = \mathcal{P}_t,\ \mathcal{S}_{t+1} = \overline{\mathcal{S}}_t = \mathcal{S}_t \setminus \mathcal{D}_t \text{ (PaVeBa lines 11 and 9).}$$

Since $\mathcal{P}_1 = \emptyset$, the above iteration will result in $\mathcal{P}_t = \emptyset$ for all t. Therefore,

$$\mathcal{U}_{t+1} = \{y \in \mathcal{P}_{t+1} \,|\, \exists x \in \mathcal{S}_{t+1}\colon \sup_{\substack{\boldsymbol{\mu}\in\mathcal{E}_t(x),\\ \boldsymbol{\nu}\in\mathcal{E}_t(y)}} m(\boldsymbol{\mu},\boldsymbol{\nu}) \geq 0\} = \emptyset, \forall t\ .$$

Then, by line 4 of PaveBa, $\mathcal{A}_t = \mathcal{S}_t$ for all t.

To conclude, PaVeBa samples all undecided arms in all rounds. PaVeBa can only discard arms, but it will never declare an arm as Pareto optimal. Since there are Pareto optimal arms in the undecided set at the beginning, it will never terminate.

To circumvent these, we offer a slight change in the definitions of $\overline{\mathcal{P}}_t$ and $\mathcal{U}_{t+1}$ as follows:

$$\overline{\mathcal{P}}_t = \{x \in \overline{\mathcal{S}}_t \,|\, \forall y \in \overline{\mathcal{A}}_t \setminus \{x\} : \sup_{\substack{\boldsymbol{\mu}\in\mathcal{E}_t(x),\\ \boldsymbol{\nu}\in\mathcal{E}_t(y)}} m(\boldsymbol{\mu},\boldsymbol{\nu}) \leq 0\}, \tag{A.1}$$

$$\mathcal{U}_{t+1} = \{y \in \mathcal{P}_{t+1} \,|\, \exists x \in \mathcal{S}_{t+1} : \sup_{\substack{\boldsymbol{\mu}\in\mathcal{E}_t(x),\\ \boldsymbol{\nu}\in\mathcal{E}_t(y)}} m(\boldsymbol{\mu},\boldsymbol{\nu}) > 0\}. \tag{A.2}$$

Let the new variant of PaVeBa using the sets defined in (A.1) and (A.2) be PaVeBa-0. Note that all the algorithm steps remain the same with PaVeBa except these two slight changes in the definitions of $\overline{\mathcal{P}}_t$ and $\mathcal{U}_{t+1}$. We now present the sample complexity analysis for PaVeBa-0.

For the sample complexity of PaVeBa when $\epsilon = 0$, please refer to [45].

A.2.1 Efficient Implementation of PaVeBa-0 via Convex Programming

We now need to find another convex programming method to compute the updated sets in (A.1) and (A.2).

Proposition 3. *Given a pair of arms x, y and associated confidence regions $\mathcal{E}_t(x), \mathcal{E}_t(y)$; $\sup_{\boldsymbol{\mu}\in\mathcal{E}_t(x),\boldsymbol{\nu}\in\mathcal{E}_t(y)} m(\boldsymbol{\mu},\boldsymbol{\nu}) \leq 0$ if and only if the optimal value of the*

following feasibility problem is $+\infty$, i.e., it yields infeasibility:

$$\begin{aligned}
&\textit{minimize} \quad 0 \\
&\textit{subject to} \quad \boldsymbol{w}_n^{\mathsf{T}}(\boldsymbol{\mu}_y - \boldsymbol{\mu}) \geq 0,\ \boldsymbol{w}_n^{\mathsf{T}}(\boldsymbol{\mu} - \boldsymbol{\mu}_x) \geq 0, \forall n \in [N], \\
&\qquad\qquad (\boldsymbol{\mu}_x - \boldsymbol{\mu}_\tau(x))^{\mathsf{T}}\boldsymbol{\Sigma}_\tau^{-1}(x)(\boldsymbol{\mu}_x - \boldsymbol{\mu}_\tau(x)) \leq \alpha_t, \forall \tau \in [t], \\
&\qquad\qquad (\boldsymbol{\mu}_y - \boldsymbol{\mu}_\tau(y))^{\mathsf{T}}\boldsymbol{\Sigma}_\tau^{-1}(y)(\boldsymbol{\mu}_y - \boldsymbol{\mu}_\tau(y)) \leq \alpha_t, \forall \tau \in [t], \\
&\qquad\qquad \boldsymbol{\mu}, \boldsymbol{\mu}_x, \boldsymbol{\mu}_y \in \mathbb{R}^D.
\end{aligned}$$

Proof. Given two vectors $\boldsymbol{\mu}_x, \boldsymbol{\mu}_y \in \mathbb{R}^D$, observe that

$$\boldsymbol{\mu}_y - \boldsymbol{\mu}_x \notin C \quad \Leftrightarrow \quad \boldsymbol{\mu}_y \notin \boldsymbol{\mu}_x + C \quad \Leftrightarrow \quad (\boldsymbol{\mu}_y - C) \cap (\boldsymbol{\mu}_x + C) = \emptyset.$$

Then, we may write

$$\begin{aligned}
&\{\forall \boldsymbol{\mu}_x \in \mathcal{E}_t(x),\ \forall \boldsymbol{\mu}_y \in \mathcal{E}_t(y) \colon \boldsymbol{\mu}_y - \boldsymbol{\mu}_x \notin C\} \\
&= \{\forall \boldsymbol{\mu}_x \in \mathcal{E}_t(x),\ \forall \boldsymbol{\mu}_y \in \mathcal{E}_t(y) \colon (\boldsymbol{\mu}_y - C) \cap (\boldsymbol{\mu}_x + C) = \emptyset\} \\
&= \left\{ \left[\bigcup_{\boldsymbol{\mu}_x \in \mathcal{E}_t(x)} (\boldsymbol{\mu}_y - C) \right] \cap \left[\bigcup_{\boldsymbol{\mu}_y \in \mathcal{E}_t(y)} (\boldsymbol{\mu}_x + C) \right] = \emptyset \right\} \\
&= \{(\mathcal{E}_t(y) - C) \cap (\mathcal{E}_t(x) + C) = \emptyset\}\ .
\end{aligned}$$

Let $x, y \in \mathcal{X}$ with $x \neq y$. In order to check if $(\mathcal{E}_t(y) - C) \cap (\mathcal{E}_t(x) + C) = \emptyset$, let us consider the following feasibility problem expressed in the form of a mathematical program:

$$\begin{aligned}
&\text{minimize} \quad 0 \\
&\text{subject to} \quad \boldsymbol{\mu} \in \mathcal{E}_t(y) - C, \quad \boldsymbol{\mu} \in \mathcal{E}_t(x) + C, \quad \boldsymbol{\mu} \in \mathbb{R}^D.
\end{aligned}$$

This problem can be rewritten more explicitly as

$$\begin{aligned}
&\text{minimize} \quad 0 \\
&\text{subject to} \quad \boldsymbol{\mu}_y - \boldsymbol{\mu} \in C, \quad \boldsymbol{\mu} - \boldsymbol{\mu}_x \in C, \\
&\qquad\qquad \boldsymbol{\mu}_x \in \mathcal{E}_t(x), \quad \boldsymbol{\mu}_y \in \mathcal{E}_t(y), \\
&\qquad\qquad \boldsymbol{\mu}, \boldsymbol{\mu}_x, \boldsymbol{\mu}_y \in \mathbb{R}^D,
\end{aligned}$$

which is equivalent to

$$\begin{aligned}
&\text{minimize} \quad 0 \\
&\text{subject to} \quad \boldsymbol{w}_n^\mathsf{T}(\boldsymbol{\mu}_y - \boldsymbol{\mu}) \geq 0, \quad \boldsymbol{w}_n^\mathsf{T}(\boldsymbol{\mu} - \boldsymbol{\mu}_x) \geq 0, \quad \forall n \in [N], \\
&\qquad (\boldsymbol{\mu}_x - \boldsymbol{\mu}_t(x))^\mathsf{T}\boldsymbol{\Sigma}_t^{-1}(x)(\boldsymbol{\mu}_x - \boldsymbol{\mu}_t(x)) \leq \alpha_t, \forall \tau \in [t], \\
&\qquad (\boldsymbol{\mu}_y - \boldsymbol{\mu}_t(y))^\mathsf{T}\boldsymbol{\Sigma}_t^{-1}(y)(\boldsymbol{\mu}_y - \boldsymbol{\mu}_t(y)) \leq \alpha_t, \forall \tau \in [t], \\
&\qquad \boldsymbol{\mu}, \boldsymbol{\mu}_x, \boldsymbol{\mu}_y \in \mathbb{R}^D.
\end{aligned}$$

This is a convex optimization (feasibility) problem with affine and quadratic constraints. We have $(\mathcal{E}_t(y) - C) \cap (\mathcal{E}_t(x) + C) = \emptyset$ if and only if the problem yields $+\infty$ as its optimal value, i.e., the problem is infeasible. Otherwise, the optimal value of the problem is zero, and the empty intersection property does not hold for x, y. □

A.3 Derivation of Confidence Regions for Gaussian Processes

We include this analysis to make our algorithm compatible with the use of GPs for improved regression in the experiments. We assume a multi-output GP with possibly correlated objectives. Then, we take our single-round confidence regions to be

$$\mathcal{B}_t(x) = \{\boldsymbol{\nu} : (\boldsymbol{\nu} - \boldsymbol{\mu}_t(x))^\mathsf{T}\boldsymbol{\Sigma}_t^{-1}(x)(\boldsymbol{\nu} - \boldsymbol{\mu}_t(x)) \leq \alpha_t\}, \tag{A.3}$$

where $\alpha_t = 8D\log(12) + 4\log(\frac{\pi^2 t^2 |\mathcal{X}|}{6\delta})$, $\boldsymbol{\mu}_t(x)$ is the posterior mean of the GP of arm x at round t, and $\boldsymbol{\Sigma}_t(x)$ is the posterior covariance matrix of the GP for arm x at round t. Further, we define

$$\mathcal{E}_t(x) \coloneqq \mathcal{E}_{t-1}(x) \cap \mathcal{B}_t(x) \tag{A.4}$$

with $\mathcal{E}_1(x) \coloneqq \mathcal{B}_1(x)$ as the confidence region of x at round t. (Note that the intersection of confidence regions is a technical detail for the proofs to work in the sample complexity analysis. However, it happens rarely in practice for $\mathcal{B}_{t+1}(x)$

not to be a subset of $\mathcal{B}_t(x)$; hence, we dropped intersecting regions $\mathcal{B}_\tau(x)$ for $\tau \leq t$ in the experiments where heuristic variants are used.)

We start with a covering lemma that is a refinement of [80, Lemma 20.1].

Lemma 1. *For every $\varepsilon \in (0, 2)$, there exists a set $\mathcal{C}_\varepsilon \subseteq \mathbb{S}^{D-1}$ such that $|\mathcal{C}_\varepsilon| \leq (\frac{6}{\varepsilon})^D$ and*

$$\forall \boldsymbol{w} \in \mathbb{S}^{D-1}\ \exists \tilde{\boldsymbol{w}} \in \mathcal{C}_\varepsilon\colon\ \|\boldsymbol{w} - \tilde{\boldsymbol{w}}\|_2 \leq \varepsilon.$$

Proof. By [80, Lemma 20.1], for every $\varepsilon > 0$, there exists a set $\tilde{\mathcal{C}}_\varepsilon \subseteq \mathbb{R}^D$ such that $|\tilde{\mathcal{C}}_\varepsilon| \leq (\frac{3}{\varepsilon})^D$ and

$$\forall \boldsymbol{w} \in \mathbb{S}^{D-1}\ \exists \tilde{\boldsymbol{w}} \in \tilde{\mathcal{C}}_\varepsilon\colon\ \|\boldsymbol{w} - \tilde{\boldsymbol{w}}\|_2 \leq \varepsilon.$$

Moreover, without loss of generality, we assume that

$$\forall \tilde{\boldsymbol{w}} \in \tilde{\mathcal{C}}_\varepsilon\ \exists \boldsymbol{w} \in \mathbb{S}^{D-1}\colon\ \|\boldsymbol{w} - \tilde{\boldsymbol{w}}\| \leq \varepsilon.$$

as otherwise one can remove from $\tilde{\mathcal{C}}_\varepsilon$ the elements $\tilde{\boldsymbol{w}}$ for which there is no $\boldsymbol{w} \in \mathbb{S}^{D-1}$ with $\|\boldsymbol{w} - \tilde{\boldsymbol{w}}\|_2 \leq \varepsilon$ and the new set still satisfies the two conditions that $\tilde{\mathcal{C}}_\varepsilon$ satisfies.

We claim that $\tilde{\mathcal{C}}_\varepsilon \subseteq B(\mathbf{0}, 1 + \varepsilon) \setminus \operatorname{int} B(\mathbf{0}, 1 - \varepsilon)$ whenever $\varepsilon \in (0, 1)$. Let $\tilde{\boldsymbol{w}} \in \tilde{\mathcal{C}}_\varepsilon$. By the assumption above, there exists $\boldsymbol{w} \in \mathbb{S}^{D-1}$ such that $\|\boldsymbol{w} - \tilde{\boldsymbol{w}}\|_2 \leq \varepsilon$. Then, by triangle inequality, we have

$$\|\tilde{\boldsymbol{w}}\|_2 \leq \|\tilde{\boldsymbol{w}} - \boldsymbol{w}\|_2 + \|\boldsymbol{w}\|_2 \leq \varepsilon + 1.$$

Hence, $\tilde{\boldsymbol{w}} \in B(\mathbf{0}, 1 + \varepsilon)$. Moreover, by reverse triangle inequality, we have

$$\begin{aligned}\|\tilde{\boldsymbol{w}}\|_2 = \|(\tilde{\boldsymbol{w}} - \boldsymbol{w}) - (-\boldsymbol{w})\|_2 \geq |\|\tilde{\boldsymbol{w}} - \boldsymbol{w}\|_2 - \|-\boldsymbol{w}\|_2| = |\|\tilde{\boldsymbol{w}} - \boldsymbol{w}\|_2 - 1| \\ \geq 1 - \|\tilde{\boldsymbol{w}} - \boldsymbol{w}\|_2 \geq 1 - \varepsilon.\end{aligned}$$

Hence, $\tilde{\boldsymbol{w}} \notin \operatorname{int} B(\mathbf{0}, 1 - \varepsilon)$, which completes the proof of the claim.

Let us fix $\varepsilon \in (0, 2)$ and define

$$\mathcal{C}_\varepsilon \coloneqq \left\{ \frac{\tilde{\boldsymbol{w}}}{\|\tilde{\boldsymbol{w}}\|_2} \colon \tilde{\boldsymbol{w}} \in \tilde{\mathcal{C}}_{\frac{\varepsilon}{2}} \right\} \subseteq \mathbb{S}^{D-1}.$$

Note that $|\mathcal{C}_\varepsilon| \le |\tilde{\mathcal{C}}_{\frac{\varepsilon}{2}}| \le (\frac{6}{\varepsilon})^D$. Let $\boldsymbol{w} \in \mathbb{S}^{D-1}$. Then, there exists $\tilde{\boldsymbol{w}} \in \tilde{\mathcal{C}}_{\frac{\varepsilon}{2}}$ such that

$$\|\boldsymbol{w} - \tilde{\boldsymbol{w}}\|_2 \le \frac{\varepsilon}{2}.$$

Then, we have

$$\left\|\boldsymbol{w} - \frac{\tilde{\boldsymbol{w}}}{\|\tilde{\boldsymbol{w}}\|_2}\right\|_2 \le \|\boldsymbol{w} - \tilde{\boldsymbol{w}}\|_2 + \left\|\tilde{\boldsymbol{w}} - \frac{\tilde{\boldsymbol{w}}}{\|\tilde{\boldsymbol{w}}\|_2}\right\|_2 \le \frac{\varepsilon}{2} + |1 - \|\tilde{\boldsymbol{w}}\|_2|.$$

Moreover, since $\tilde{\boldsymbol{w}} \in B(\boldsymbol{0}, 1 + \frac{\varepsilon}{2}) \setminus \operatorname{int} B(\boldsymbol{0}, 1 - \frac{\varepsilon}{2})$, we have

$$-\frac{\varepsilon}{2} \le 1 - \|\tilde{\boldsymbol{w}}\|_2 \le \frac{\varepsilon}{2},$$

that is, $|1 - \|\tilde{\boldsymbol{w}}\|_2| \le \frac{\varepsilon}{2}$. It follows that

$$\left\|\boldsymbol{w} - \frac{\tilde{\boldsymbol{w}}}{\|\tilde{\boldsymbol{w}}\|_2}\right\|_2 \le \frac{\varepsilon}{2} + \frac{\varepsilon}{2} = \varepsilon.$$

This completes the proof since $\frac{\tilde{\boldsymbol{w}}}{\|\tilde{\boldsymbol{w}}\|_2} \in \mathcal{C}_\varepsilon$. □

Now, we present a lemma showing our choice of confidence regions indeed gives at least $1 - \delta$ confidence.

Lemma 2. *Using the definitions in* (A.3) *and* (A.4)*, we have* $\mathbb{P}\{\forall x \in \mathcal{X}, \forall t \in \mathbb{N}\colon \boldsymbol{f}(x) \in \mathcal{E}_t(x)\} \ge 1 - \delta$.

Proof. Since $\mathcal{E}_t(x) = \bigcap_{\tau=1}^{t} \mathcal{B}_\tau(x)$ for each $t \in [T]$, we have

$$\{\forall x \in \mathcal{X}, \forall t \in \mathbb{N}\colon \boldsymbol{f}(x) \in \mathcal{E}_t(x)\} = \{\forall x \in \mathcal{X}, \forall t \in \mathbb{N}\colon \boldsymbol{f}(x) \in \mathcal{B}_t(x)\}.$$

So, instead, we prove $\mathbb{P}\{\forall x \in \mathcal{X}, \forall t \in \mathbb{N}\colon \boldsymbol{f}(x) \in \mathcal{B}_t(x)\} \ge 1 - \delta$.

Let $\mathcal{F}_t$ be the information at round t. The conditional distribution of $\boldsymbol{f}(x)$ given $\mathcal{F}_t$ is the Gaussian distribution with mean vector $\boldsymbol{\mu}_t(x)$ and covariance matrix $\boldsymbol{\Sigma}_t(x)$. Thus, for each $\boldsymbol{w} \in \mathbb{R}^D \setminus \{\boldsymbol{0}\}$, the conditional distribution of $\boldsymbol{w}^\mathsf{T}\boldsymbol{f}(x)$ is the Gaussian distribution with mean $\boldsymbol{w}^\mathsf{T}\boldsymbol{\mu}_t(x)$ and variance $\boldsymbol{w}^\mathsf{T}\boldsymbol{\Sigma}_t(x)\boldsymbol{w}$. In particular, applying Gaussian concentration gives for $\tilde{\alpha} > 0$

$$\begin{aligned}\mathbb{P}\left\{\boldsymbol{w}^\mathsf{T}\boldsymbol{f}(x) > \boldsymbol{w}^\mathsf{T}\boldsymbol{\mu}_t(x) + \sqrt{\tilde{\alpha}\boldsymbol{w}^\mathsf{T}\boldsymbol{\Sigma}_t(x)\boldsymbol{w}}\right\} &= \mathbb{P}\left\{\frac{\boldsymbol{w}^\mathsf{T}(\boldsymbol{f}(x) - \boldsymbol{\mu}_t(x))}{\sqrt{\boldsymbol{w}^\mathsf{T}\boldsymbol{\Sigma}_t(x)\boldsymbol{w}}} > \sqrt{\tilde{\alpha}}\right\} \\ &\le e^{-\frac{\tilde{\alpha}}{2}}. \end{aligned} \tag{A.5}$$

Moreover, the ellipsoid $\mathcal{B}_t(x)$ is a closed convex set. Hence, an application of separation theorem in convex analysis yields

$$\mathbb{P}\{\boldsymbol{f}(x) \notin \mathcal{B}_t(x)\} = \mathbb{P}\left\{\exists \boldsymbol{w} \in \mathbb{S}^{D-1} \colon \boldsymbol{w}^\mathsf{T}\boldsymbol{f}(x) > \sup_{\boldsymbol{\mu}_x \in \mathcal{B}_t(x)} \boldsymbol{w}^\mathsf{T}\boldsymbol{\mu}_x\right\}.$$

Here, note that

$$\begin{aligned}\mathcal{B}_t(x) &= \{\boldsymbol{\mu}_x \in \mathbb{R}^D \mid (\boldsymbol{\mu}_x - \boldsymbol{\mu}_t(x))^\mathsf{T}(\alpha_t\boldsymbol{\Sigma}_t(x))^{-1}(\boldsymbol{\mu}_x - \boldsymbol{\mu}_t(x)) \leq 1\} \\ &= \boldsymbol{\mu}_t(x) + \sqrt{\alpha_t}\boldsymbol{\Sigma}_t^{1/2}(x)B(\mathbf{0}, 1)\ .\end{aligned}$$

Hence,

$$\sup_{\boldsymbol{\mu}_x \in \mathcal{B}_t(x)} \boldsymbol{w}^\mathsf{T}\boldsymbol{\mu}_x = \boldsymbol{w}^\mathsf{T}\boldsymbol{\mu}_t(x) + \sqrt{\alpha_t \boldsymbol{w}^\mathsf{T}\boldsymbol{\Sigma}_t(x)\boldsymbol{w}}\ . \tag{A.6}$$

Using (A.6) we get

$$\mathbb{P}\{\boldsymbol{f}(x) \notin \mathcal{B}_t(x)\} = \mathbb{P}\left\{\exists \boldsymbol{w} \in \mathbb{S}^{D-1} \colon \boldsymbol{w}^\mathsf{T}\boldsymbol{f}(x) > \boldsymbol{w}^\mathsf{T}\boldsymbol{\mu}_t(x) + \sqrt{\alpha_t \boldsymbol{w}^\mathsf{T}\boldsymbol{\Sigma}_t(x)\boldsymbol{w}}\right\}. \tag{A.7}$$

Let $\varepsilon \in (0, 2)$ and $\tilde{\alpha} > 0$. By Lemma 1, there exists a set $\mathcal{C}_\varepsilon \subseteq \mathbb{S}^{D-1}$ such that $|\mathcal{C}_\varepsilon| \leq (\frac{6}{\varepsilon})^D$ and

$$\forall \boldsymbol{w} \in \mathbb{S}^{D-1}\ \exists \tilde{\boldsymbol{w}} \in \mathcal{C}_\varepsilon \colon\ \|\boldsymbol{w} - \tilde{\boldsymbol{w}}\|_2 \leq \varepsilon. \tag{A.8}$$

For each $\tilde{\boldsymbol{w}} \in \mathcal{C}_\varepsilon$, we may take $\boldsymbol{w} = \boldsymbol{\Sigma}_t^{-1/2}(x)\tilde{\boldsymbol{w}}$ in (A.5), which gives

$$\mathbb{P}\left\{\tilde{\boldsymbol{w}}^\mathsf{T}\boldsymbol{\Sigma}_t^{-1/2}(x)(\boldsymbol{f}(x) - \boldsymbol{\mu}_t(x)) > \sqrt{\tilde{\alpha}}\right\} \leq e^{-\frac{\tilde{\alpha}}{2}}$$

since $\|\tilde{\boldsymbol{w}}\|_2 = 1$. After applying a union bound, we obtain

$$\mathbb{P}\{\exists \tilde{\boldsymbol{w}} \in \mathcal{C}_\varepsilon \colon \tilde{\boldsymbol{w}}^\mathsf{T}\boldsymbol{\Sigma}_t^{-1/2}(x)(\boldsymbol{f}(x) - \boldsymbol{\mu}_t(x)) > \sqrt{\tilde{\alpha}}\} \leq \left(\frac{6}{\varepsilon}\right)^D e^{-\frac{\tilde{\alpha}}{2}}. \tag{A.9}$$

It remains to make the connection between (A.9) and (A.7). To that end, let us introduce the notation $\|\boldsymbol{\mu}\|_{\boldsymbol{\Sigma}} = \sqrt{\boldsymbol{\mu}^\mathsf{T}\boldsymbol{\Sigma}\boldsymbol{\mu}}$ for a $D \times D$ symmetric positive definite matrix $\boldsymbol{\Sigma}$ and $\boldsymbol{\mu} \in \mathbb{R}^D$. Note that $\|\boldsymbol{\mu}\|_{\boldsymbol{\Sigma}} = \|\boldsymbol{\Sigma}^{1/2}\boldsymbol{\mu}\|_2$. Also recall that the ℓ^2

norm has the variational characterization $\|\boldsymbol{\mu}\|_2 = \max_{\boldsymbol{w}\in\mathbb{S}^{D-1}} \boldsymbol{w}^\mathsf{T}\boldsymbol{\mu}$. Then, under the complement of the event in (A.9), we have

$$\begin{aligned}
\|\boldsymbol{f}(x) - \boldsymbol{\mu}_t(x)\|_{\boldsymbol{\Sigma}_t^{-1}(x)} &= \left\|\boldsymbol{\Sigma}_t^{-1/2}(x)(\boldsymbol{f}(x) - \boldsymbol{\mu}_t(x))\right\|_2 \\
&= \max_{\boldsymbol{w}\in\mathbb{S}^{D-1}} \boldsymbol{w}^\mathsf{T}\boldsymbol{\Sigma}_t^{-1/2}(x)(\boldsymbol{f}(x) - \boldsymbol{\mu}_t(x)) \\
&= \max_{\boldsymbol{w}\in\mathbb{S}^{D-1}} \min_{\tilde{\boldsymbol{w}}\in\mathcal{C}_\varepsilon} \left[(\boldsymbol{w} - \tilde{\boldsymbol{w}})^\mathsf{T}\boldsymbol{\Sigma}_t^{-1/2}(x)(\boldsymbol{f}(x) - \boldsymbol{\mu}_t(x)) + \tilde{\boldsymbol{w}}^\mathsf{T}\boldsymbol{\Sigma}_t^{-1/2}(x)(\boldsymbol{f}(x) - \boldsymbol{\mu}_t(x))\right] \\
&\leq \sup_{\boldsymbol{w}\in\mathbb{S}^{D-1}} \min_{\tilde{\boldsymbol{w}}\in\mathcal{C}_\varepsilon} \left[\|\boldsymbol{w} - \tilde{\boldsymbol{w}}\|_2 \|\boldsymbol{\Sigma}_t^{-1/2}(x)(\boldsymbol{f}(x) - \boldsymbol{\mu}_t(x))\|_2 + \sqrt{\tilde{\alpha}}\right] \\
&= \sup_{\boldsymbol{w}\in\mathbb{S}^{D-1}} \min_{\tilde{\boldsymbol{w}}\in\mathcal{C}_\varepsilon} \left[\|\boldsymbol{w} - \tilde{\boldsymbol{w}}\|_2 \|\boldsymbol{f}(x) - \boldsymbol{\mu}_t(x)\|_{\boldsymbol{\Sigma}_t^{-1}(x)} + \sqrt{\tilde{\alpha}}\right] \\
&\leq \varepsilon\|\boldsymbol{f}(x) - \boldsymbol{\mu}_t(x)\|_{\boldsymbol{\Sigma}_t^{-1}(x)} + \sqrt{\tilde{\alpha}},
\end{aligned}$$

where the first inequality follows from Cauchy-Schwarz-Bunyakovski inequality and the definition of the event, the second inequality follows from the covering property in (A.8). Rearranging the terms gives

$$\|\boldsymbol{f}(x) - \boldsymbol{\mu}_t(x)\|_{\boldsymbol{\Sigma}_t^{-1}(x)} \leq \frac{\sqrt{\tilde{\alpha}}}{1-\varepsilon}.$$

Note that

$$\boldsymbol{f}(x) \in \mathcal{B}_t(x) \;\Leftrightarrow\; \|\boldsymbol{f}(x) - \boldsymbol{\mu}_t(x)\|^2_{\boldsymbol{\Sigma}_t^{-1}(x)} \leq \alpha_t.$$

Hence, for a given probability level $\delta_t' \in (0,1)$, in order to ensure that $\mathbb{P}\{\boldsymbol{f} \notin \mathcal{B}_t(x)\} \leq \delta_t'$, it suffices to choose $\varepsilon, \tilde{\alpha}$ such that

$$\left(\frac{6}{\varepsilon}\right)^D e^{-\frac{\tilde{\alpha}}{2}} = \delta_t', \quad \frac{\sqrt{\tilde{\alpha}}}{1-\varepsilon} \leq \sqrt{\alpha_t}.$$

Hence, we take

$$\tilde{\alpha} = 2\log\left(\frac{\left(\frac{6}{\varepsilon}\right)^D}{\delta_t'}\right) = 2D\log\left(\frac{6}{\epsilon}\right) + 2\log\left(\frac{1}{\delta_t'}\right).$$

Then, choosing $\varepsilon = \frac{1}{2}$ imposes the following condition on α_t:

$$\frac{\sqrt{\tilde{\alpha}}}{1-\varepsilon} = 2\sqrt{2(D\log(12) + \log(1/\delta_t'))} \leq \sqrt{\alpha_t} \;\Leftrightarrow\; 8D\log(12) + 4\log(1/\delta_t') \leq \alpha_t.$$

for every $\delta_t' \in (0,1)$. Let us set $\delta_t' := \frac{6\delta}{\pi^2 t^2 |\mathcal{X}|}$. Then, a union bound gives

$$\mathbb{P}\{\forall x \in \mathcal{X}, \forall t \in \mathbb{N}\colon \boldsymbol{f}(x) \in \mathcal{B}_t(x)\} \geq 1 - \sum_{x\in\mathcal{X}}\sum_{t\in\mathbb{N}} \frac{6\delta}{\pi^2 t^2 |\mathcal{X}|} = 1 - \sum_{x\in\mathcal{X}} \frac{\delta}{|\mathcal{X}|} \geq 1 - \delta,$$

where $\mathcal{B}_t(x) = \{\boldsymbol{\nu} : (\boldsymbol{\nu} - \boldsymbol{\mu}_t(x))^\intercal \boldsymbol{\Sigma}_t^{-1}(x)(\boldsymbol{\nu} - \boldsymbol{\mu}_t(x)) \leq \alpha_t\}$ where $\alpha_t = 8D \log(12) + 4 \log(\frac{\pi^2 t^2 |\mathcal{X}|}{6\delta})$.

This finishes the proof. $\square$