Lorenzo Bottarelli

Optimizing Information Gathering for Environmental Monitoring Applications

Ph.D. Thesis

XXXI Cycle (October 2015–September 2018)

Università degli Studi di Verona
Dipartimento di Informatica

Advisor:
Prof. Alessandro Farinelli

Università di Verona
Dipartimento di Informatica
Strada le Grazie 15, 37134 Verona
Italy

Abstract

The goal of environmental monitoring is to collect information from the environment and to generate an accurate model for a specific phenomena of interest.

We can distinguish environmental monitoring applications into two macro areas that have different strategies for acquiring data from the environment. On one hand the use of fixed sensors deployed in the environment allows a constant monitoring and a steady flow of information coming from a predetermined set of locations in space. On the other hand the use of mobile platforms allows to adaptively and rapidly choose the sensing locations based on needs. For some applications (e.g. water monitoring) this can significantly reduce costs associated with monitoring compared with classical analysis made by human operators.

However, both cases share a common problem to be solved. The data collection process must consider limited resources and the key problem is to choose where to perform observations (measurements) in order to most effectively acquire information from the environment and decrease the uncertainty about the analyzed phenomena. We can generalize this concept under the name of *information gathering*. In general, maximizing the information that we can obtain from the environment is an NP-hard problem. Hence, optimizing the selection of the sampling locations is crucial in this context.

For example, in case of mobile sensors the problem of reducing uncertainty about a physical process requires to compute sensing trajectories constrained by the limited resources available, such as, the battery lifetime of the platform or the computation power available on board. This problem is usually referred to as *Informative Path Planning (IPP)*. In the other case, observation with a network of fixed sensors requires to decide beforehand the specific locations where the sensors has to be deployed. Usually the process of selecting a limited set of informative locations is performed by solving a combinatorial optimization problem that model the information gathering process.

This thesis focuses on the above mentioned scenario. Specifically, we investigate diverse problems and propose innovative algorithms and heuristics related to the optimization of information gathering techniques for envi-

ronmental monitoring applications, both in case of deployment of mobile and fixed sensors. Moreover, we also investigate the possibility of using a quantum computation approach in the context of information gathering optimization.

Contents

Chapter 1

Introduction

1.1 Environmental Monitoring

The goal of environmental monitoring is to collect information and generate an accurate model for a specific environmental phenomena of interest. In general, environmental analysis aims at understanding, preserving and improving the quality of the environment. For example, when monitoring the quality of a body of water, operators might be interested in modeling how crucial parameters such as pH level, Dissolved Oxygen and temperature vary across time and space. Other examples can be represented by monitoring the quality and the amount of CO_2 in the air or the leakage of dangerous gas.

In such applications, common problems to solve are the minimization of the uncertainty of measurements or the *level set estimation*. In the level set estimation problem we need to identify in which portions of the environment the phenomena is above or below a given threshold level, e.g. where the pH level is above a level considered to be dangerous .

Environmental analysis usually requires the collection of large data sets sometimes in harsh conditions, hence the use of networks composed by fixed sensors or mobile platforms can be very beneficial. Typical examples include fixed stations deployed in the environment or mobile drones such as unmanned ground vehicles (UGVs), unmanned aerial vehicles (UAVs) or autonomous surface vessels (ASVs) such as the one in Figure 1.1.

On one hand, the use of a sensor network allows a constant monitoring of the environment with a steady flow of informations coming from the deployed sensors [121].

On the other hand the use of mobile platforms allows to adaptively and rapidly choose the sensing locations based on needs and to obtain data hence facilitating and reducing costs compared with classical analysis made by human operators. For an exhaustive overview on advancements and applications see [14, 67].

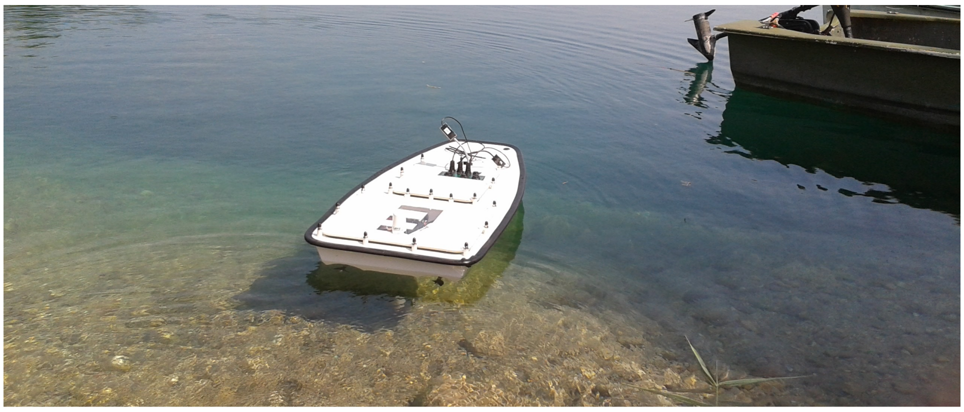


Figure 1.1: Mobile platform that we used: Platypus Lutra equipped with pH, dissolved oxygen, temperature and electrical conductivity sensors. The computation is on board and performed by an Arduino Due and a smartphone.

A successful monitoring operation must acquire a sufficient amount of data to build an accurate model of the environmental phenomena of interest. However, the data collection process must consider limited resources such as money, energy or time depending on the means used to take the measurements. Therefore, it is crucial to carefully select measurement locations to acquire as much information as possible while minimizing the resources required to collect the data.

In this thesis we focus on diverse problems related to the optimization of information gathering techniques for environmental monitoring applications both in case of deployment of mobile and fixed sensors.

1.2 Information Gathering

As mentioned above, environmental monitoring requires to gather information from the environment in order to build an accurate model of some physical phenomena. In many practical applications in order to monitor a spatial phenomena with sensor networks or mobile robots, the most crucial problem is to choose where to perform observations (measurements) in order to most effectively decrease the uncertainty about the phenomena. We can generalize this concept under the name of information gathering. In general, maximizing the information we can obtain from the environment is a hard problem from a computational standpoint, and the complexity depends on the specific setting of our application.

For example, in case of mobile sensors the problem of reducing uncertainty about a physical process requires to compute sensing trajectories constrained by the limited resources available such as the computation power or the battery lifetime of the platform. This problem is usually referred to as Informative Path Planning (IPP). IPP is used to design paths for mobile sensors in order to extract the maximum information while operating under a set of constraint and it has been shown to be NP-hard [87]. The key components of an informative path planner are the selection of locations to visit and the processing of the information gathered along the path.

On the other hand, observation with a network of fixed sensors requires to decide beforehand the specific locations where they have to be deployed. Usually performing observations or deploying sensors is costly, this implies that we have only a limited number of measurement locations available. The process of selecting a limited set of informative locations is again an NP-hard problem and usually performed by solving a combinatorial optimization problem.

To this aim, in this thesis we propose innovative algorithms and heuristics with the aim of optimizing crucial aspects of information gathering techniques both in the context of mobile and fixed sensor. Specifically, in the former case we propose two main algorithms that allow us to compute efficient sampling paths while improving computation time or path length (battery consumption) with respect to previous state-of-the-art algorithms. In case of fixed sensors, we propose three different techniques with the aim of computing effective deployment positions (sampling location) for a given set of available sensors (possible observations). In this context, the process of information gathering is modeled as an optimization (or a constrained optimization) problem and we developed techniques for the cases where the measurement domain is either continuous or discrete. We also investigate the possibility of using a quantum approach in the context of information gathering to exploit recent technological advances provided by quantum annealing machines such as the D-Wave.

1.3 Optimization

In computer science and mathematics, optimization is the selection of the best element from a set of available alternatives or in other words, an optimization problem is the problem of finding the best solution from all feasible solutions.

Optimization problems typically are composed by three fundamental elements, an objective function, a collection of variables and a set of constraints which are restrictions on the values that the variables can take. In the simplest case, an optimization problem consists of maximizing or minimizing a real function by choosing input values within an allowed set (domain) and satisfy the constraints.

We can divide optimization problems into two categories depending on whether the variables are continuous or discrete. In continuous optimization the variables used in the objective function are required to be continuous, that is, chosen from a set of real values. Because of this continuity assumption, continuous optimization allows the use of calculus techniques. On the other hand, in discrete optimization (also known as combinatorial optimization) the variables used in the objective function are restricted to be discrete, that is, to assume only a discrete set of values, such as binary or integers values.

Optimization can be seen as a tool with applications that span across many fields. For example, in the investment context, the portfolio optimization requires to allocate funds to stocks in such a way to minimize risk. In the production context, the machine allocation optimization requires to allocate the production of a product to different machines, with different capacities, startup cost and operating cost, to meet production target at minimum cost. In the logistic context we want to generate a transportation model that determines how many products to ship from each factory to each warehouse so as to minimize the shipping cost while meeting warehouse demands.

In the information gathering context discussed in this thesis, we need to optimize the sensing locations in order to generate a model of a physical environmental phenomena of interest, while satisfying the constraint given by the number of allowed samples.

1.4 Quantum computation

As previously mentioned in this thesis we also investigate the use of a quantum approach to solve optimization problem related to the information gathering context.

Quantum computation was first suggested in 1982 by Richard Feynman [70] as a way to overcome the problem of the exponential growth in resources required to simulate quantum phenomena on a classical computer. Since then, there have been numerous studies combining mathematics, physics and computer science involving both theoretical and practical advantages and difficulties.

Since many computationally interesting problems can be recast into an equivalent problem of finding the minimum of an objective function, in 2000 *Farhi et al.* [69] suggested the construction of a quantum algorithm using adiabatic evolution. Adiabatic Quantum Computation (AQC) is based on the Adiabatic Theorem stated in 1928 by Born and Fock [31]:

> *"A physical system remains in its instantaneous eigenstate if a given perturbation is acting on it slowly enough and if there is a gap between the eigenvalue and the rest of the Hamiltonian's spectrum."*

In other words, a quantum mechanical system remains in its ground state (lowest energy state) if a perturbation is acting on it slowly enough.

1.4.1 Quantum Annealing

Quantum Annealing (QA) [59, 71, 91, 155] is an optimization meta-heuristic that aims at optimizing an objective function. The process is realized in principle by the adiabatic evolution previously mentioned, where a quantum annealer minimizes a discrete variables objective function with a vast and complex landscape by physically exploiting quantum effects.

Quantum annealing was suggested as an improvement of the simulated annealing [96], an approach to optimization based on the observation that the cost function of an optimization problem can be viewed as the energy of a physical system, and that local minima can be escaped thanks to thermal hopping. In the quantum case the computation is driven by quantum fluctuation instead of thermal fluctuation. The advantage in the quantum case depends on the fact that unlike classical annealing, where the system climbs barriers, in quantum annealing fluctuation can help “tunneling” through these barriers and escape points of local minima (sketched in Figure 1.2) i.e., it uses some quantum effects that allows the tunneling through narrow barriers separating local minima, rather than climbing over them as done classically by using thermal fluctuations.

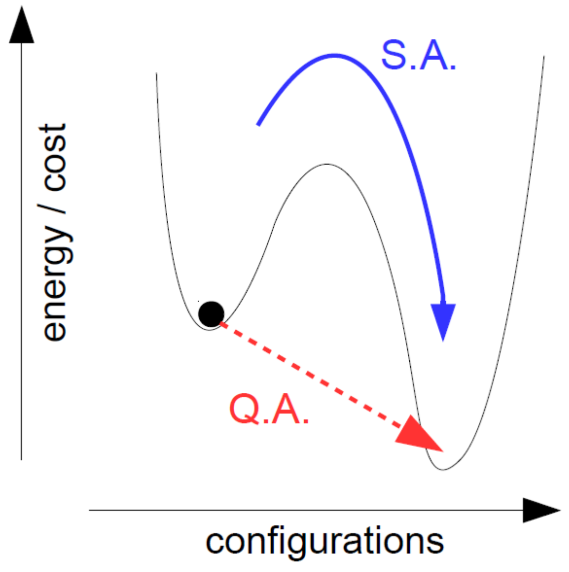


Figure 1.2: Graphical representation of the different behavior between quantum and simulated annealing.

However, there are similarities between quantum and simulated annealing. In both techniques it is important to control the relevant parameters and change them slowly to tune the strengths of quantum or thermal fluctuations.

Apart from theoretical demonstrations [71, 91] there are experimental evidence [39, 60, 76, 90] that quantum annealing could be advantageous to solve optimization problems compared to its classical analogue, demonstrating the power of quantum tunneling for reaching a better solution. A fundamental contribution in this direction is due to D-Wave Systems Inc., which has commercialized some analog quantum devices designed to use quantum annealing to solve quadratic optimization problems.

1.4.2 The D-Wave machine

D-Wave System Inc[1] is a quantum computing company based in Canada. They propose the D-Wave, a quantum annealer that uses quantum mechanical effects to efficiently sample low-energy configurations of particular cost functions on binary variables.

Specifically, a D-Wave minimizes an objective function of a specific class of problem, the Ising Model. Such energy minimization problems is NP-hard [11] and it is equivalent, through an arithmetic transformation, to a Quadratic Unconstrained Binary Optimization (QUBO) problem [98]. In QUBO, we aim to find an assignment for binary variables so as to minimize a quadratic objective function.

Such QUBO formulation can be embedded on the D-Wave architecture and the quantum annealing process will then return a set of solutions sampled from the energy function. Although, as any quantum annealing system the D-Wave machine is not guaranteed to return optimal solutions, it has been shown to outperform a range of classical algorithms on certain problems designed to match its hardware structure [60, 94]. Successful application [28, 29, 141, 164] opened a new era in quantum computing.

By using this protocol many interesting problems in artificial intelligence have been addressed (see Section 2.3). In this thesis we address two problems related to environmental monitoring applications by formulating Quadratic Unconstrained Binary Optimization models (see Section 3.5).

1.5 Contributions

As previously mentioned, the aim of this thesis is to propose techniques to optimize the information gathering process in environmental monitoring applications. Specifically, the main contributions of this thesis are the following:

1. In the context of mobile sensors we focus on the level set estimation problem. Specifically, we consider the case where a mobile platform

[1] `http://www.dwavesys.com/`

with low computational power can continuously acquire measurements with a negligible energy cost. This scenario imposes a change in the perspective with respect to other works in literature, since now efficiency is achieved by reducing the distance traveled by the mobile platform and the computation required by this path selection process, as opposed to minimizing the number of measurement required as in [79]. In this thesis we propose two active learning algorithms aimed at facing this issue, specifically:

- SBOLSE that casts informative path planning into an orienteering problem.
- PULSE that exploits a less accurate but computationally faster path selection procedure.

Evaluation of our algorithms, both on a real world and a synthetic dataset, shows that our approaches can compute informative paths that achieve a high quality classification, while significantly reducing the travel distance and the computation time.

2. In the context of fixed sensors, we focus on the variance minimization of a Gaussian process. A key issue in Gaussian Process modeling is to decide on the locations where measurements are going to be taken. A good set of observations will provide a better model. Current state of the art selects such a set so as to minimize the posterior variance of the Gaussian Process by exploiting submodularity. In this thesis we propose two contributions in this direction, specifically:

   - A gradient descent procedure to iteratively improve an initial set of observations so as to minimize the posterior variance directly.
   - A novel heuristic whose behavior is an approximation to that of gradient descent but with faster computation.

   The performance of these two techniques are analyzed under different conditions by varying the number of measurement points and the hyperparameters of underlying the Gaussian Process. Results show the applicability of both our techniques and the clear improvements that can be obtain in different settings. Moreover, these techniques has been tested by varying the dimensionality of the domain to show the generality of the approach and making it a viable option in other contexts besides environmental monitoring.

3. We investigate the use of quantum annealing by providing two novel Quadratic Unconstrained Binary Optimization (QUBO) models for problems related to environmental monitoring applications. Specifically:

- A QUBO model specifically designed to implement the problem of selecting a set of sampling locations. The aim is to minimize the posterior variance of a Gaussian process as proposed in the previous contributions, but this case in a discrete domain.
- Recent contributions [43,44] make use of clustering and biclustering techniques to analyze data of unsupervised mobile platforms in environmental monitoring operations. We focus on biclustering and propose a QUBO model designed to tackle this problem.

For both these models we provide mathematical proofs of the correctness. Moreover, for what concerns the biclustering model, we also provide an empirical evaluation performed on a D-Wave machine, discussing its applicability and embedding properties.

1.6 Organization of the thesis

This thesis is divided in four parts. The first part discusses related works in literature and the background knowledge required to define and analyze the problems and algorithms proposed in this thesis. The three remaining parts present the above mentioned contributions of the manuscript.

More in detail, the rest of the document is organized as follows: Chapter 2 and 3 present the related works in literature and the background knowledge required for this thesis. Chapter 4 and 5 discuss the SBOLSE and PULSE algorithms respectively and in Chapter 6 we present a comprehensive empirical evaluation of these techniques. In Chapter 7 and 8 we present the gradient descent and heuristic algorithms for the Gaussian process posterior variance minimization. Chapter 9 and 10 propose the QUBO models for the variance minimization and biclustering problem respectively. Finally, in Chapter 11 we draws conclusions and discuss future research directions.

Figure 1.3 provides a taxonomy of the contributions and organization of this thesis.

1.7 Publications

Most of the contents proposed in this thesis have been published in prestigious journal and conferences. Specifically, the contents of the Part II of the thesis (Chapters 4, 5 and 6) have been published in [32–34,37]. Regarding Part III, Chapter 7 have been published in [38] and regarding Part IV, Chapter 10 have been published in [35,36]. Specifically, the model proposed in Chapter 10 has been previously presented during the master's thesis, whereas the empirical evaluation using a D-Wave machine constitutes a new contribution of this thesis. The remaining contributions presented in Chapters 8 and 9 are currently in preparation.

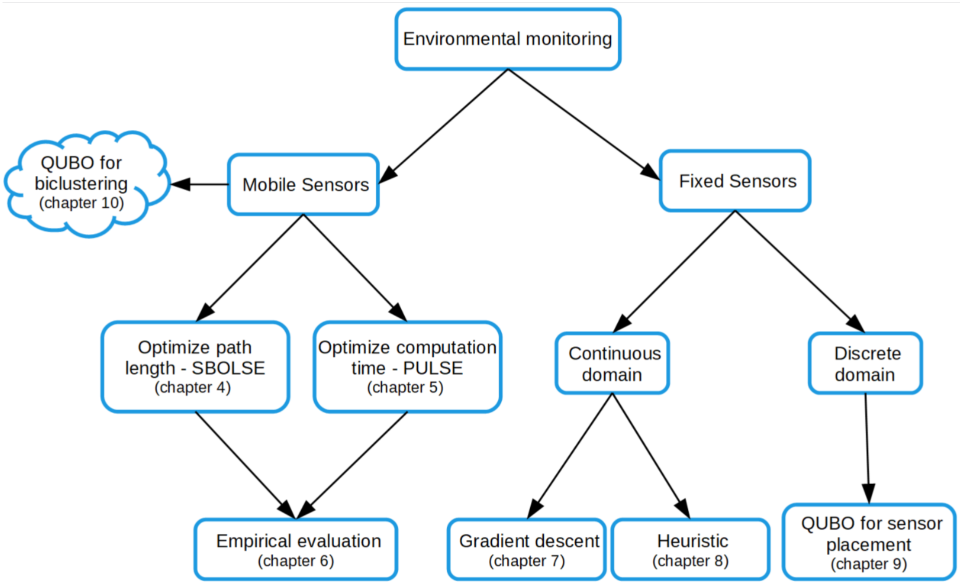

Environmental monitoring
Mobile Sensors
Fixed Sensors
QUBO for biclustering (chapter 10)
Optimize path length - SBOLSE (chapter 4)
Optimize computation time - PULSE (chapter 5)
Continuous domain
Discrete domain
Empirical evaluation (chapter 6)
Gradient descent (chapter 7)
Heuristic (chapter 8)
QUBO for sensor placement (chapter 9)

Figure 1.3: Taxonomy of the contributions and organization of the thesis.

Part I

Related Work & Background

Chapter 2

Related Work

As previously mentioned in the introduction, environmental monitoring encompasses the analysis and actions required to characterize and monitor the quality of the environment. This includes the collection of information from the environment and the generation of a model that represents a specific phenomena of interest [75, 110, 111].

Computational methods are often used to facilitate environmental monitoring, for example Cheng et al. [46] propose an expert system for the analysis of the water quality in a city. Another example is the monitoring of a body of water (e.g., lakes, rivers, coastal areas and so forth). In this case the analysis focuses on the generation of a model that describes how crucial parameters such as the presence of harmful algal blooms [126] or the dissolved oxygen (DO) vary across the environment.

In this chapter we position the contributions proposed in this thesis with respect to the existing literature in the areas of informative path planning for mobile sensors (Section 2.1) and optimization of sampling locations for fixed sensors (Section 2.2). Moreover, in the context of quantum computation for optimization problems we propose a review on quadratic unconstrained binary models (Section 2.3).

2.1 Mobile sensors

The goal of the information gathering process is to obtain data from the environment and to generate an accurate model for the phenomena of interest. In many applications the information gathering process requires to obtain measurement of the phenomena in harsh or dangerous conditions (e.g., environmental monitoring applications of water in a lake or search and rescue operations in disaster response). At the same time, a successful monitoring operation must acquire large datasets to build an accurate model of the environmental phenomena of interest.

To solve this problems, in recent years the interest towards robotic mobile sensors such as Unmanned Ground Vehicles (UGVs), Unmanned Aerial Vehicles (UAVs) or Autonomous Surface Vessels (ASVs) for information gathering applications has been steadily increasing. Significant advances in mobility and sensing capabilities have enabled the effective use of robotic systems in many contexts that require rapid and accurate information collection such as search and rescue [109, 127, 137, 181], source seeking [8], autonomous mapping [4, 5, 42, 100, 159], and environmental monitoring [41, 49, 168]. As a result, the use of autonomous vehicles is becoming an essential tool in a wide variety of applications.

Several examples specific for environmental monitoring includes attempts to use robots for underground gas leakage tracing [154], plume tracing [130], missions which involves mapping the magnitude of measured data (e.g. an RF signal) in a specific region [48] and microbe detection [165], For an exhaustive overview on advancements and applications of mobile sensors for environmental monitoring see [14, 67].

2.1.1 Informative path planning

In general, when deploying any kind of unmanned vehicles the data collection process must consider limited resources such as time or energy that constrain the operation range of the platforms. For example, one of the main constraints in path generation when operating a mobile system is the limited power source.

In some studies proposed in literature, it has been shown that exploiting wind-energy is one of the most effective ways of decreasing UAV energy consumption [3, 112, 180], however this is not an option available to every type of mobile platforms. Consequently, it is necessary to study and develop techniques specifically designed to minimize energy consumption by reducing the length of the path performed by mobile sensors as we propose in this thesis.

Specifically, to generate an accurate model of the environmental phenomena of interest in this case [89], it is important to select an informative path for the mobile agents to acquire as much information as possible while reducing the total traveled distance and hence the time and energy required to perform the analysis.

As a further issue, autonomous mobile systems are usually equipped with low computational capacity on board such as the one showed in Figure 1.1. As a consequence, when the path selection procedure is performed on-board during the monitoring operation, it is crucial to reduce as much as possible the computational complexity of the algorithms.

The problem that combines these aspects and requires to efficiently compute informative paths is generally known in literature as *Informative*

Path Planning (IPP). Informative path planning is used to design paths for robotic mobile sensors in order to extract the maximum possible information about a quantity of interest while operating under the set of constraints previously mentioned. The informative path planning problem is complex because it involves a combination of perception, estimation and inference, and the control of the mobile sensors. The key challenges of informative path planning are the strong coupling in multiple layers of decisions: the selection of locations to visit, and the processing of the gathered information along the paths.

In general, when using mobile robotic systems, different path selection strategies could be identified [161]. Traditional nonadaptive (offline) methods generate the path before any observations are made, hence the path is independent from the data that the sensors will read from the environment. In contrast, adaptive (online) methods plan the path based on the previously collected data during the operation [12, 148, 160]. As a consequence the path selection procedure is strictly dependent on the data that has been previously measured from the environment. In the first contributions of this thesis (discussed in Part II) we focus on this latter path selection scheme since usually informative path planning studies use an adaptive formulation [115, 162, 163].

Informative path planning problems have been studied for a long time, and most of the research has focused on minimizing the path length or the moving cost of the platforms. However, in many cases, the highest priority is to design paths that make the best use of the resources and maximize the amount of information acquired in a given mission. Especially considering that the quantity of information is not equally distributed at every point in space and hence, the planned path should prioritize areas where there is more "information" [15].

Several information measurement criteria have been used in literature to formulate informative path planning problems. Examples are the Fisher information [114], the Kullback-Leibler divergence [117], the mutual information [50] and, more related to the topics that we cover in this thesis, the Gaussian process uncertainty [25, 48]. Most of the studies about informative path planning have formulated the location where to acquire information as a constraint [108] or a reward [25] based on these information measurement criteria.

Our contributions proposed in this thesis fall within this context where, the environment locations have a specific information content and these values can be further used to drive the decision making process of the informative path planning algorithms.

The common aspect between these adaptive techniques is that they incrementally generate the model of the environmental phenomena of interest during the data collection phase and focus the information collection process on specific regions of the environment where the phenomena exhibits critical values.

An example where this kind of strategy is important is given by the pollution source localization problem [14]. In this case the aim is to find the location in space that represents the source of a given substance of interest (e.g. chemical pollutant at sea). The localization of toxic pollutant sources is an important as well as a challenging issue in many environment related areas. In order to reduce the impact of pollutants on the environment, it is imperative to localize pollutant sources so as to devise an effective control strategy.

2.1.2 Level set estimation

Another context that has drawn increasing attention due to its scientific and commercial interest, is to determine not the single source but instead the regions where the phenomena of interest exhibit dangerous conditions. For example, in a lake such regions could encompass the locations where the water's dissolved oxygen level is considered harmful for the environment. Another example could be the detection of contours of biological or chemical plumes [139]. From a general perspective, this can be seen as the problem of deciding if a quantity of interest is above or below a pre-specified threshold. This problem is typically referred to as the "level set estimation problem" in the literature [88].

Previous work on the level set estimation problem [57] focused on a network composed by a combination of static and mobile sensors. In the manuscript of Gotovos et al. [79] the proposed LSE algorithm uses Gaussian Processes (GPs) to identify sampling points that reduce uncertainty around a given threshold level of the modeled function. Even if the authors obtain a high quality classification with respect to threshold level (above or below) for the regions of the space using a low number of sampled locations, in their contribution the main algorithm does not explicitly take into account the path between the sampling locations. To partially consider this aspect, the authors propose a *batch* variant where a set of new sampling locations is selected in a batch such that it is possible to compute an efficient path between these points.

Hitz et al. [88] describe a method designed for ASVs equipped with a probe that allows an aquatic sensor to be lowered into the water. Their LSE-DP algorithm, built on the LSE algorithm [79], uses a dynamic programming approach with a receding horizon to plan a feasible sampling path for the probe within a predefined vertical transect plane.

In a more recent work Hitz et al. [87] introduce an evolutionary strategy to optimize a path in continuous space. Specifically, authors parametrize a path as a cardinal B-spline with n control points and propose a re-planning scheme to adapt the planned paths according to the measurements obtained from the environment.

The contributions of this thesis presented in Part II (Chapters 4, 5 and 6) are inserted in the aforementioned scenario where we aim at facing the problem of level set estimation by devising informative path planning algorithms specifically designed for low-cost, small mobile platforms that can perform sampling in various body of waters. As we will better explain in Part II, in the empirical evaluation we compare our techniques with the most similar ones in this context [79, 87].

However, a key element of novelty of our work with respect to the state of the art is that we exploit the fact that such platforms can acquire measurements while moving. Hence, our algorithms are specifically designed to optimize the information gathered along the path while the platform is moving instead of a set of sampling locations selected in the environment. We show that in this context our techniques are able to devise better solutions both in terms of path length and computation time.

2.2 Optimizing sampling locations

Many practical applications require to select among a set of feasible informative observations in order to monitor space, objects or relationships between entities. Consider for example problems where we want to perform observations in order to monitor traffic conditions in a road network, observe the relationship between input and outputs of an expensive computer simulation [166], or select between a set of feasible tests to administer to a patient in order to decide the most appropriate treatment [56]. In such problems, we can make observations by placing a set of sensors on the roads, trying some computer simulation or apply a medical testing procedure respectively. In practice, these observations are typically expensive in terms of sensor and deployment cost, power and time consumption, or cost for performing medical tests.

Hence, the common problem to solve is to decide which observations to perform in order to acquire the highest amount of information and at the same time be either cost effective or to satisfy given constraints, such as the maximum number of sensors available. This class of problems in literature is usually referred to as *sensing problems* [101], with a wide definition of the notion of "sensing". These problems have gained much attention in areas such as statistical experimental design, decision theory, operations research, probabilistic planning, and also sensing in environmental monitoring applications.

Network of small, wireless sensors are becoming increasingly popular for monitoring spatial phenomena [121], such as the temperature distribution in building [63], light prediction tasks [104], water supply contamination monitoring [106, 138] or air pollution [93].

When monitoring environmental spatial phenomena with a network of deployed sensors (or a set of handmade analysis), we need to decide which locations to observe in order to most effectively decrease the uncertainty about the phenomena. Sensor planning is particularly relevant for environmental monitoring tasks as there are limited resources and sparse measurements available. Hence, we need to perform the sampling process as efficiently as possible in order to maximize the information extracted from the environment while meeting constraints on resources or minimizing the associated costs.

2.2.1 Gaussian processes

As we will describe thorough the thesis, in many environmental analyses the spatial phenomena of interest is modeled using Gaussian Processes [149]. When tackling the analysis in a data-driven manner, the better the choice of locations we sample, the better the Gaussian process will approximate the true underlying functional relationship. Otherwise, we can consider the dual problem of obtaining a prespecified level of performance using the fewer measurements possible. More in general, a good set of observations will provide a better Gaussian process model.

Since only a limited number of sensors can be placed, it is important to deploy them in locations that are especially informative with respect to some objective function [80, 105, 108]. The choice of the locations to sample is usually a combinatorial NP-hard problem, thus in general it is not possible to find the optimal solution with an efficient procedure.

Classical approaches for tackling these sensing problems have mainly relied on myopic heuristics, i.e., approaches which only consider iteratively the next best observation to add (greedy heuristics), without planning for a set of future sensing opportunities. On the other hand non-myopic approaches try to find a better solution by planning ahead, but often are computationally very difficult to scale to larger problems. Typical sensor placement techniques greedily add measurements or sensors where uncertainty about the phenomena is the highest, i.e. the highest entropy location of the Gaussian process [55].

2.2.2 Variance reduction and submodularity

Recent research in the context of optimizing sampling locations aimed at selecting a set of measurements so as to optimize an important sensing quality function for spatial prediction that is represented by the reduction of predictive variance of the Gaussian Process [107] (also called Kriging variance in the geostatistics literature).

Another example, besides environmental monitoring applications, where minimizing the predictive variance of a Gaussian process is an important task is represented by the context of emulators for computer experiments [166]. In this context it is possible to use a Gaussian process to model an emulator for computationally expensive computer experiments. In order to provide a robust model and an accurate approximation of the relationship between simulation output and untried inputs at a reduced computational cost, a small predictive variance of the Gaussian process is an important objective.

The selection of measurement locations in order to minimize the posterior predictive variance unfortunately is NP-hard in general and the number of candidate solutions is very large. However, Das and Kempe [58] showed that, in many cases, the variance reduction at any particular location is submodular.

Submodularity is a property of a specific class of set functions. They encode an intuitive diminishing returns property that allows for a simple greedy forward-selection algorithm with an optimality bound guarantees [129]. This is widely exploited in sensing optimization literature [102,103,105,145,174]. The idea behind submodular techniques (that we better describe in Section 3.3) is that adding a new measurement to a small set helps more than adding it to a large set of measurements.

Although submodular objective functions allow for an efficient optimization, the solution is inherently discrete since we can select the sensing points only within a given set of feasible locations. In an environmental monitoring application where the feasible sensing locations are not confined to a given set of points, i.e. sensing locations can be arbitrarily adapted in a continuous domain of interest, the solution of such a technique can deviate considerably from the optimum. In other words, there is definitely room for improvement, which is the main goal of some contributions proposed in this thesis.

Specifically, the contributions presented in Part III (Chapters 7 and 8) are inserted in this context. By comparing our techniques with a submodular selection process, we show that when the sensing locations range in a continuous space of interest, continuous optimization techniques can obtain better results.

2.3 QUBO models

Several problems in Artificial Intelligence and Pattern Recognition are computationally challenging due to their inherent complexity and the exponential size of the solution space. To solve these problem numerous classic heuristic algorithm has been developed.

As previously mentioned in the introduction (Section 1.4), the emergence of quantum computation could provide a viable alternative to combat the complexity of hard optimization problems. A notable progress in this direction is represented by the recent development of the D-Wave quantum annealer, whose processor has been designed to the purpose of solving Quadratic Unconstrained Binary Optimization (QUBO) problems (more details in Section 3.5).

The general framework to solve a problem by using a D-Wave machine consists of two steps:

1. Map the problem of interest into a QUBO formulation, i.e. into a quadratic unconstrained cost function over binary variables.

2. Embed the QUBO formulation into the architecture of the D-Wave processor.

The embedding that represent the second step of this framework can be carried out with automated techniques [40]. As a consequence many works in literature investigate the possibility of using quantum annealing to address hard artificial intelligence and pattern recognition problems by proposing their QUBO formulation.

Examples include image recognition [132], bayesian network structure learning [135], fault detection and diagnosis [143], training a binary classifier [131] and portfolio optimization [122, 152, 176]. Moreover, in the context of mathematical logic, Bian et al. [22] propose a QUBO formulation to tackle the maxSAT problem, an optimization extension of the well known SAT (boolean satisfiability) problem [53].

NASA's Quantum Artificial Intelligence Laboratory (QuAIL) team[1] hosts one of the D-Wave machine and aims to investigate whether quantum computing can improve the ability to address difficult optimization and machine learning problems related to several fields that include NASA's aeronautics, Earth and space sciences, and space exploration missions. The focus of the QuAIL team is both theoretical and empirical investigations of quantum annealing. Biswas et al. [26] reviews NASA perspective on quantum computing of three potential application areas such as planning and scheduling [150, 164, 170, 171, 178], fault detection and diagnosis [143], and sampling/machine learning [2, 6, 17, 18]. These works are part of the emerging field of quantum machine learning [157] where the use of quantum computing technologies for sampling and machine learning applications has attracted increasing attention in recent years.

In this thesis we propose two contributions that are positioned in this context and that constitute a connection bridge between sampling/machine learning and environmental monitoring applications. These contributions

[1]https://ti.arc.nasa.gov/tech/dash/groups/physics/quail/

are described in Part IV of this thesis, where we investigate the use of quantum annealing by providing two QUBO models. Specifically, Chapter 9 presents a model that optimizes sampling locations by reducing the variance of a Gaussian process, addressing a problem similar to the one discussed in Part III of the thesis, but considering discrete domains for the locations of the sensors. Chapter 10 proposes a QUBO model for the biclustering problem.

Chapter 3

Background

In this chapter we formally describe the main concepts and problems later discussed in this thesis. We start by presenting the two most recurrent topics in this thesis, specifically, in Section 3.1 Gaussian processes and in Section 3.2 the level set estimation problem. Then in Sections 3.4, 3.3 and 3.5 we introduce gradient descent, submodular functions and Quadratic Unconstrained Binary Optimization problems. We conclude this chapter with a brief introduction to clustering in Section 3.6, orienteering in Section 3.7 and topological skeletonization in Section 3.8.

3.1 Gaussian processes

Gaussian processes are a widely used tool in machine learning [125,149] and provides a statistical distribution together with a way to model an unknown function f. In probability theory and statistics, a Gaussian process is a stochastic process (a collection of random variables), such that every finite collection of those random variables has a multivariate normal distribution. A Gaussian process defines a prior distribution over functions, which can be converted into a posterior distribution over functions once we have seen some data.

Although it might seem difficult to represent a distribution over a function, we only need to be able to define a distribution over the function's values at arbitrary and finite set of points $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$.

Gaussian Processes are a powerful technique for modeling and predicting data and they have some strong advantages:

1. Flexibility: can be used to model many different functions.
2. Non-parametric model: we do not have to worry about whether it is possible for the model to fit the data as would be the case if for example we were using a linear model on strongly non-linear data.
3. Based on probability theory, hence with a solid mathematical theory.

A Gaussian process is completely specified by its mean function and covariance function. The mean function $m(\mathbf{x})$ and the covariance function $k(\mathbf{x}, \mathbf{x}')$ of a real process $f(\mathbf{x})$ are defined as follows:

$$\begin{aligned} m(\mathbf{x}) &= \mathbb{E}[f(\mathbf{x})] \\ k(\mathbf{x}, \mathbf{x}') &= \mathbb{E}[(f(\mathbf{x}) - m(\mathbf{x}))(f(\mathbf{x}') - m(\mathbf{x}'))] \end{aligned}$$

and the Gaussian process can be written as

$$f(\mathbf{x}) \sim \mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')) \tag{3.1}$$

For notational simplicity the mean function can be zero without loss of generality, since the Gaussian process is flexible enough to model the mean arbitrarily well. In a Gaussian process the random variables represent the value of the function $f(\mathbf{x})$ at location $\mathbf{x}$.

Now, given a training set $\mathcal{D}$ of n observations, $\mathcal{D} = \{(\mathbf{x}_i, y_i) | i = 1, \ldots, n\}$, where $\mathbf{x}$ denotes an input vector of dimension D and y denotes a scalar output; the column input vectors for all n cases are aggregated in a $D \times n$ matrix X, and the outputs are collected in the vector $\mathbf{y}$, so we can write $\mathcal{D} = (X, \mathbf{y})$. We are interested in making inferences about the relationship between inputs and outputs, i.e. the conditional distribution of the outputs given the inputs.

Gaussian processes can be used under two different conditions, i.e. the prediction using noise-free and the prediction using noisy observations. In the case of noise-free observation if we ask the Gaussian process to predict $f(\mathbf{x})$ for a value of $\mathbf{x}$ that has already seen, it would return the answer $f(\mathbf{x})$ with no uncertainty. In other words, in this case the Gaussian process acts as an interpolator of the training data.

Throughout this thesis we focus instead on the case of noisy observation, hence, in the next section we briefly introduce Gaussian processes predictions under this second condition. Our presentation is based on [125, 149], which should be consulted for further details.

3.1.1 Predictions with noisy observations

Suppose we observe a training set $\mathcal{D} = \{(\mathbf{x}_i, y_i) | i = 1, \ldots, n\}$, with $y_i = f(\mathbf{x}_i) + \epsilon$ where $\epsilon \sim \mathcal{N}(0, \sigma_n^2)$, that is, observation with additive independent identically distributed Gaussian noise ϵ with variance σ_n^2.

Given a test set X_* of size n_*, we want to predict the function outputs $\mathbf{f}_*$. By definition of the Gaussian process (eq. 3.1), the joint distribution of the observed output values and the function values at the test locations is:

$$\begin{bmatrix} \mathbf{y} \\ \mathbf{f}_* \end{bmatrix} \sim \mathcal{N}\left(0, \begin{bmatrix} K(X, X) + \sigma_n^2 I & K(X, X_*) \\ K(X_*, X) & K(X_*, X_*) \end{bmatrix}\right) \tag{3.2}$$

where we are assuming the mean is zero, for notational simplicity. By the standard rules for conditioning Gaussians [125, 149], we obtain the key predictive equations for Gaussian process regression as follows:

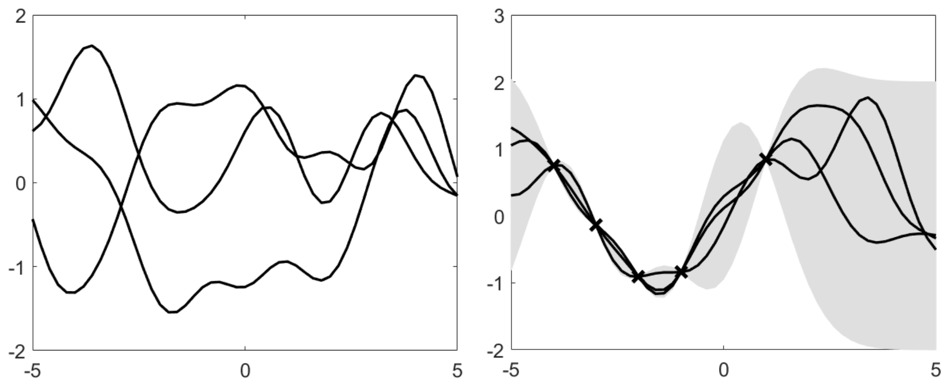
$$\mathbf{f}_*|X,\mathbf{y},X_* \sim \mathcal{N}\big(\bar{\mathbf{f}}_*, \operatorname{cov}(\mathbf{f}_*)\big), \text{ where} \tag{3.3}$$

$$\bar{\mathbf{f}}_* \triangleq \mathbb{E}[\mathbf{f}_*|X,\mathbf{y},X_*] = K(X,X)\big[K(X,X)+\sigma_n^2 I\big]^{-1}\mathbf{y} \tag{3.4}$$

$$\operatorname{cov}(\mathbf{f}_*) = K(X_*,X_*) - K(X_*,X)\big[K(X,X)+\sigma_n^2 I\big]^{-1}K(X,X_*) \tag{3.5}$$

If there are n training points and n_* test points, then $K(X,X_*)$ denotes the $n \times n_*$ matrix of the covariances evaluated at all pairs of training and test points with some kernel function $k(\cdot,\cdot)$. Similarly for the other entries $K(X,X)$ and $K(X_*,X_*)$.

This process is illustrated in a 1-dimensional example in Figure 3.1. On the left we show samples from the prior and on the right samples from the posterior given a set of observations.



(a) (b)

Figure 3.1: 3.1a functions sampled from a Gaussian process prior. 3.1b samples from a Gaussian process posterior with 5 noise-free observations. (Pictures generated using *probabilistic modeling toolkit PMTK3* - https://github.com/probml/pmtk3)

The notation of equations 3.4 and 3.5 can be simplified by using $\mathbf{K} = K(X,X)$ and $\mathbf{K}_* = K(X,X_*)$. If we have only one test point $\mathbf{x}_*$ we can denote the vector of covariances between the test point and the n training points with $\mathbf{k}_*$.

Using this simplified notation we can compute the mean and variance for a single test point $\mathbf{x}_*$ as follows:

$$\overline{f}_* \triangleq \mu(\mathbf{x}_*) = \mathbf{k}_*^T(\mathbf{K}+\sigma_n^2\mathbf{I})^{-1}\mathbf{y} \tag{3.6}$$

$$\mathbb{V}[f_*] \triangleq \sigma^2(\mathbf{x}_*) = k(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{k}_*^T(\mathbf{K} + \sigma_n^2\mathbf{I})^{-1}\mathbf{k}_* \tag{3.7}$$

Using the above equations we can compute the GP to update our knowledge about the unknown function f based on information acquired through observations and all these computations can be done in closed form, in $O(n^3)$ time.

Note that variance in equation 3.7 does not depend of the observed output values but only on the inputs of the training set. This is a property of Gaussian distribution and will play an important role throughout this thesis.

Gaussian processes use a measure of the similarity between points (the kernel function $k(\cdot, \cdot)$) to predict values of unseen point from training data. In the next section we briefly discuss the role of the kernel function and its hyperparameters.

3.1.2 Role of the kernel

The kernel function is at the core of a specific Gaussian Process and describes the similarity between two points. Specifically, a kernel function takes as input two points and gives as outputs the covariance between them. That determines how strongly correlated these two points are. The key idea is that if $\mathbf{x}_i$ and $\mathbf{x}_j$ are deemed by the kernel to be similar, then we expect the output of the function at those points to be similar too.

The predictive performance of Gaussian processes depends exclusively on the suitability of the chose kernel and parameters. There are lots of possible kernel functions to choose from [125, 149]. A common property is to have the covariance decrease as the distance between the points grows, so that the prediction is mostly based on the near locations. Alternatively, the kernel function could include a periodic part, such as a sine wave, to model a period component of the underlying function $\mathbf{f}$.

A famous and widely used kernel is the squared exponential, also known as Gaussian kernel or RBF kernel:

$$k(\mathbf{x}, \mathbf{x}') = \sigma_f^2 \exp\left(- \frac{(\mathbf{x} - \mathbf{x}')^T(\mathbf{x} - \mathbf{x}')}{2l^2} \right) \tag{3.8}$$

The kernel function will often have some parameters. For example, a length parameter that determines how quickly the covariance decreases with distance. For example in the squared exponential kernel here above, l controls the horizontal length scale over which the function varies, and σ_f^2 controls the vertical variation.

Suppose we choose this squared-exponential kernel (equation 3.8), Figure 3.2 illustrates the effect of changing the parameters l = horizontal length-scale, σ_f^2 = vertical length scale and σ_n^2 noise of the measurement on a 1 dimensional functions. Similarly Figure 3.3 illustrate the effect on the 2-dimensional case.

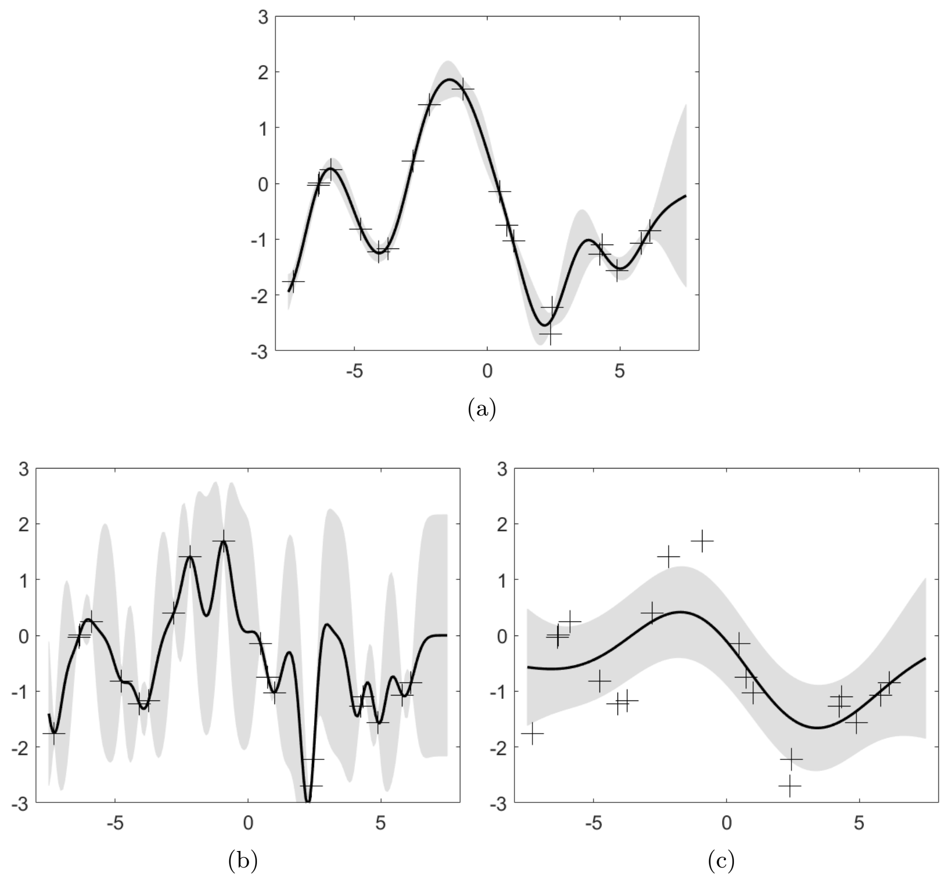
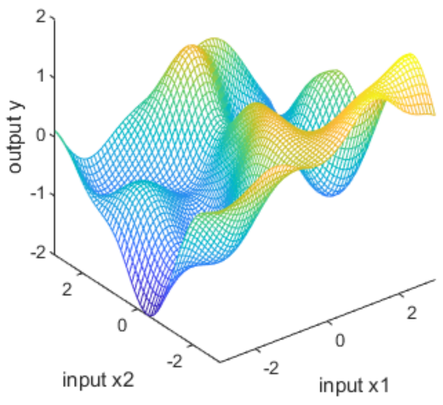


Figure 3.2: 1-dimensional Gaussian processes using the squared exponential kernel with different hyperparameters fit to 20 noisy observations. The specific hyperparameters (l, σ_f, σ_n) used are: (a) (1,1,0.1) - (b) (0.3,1.08,0.00005) - (c) (3,1.16,0.89). (Pictures generated using *probabilistic modeling toolkit PMTK3* - https://github.com/probml/pmtk3)

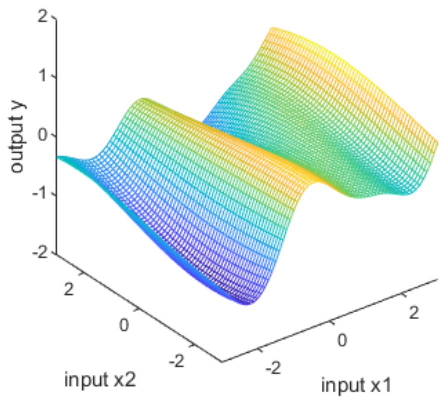
To estimate the kernel parameters a common practice is to use an empirical Bayes approach, which allows to use continuous optimization methods, in particular the maximization of the marginal likelihood [125, 149].

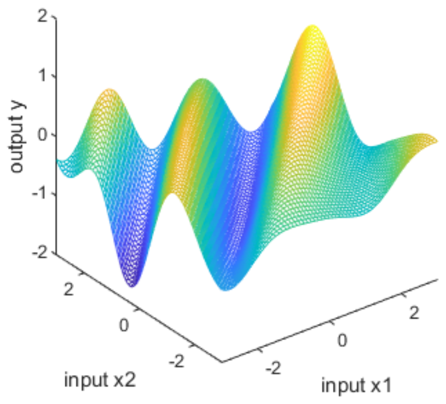
3.2 Level Set Estimation problem

In the level set estimation problem we have an unknown scalar field that can represents for example an environmental phenomena of interest. The objective is to partition each location of the scalar field into two groups that represent locations either above or below a given threshold level h. More formally, given a set of locations $\mathcal{X} \subseteq \mathbb{R}^d$ and a threshold value h, we want to model the unknown scalar field $f : \mathbb{R}^d \mapsto \mathbb{R}$ in order to classify all the



(a)





(b) (c)

Figure 3.3: 2-dimensional functions sampled from a Gaussian process using the squared exponential kernel with different hyperparameters. (Pictures generated using *probabilistic modeling toolkit PMTK3* - https://github.com/probml/pmtk3)

locations $x \in \mathcal{X}$ into either a superlevel set $H = \{x \mid f(x) > h\}$ or a sublevel set $L = \{x \mid f(x) \leq h\}$ (see Figure 3.4).

The problem is defined as the selection of the set of locations x_i where to perform (possibly noisy) measurements $y_i = f(x_i) + e_i$ while minimizing some criteria or satisfying some constraints depending on the context.

In an environmental monitoring application the unknown scalar field could represent the environmental phenomena of interest that has to be modeled for example the pH value of a body of water, and the threshold could represent a critical level for the safety of the environment. In this

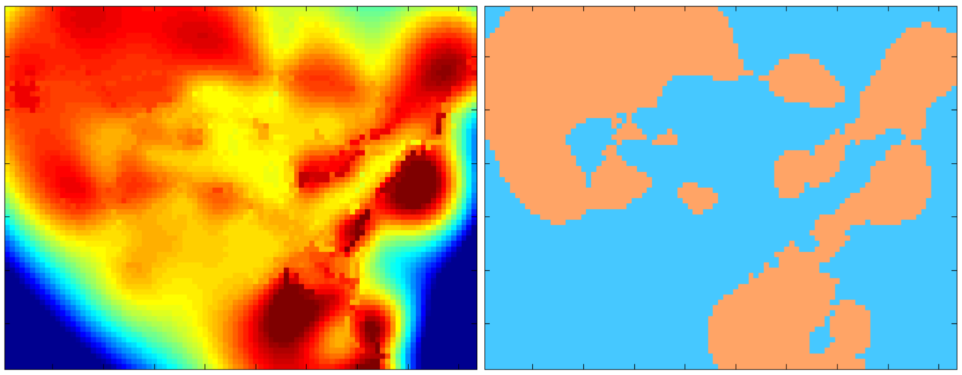


Figure 3.4: On the left side a real-world scalar field of pH level of waters extracted in the Persian Gulf near Doha (Qatar). On the right side the partition given a threshold $h = 7.4$ with orange above and blue below the threshold.

context, measurement locations should be selected to maximize the classification accuracy of the domain, while minimizing the resources required to obtain such a partition, such as total traveled distance required for a mobile sensor to analyze these locations, or the number of points to samples.

In the next section we summarize the method proposed by Gotovos et al. [79], which falls within this context and represents the starting point for some of our contributions presented in this thesis.

3.2.1 The LSE algorithm

The LSE algorithm presented by Gotovos et al. [79] is based on Gaussian Processes, a technique which offers a way to model unknown functions as we previously discussed in Section 3.1.

Specifically, in [79] authors propose to model the unknown scalar field f of an environmental phenomena of interest (the algal population in a lake in their case) by mean of a sequential selection of sampling locations. By doing this, the Gaussian process (and our knowledge about the scalar field) is sequentially updated with the new measurements acquired.

Authors in [79] consider a set of noisy measurements $Y_t = \{y_1, y_2, \cdots, y_t\}$ taken at locations $X_t = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_t\}$ at time t, and assume that $y_i = f(\mathbf{x}_i) + e_i$ where $e_i \sim \mathcal{N}(0, \sigma_n^2)$, i.e., measurement noise with zero mean. Given this set of measurements we can compute the mean and variance of the scalar field in every point in space using the equations 3.6 and 3.7 previously presented.

Given a region of interest in $\mathbb{R}^2$, they discretize it into a grid where each element represents a small portion of the surface. These elements compose the set of possible sampling locations (points) $\mathcal{X}$, and the goal is to classify each location $\mathbf{x}_i \in \mathcal{X}$ into two sets H or L with respect to a threshold level h. The LSE algorithm uses the inferred mean (eq. 3.6) and variance (eq. 3.7) of the Gaussian process to construct an interval:

$$Q_t(\mathbf{x}) = \left[\mu_{t-1}(\mathbf{x}) \pm \beta_t^{1/2}\sigma_{t-1}(\mathbf{x})\right] \tag{3.9}$$

for any $\mathbf{x} \in \mathcal{X}$. The parameter β_t represents a scaling factor for the interval. The procedure for tuning this parameter can be found in theorems 1 and 2 in [79].

Then, in order to classify every point $\mathbf{x}$ into either H or L, they define the following confidence interval using the intersection of all previous $Q_t(\mathbf{x})$ intervals for every point $\mathbf{x}$:

$$C_t(\mathbf{x}) = \bigcap_{i=1}^{t} Q_i(\mathbf{x}) \tag{3.10}$$

The classification of a point $\mathbf{x}$ depends on the position of its confidence interval with respect to the threshold level h. Specifically, for each location $\mathbf{x} \in \mathcal{X}$ if its confidence interval $C_t(\mathbf{x})$ lies entirely above h, then $f(\mathbf{x}) > h$ with high probability, and we can classify $\mathbf{x}$ into the superlevel set H. Similarly, when the entire confidence interval $C_t(\mathbf{x})$ lies below h then we can classify $\mathbf{x}$ into the sublevel set L. These conditions are relaxed with an accuracy parameter ϵ as shown in the following equations:

$$H_t = \{\mathbf{x} \mid min(C_t(\mathbf{x})) + \epsilon > h\} \tag{3.11}$$

$$L_t = \{\mathbf{x} \mid max(C_t(\mathbf{x})) - \epsilon \leq h\} \tag{3.12}$$

At time t, for every point with a confidence interval that crosses the threshold, we have to defer the decision until more information is available. The set of unclassified locations is then identified as:

$$U_t = \mathcal{X} \setminus (L_t \cup H_t) \tag{3.13}$$

In order to classify the points in U_t according to the equations (3.11) and (3.12), it is necessary to acquire more data by selecting new sampling locations $\mathbf{x}_i \in U_t$. To this end, the algorithm at each iteration uses the confidence interval for each unclassified point to derive the following ambiguity value:

$$a_t(\mathbf{x}) = min\{max(C_t(\mathbf{x})) - h, h - min(C_t(\mathbf{x}))\} \tag{3.14}$$

The LSE algorithm uses the measure of ambiguity for each unclassified point to guide the selection of the sampling locations. Specifically, the point $\mathbf{x}_t$ with the highest ambiguity value represents the location with the highest information content. As such, it becomes the next point to measure.

3.3 Submodular functions

As previously mentioned in Section 2.2.2, submodular functions have properties that can be exploited to find greedy solutions to optimization problems. The variance of a Gaussian process given a set of observations falls into this category of functions and greedy submodular techniques represent the baseline of comparison with some of our contributions presented in this thesis. Therefore, in what follows we better describe this specific class of set functions.

In mathematics we define as a set function, a function whose inputs is a set of elements. Particular classes of set functions turn out to be submodular. The property of submodularity formally translates as follows:

Definition 1. *A set function $F : 2^X \to \mathbb{R}$ is submodular if for all subsets $A, B \subseteq X$ it holds that*

$$F(A \cup B) + F(A \cap B) \leq F(A) + F(B) \tag{3.15}$$

A maybe more intuitive characterization of a submodular function has been given by Nemhauser et al. [129] as follows:

Definition 2. *A function F is submodular if and only if for all $A \subseteq B \subseteq X$ and $x \in X \setminus B$ it holds that:*

$$F(A \cup \{x\}) - F(A) \geq F(B \cup \{x\}) - F(B) \tag{3.16}$$

This second definition captures a concept known as diminishing return property. Informally we can say that if F is submodular, adding a new element x to a set increases the value of F more if we have fewer elements than if we have more.

This concept is of our interest as it directly apply to environmental monitoring applications. For example, in a environmental monitoring application, performing an additional measurement in a environment where we have few measurements helps more (gives more information) that adding a new measurement in a environment where we have already obtained many observations.

Specifically, the posterior variance of a Gaussian process belongs to this class of submodular functions. Das and Kempe [58] show that the variance reduction:

$$F_{\mathbf{x}}(A) = \sigma^2(\mathbf{x}) - \sigma^2(\mathbf{x}|A) \tag{3.17}$$

at any particular point $\mathbf{x}$, satisfies the diminishing returns property: adding a new observation reduces the variance in $\mathbf{x}$ more if we have made few observations so far, and less if we have already made many observations.

Since each of the variance reduction functions $F_{\mathbf{x}}$ is submodular, the average variance reduction

$$F(A) = \frac{1}{|X|} \sum_{\mathbf{x}} F_{\mathbf{x}}(A) \tag{3.18}$$

is also submodular.

3.4 Gradient descent

In the context of optimizing a set of sampling locations, one of the contributions proposed in this thesis (Chapter 7) is a gradient descent algorithm to minimize the posterior variance of a Gaussian process. Therefore, in this section we briefly introduce gradient descent techniques.

Gradient Descent (GD) is a very common and simple iterative optimization algorithm used to minimize an objective function. The basic idea behind gradient descent is that the algorithm starts from an initial point in the desired objective function and then takes a step in the direction of the negative of the gradient of the function itself in that point. The process iterates until it converges to a local minimum of the objective function. This simple idea is represented in Figure 3.5.

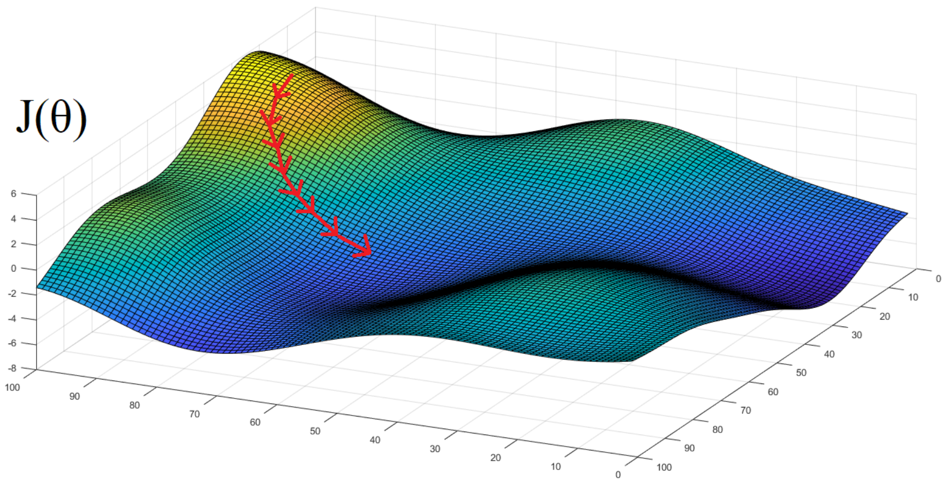


Figure 3.5: Abstract representation of gradient descent on a 2-dimensional objective function.

In more details, given a multi-variable function $J(\theta)$ differentiable in a neighborhood of a point θ, gradient descent is based on the observation that $J(\theta)$ decreases fastest in the direction of the negative gradient, that is $-\nabla J(\theta)$. Whenever the gradient of the function is non-zero we know that in the direction of the negative gradient we can minimize the objective function further, and in that sense improve upon the current solution.

The equation below describe the basic iteration of gradient descent:

$$\theta_{i+1} = \theta_i - \alpha \nabla J(\theta_i) \tag{3.19}$$

This formula basically tells that the next position where the algorithm goes (θ_{i+1}), is in the direction of the steepest descent from the current position θ_i modulated by a parameter α called learning rate. This learning rate α determines how fast or slow we will move towards the current basin of attraction of the objective function.

In order for gradient descent to reach the local minimum, we have to set this learning rate α to an appropriate value, which is neither too low nor too high (see figure 3.6). If we set the learning rate to a very small value, gradient descent will eventually reach the local minimum but it will maybe take too much time like you can see on the left side of the image. If alpha is too large (as represented on the right side of the picture), gradient descent can overshoot the minimum. It may fail to converge, or in some cases even diverge.

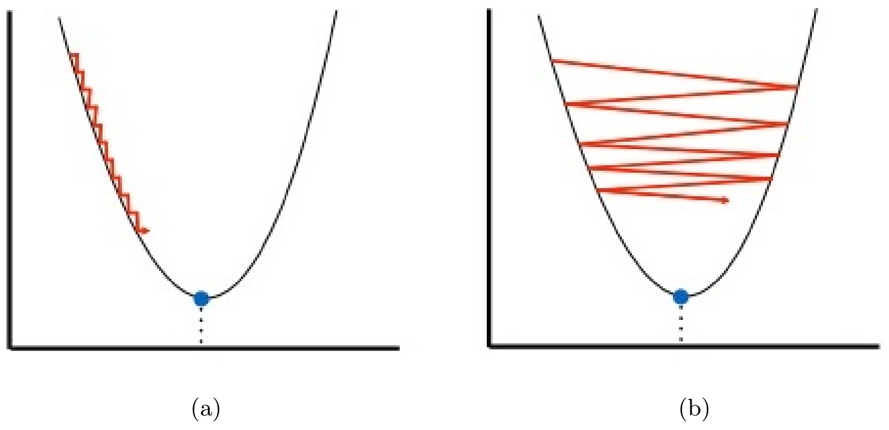

(a) (b)

Figure 3.6: Iterations of gradient descent using: 3.6a a learning parameter α too small and 3.6b a learning parameter α too big.

The value of the learning parameter α can be determined mostly by trial and error, there is no one-fits-all for parameter and it depends on the context of the objective function.

When gradient descent cannot decrease the function anymore or remains more or less on the same level (i.e. the improvements between an iteration and the other are less than a given threshold), we say it has converged. Note that the number of iterations that gradient descent has to perform in order to converge can sometimes vary significantly. Therefore the number of iterations is hard to estimate in advance. There are also some algorithms that can tell automatically if the gradient descent iteration has converged but with the requirement to define a threshold for the convergence beforehand which is also pretty hard to estimate.

Gradient descent is a widely used algorithm in machine learning. In this context the objective function is used to monitor the error in predictions of a machine learning model. Minimizing this error function basically means obtaining the lowest error value or in other words increasing the accuracy of the model. In this context we can observe variants of gradient descent in literature that mainly differ in the amount of data they use [153].

As previously mentioned, as far as we are concerned in this thesis we will make use of a gradient descent technique in order to minimize the variance of a Gaussian process in the context of sensor placement (sensing location) for environmental monitoring applications.

3.5 Quadratic Unconstrained Binary Optimization (QUBO)

As described in Section 2.3, the emergence of quantum computation could provide a viable alternative to tackle hard optimization problems. A notable progress in this direction is represented by the development of the D-Wave quantum annealer, designed to solve Quadratic Unconstrained Binary Optimization (QUBO) problem.

In this thesis we investigate the possibility of using a quantum computation approach in the context of environmental monitoring applications by providing two QUBO models (Chapters 9 and 10). Therefore, in what follows we provide the formal description regarding QUBOs.

The goal of a Quadratic Unconstrained Binary Optimization problem is to find the assignment of a set of binary variables $z_1 ... z_n$ so as to minimize a given objective function

$$O(z_1, ..., z_n) = \sum_{i=1}^{n} a_i z_i + \sum_{1 \leq i < j \leq n} b_{i,j} z_i z_j \tag{3.20}$$

Each instance of a QUBO problem can be conveniently represented by using a weighted graph where:

- There is a node for every binary variable z_i
- A linear coefficient a_i encodes the value associated to the node z_i
- A quadratic coefficient $b_{i,j}$ encodes the value associated to the edge between nodes z_i and z_j

In this graphical representation the QUBO objective function (3.20) corresponds to the summation of the values in the graph, namely the sum of linear terms will be the sum of the node values and the sum of the quadratic terms will be the sum of the edge values:

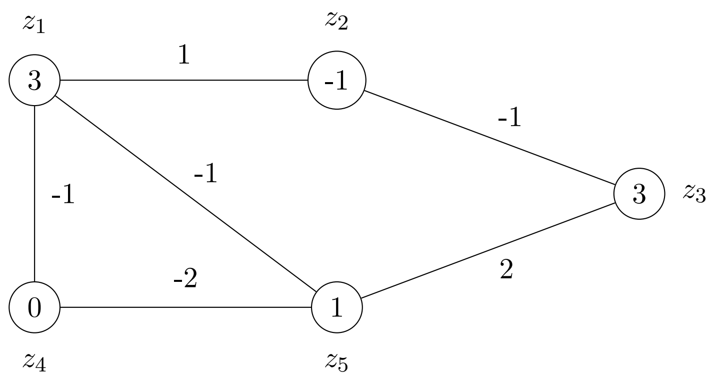
$$O(z_1, ..., z_n) = \underbrace{\sum_{i=1}^{n} a_i z_i}_{\text{node values}} + \underbrace{\sum_{1 \leq i < j \leq n} b_{i,j} z_i z_j}_{\text{edge values}} \tag{3.21}$$

By using this convenient graphical representation of a QUBO instance, we can also imagine the problem of minimizing the objective function as follows:

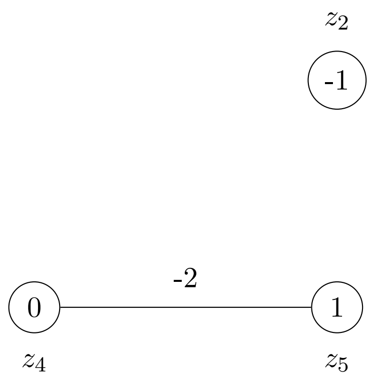
- $z_i = 1$ is equivalent to keeping the node z_i in the graph.
- $z_i = 0$ is equivalent to removing the node z_i from the graph.
- Minimizing the function is equivalent to decide which nodes to remove from the graph, where removing a node implies the removal of all edges that are incident to that node.

Example

The function $O(z_1, z_2, z_3, z_4, z_5) = 3z_1 - z_2 + 3z_3 + z_5 + z_1z_2 - z_2z_3 - z_1z_4 - z_1z_5 - 2z_4z_5 + 2z_3z_5$ can be represented as:



and minimizes for $O(0, 1, 0, 1, 1) = -2$ represented as:



3.6 Clustering

Clustering (or cluster analysis) refers to the division of data into groups (clusters) of similar objects, based on the concept of similarity: object in the same group should be similar, whereas objects belongings to different groups should be dissimilar.

Clustering can be viewed as a data modeling technique that provides for concise summaries of the data. In clustering we can represent a quantity of data with a relatively small number of clusters, that is, some details are disregarded in exchange for data simplification. Clustering is therefore related to many disciplines and plays an important role in a broad range of applications and usually deals with large datasets and data with many attributes.

Cluster analysis is in many cases an essential data analysis task and it can be used as an independent data mining process to disclose intrinsic characteristics of data, or as a preprocessing step with the clustering results used further in other task. Clustering has been studied extensively in various research fields, including machine learning, pattern recognition, engineering, social, economic, and biomedical data analysis. As a consequence cluster analysis can be performed by numerous algorithms that differ significantly in what constitutes a cluster and how to efficiently find them [1, 19, 183].

Popular notions of clusters include groups with small distances between cluster members (as shown in Figure 3.7), intervals or particular statistical distributions. The appropriate clustering algorithm depend on the individual data set and intended use of the results. Cluster analysis as such is not a straightforward task, but instead an iterative process of knowledge discovery that involves trial and failure. It is often necessary to modify data preprocessing and model parameters until the result achieves the desired properties.

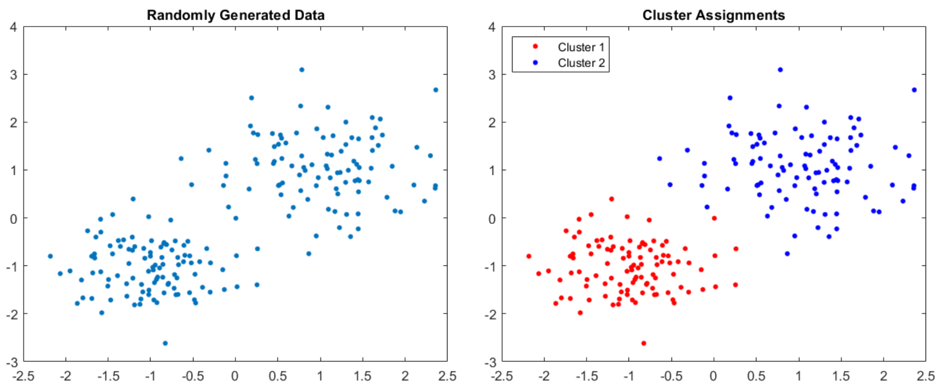


Figure 3.7: Simple example of clustering (performed with k-means algorithm)

Although the wide variety of clustering techniques that is possible to find in literature [99, 182], in our contributions proposed in this thesis we make use of three specific types that we introduce here below. Specifically, soft K-means, exemplar based clustering through an affinity propagation algorithm and biclustering.

3.6.1 K-Means and soft K-means

In our contribution presented in Chapter 8, we devise an heuristic approach for the problem of minimizing the posterior variance of a Gaussian process and the core of our algorithm is inspired by a soft K-means technique. Hence in what follows we describe the basic knowledge regarding this topic.

K-means is one of the most popular techniques to cluster a given data set $\mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\} \subset \mathbb{R}^d$ into k subsets (clusters) $C_1, \ldots, C_k$ such that:

$$C_1 \cup C_2 \cup \cdots C_k = \mathcal{X} \tag{3.22}$$

and for $i \neq j$

$$C_i \cap C_l = \emptyset \tag{3.23}$$

K-means clustering aims to partition the n data points so as to minimize the within-cluster sum of squares, formally:

$$\underset{\boldsymbol{\mu}_1,\ldots,\boldsymbol{\mu}_K}{\arg\min} \sum_{i=1}^{K} \sum_{\mathbf{x}_j \in C_i} \left|\left|\mathbf{x}_j - \boldsymbol{\mu}_i\right|\right|^2 \tag{3.24}$$

where $\boldsymbol{\mu}_i$ is the mean point (centroid) of C_i.

The objective function in equation 3.24 has numerous local minima and since the problem is NP-hard there is no algorithm known today that is guaranteed to find the optimal solution. However, the underlying ideas of k-means clustering are intuitive, several heuristic algorithms have been developed [82, 116, 119] and most theoretical properties are well established [74, 118].

We can observe that equation 3.24 can also be expressed in terms of binary *indicator variables* $z_{i,j}$ which indicates whether or not a point $\mathbf{x}_j$ belongs to cluster C_i. Specifically, with:

$$z_{i,j} = \begin{cases} 1, & \text{if } \mathbf{x}_j \in C_i \\ 0, & \text{otherwise} \end{cases} \tag{3.25}$$

the objective function in equation 3.24 becomes equivalent to:

$$\underset{\boldsymbol{\mu}_1,\ldots,\boldsymbol{\mu}_K}{\arg\min} \sum_{i=1}^{K} \sum_{j=1}^{n} z_{i,j} \left|\left|\mathbf{x}_j - \boldsymbol{\mu}_i\right|\right|^2 \tag{3.26}$$

The above formulation represents the standard k-means clustering where each point belong to one and only one cluster. We can however relax the assumption in equation 3.23 and define a formulation of a *soft* K-means [13], also referred to as *fuzzy k-means clustering* [20, 68].

There is a fundamental property with regard to this idea. Soft K-means clustering allows every data point to belong to several clusters simultaneously but this is not meant in the sense that clusters may overlap. Rather, soft K-means clustering determines to which degree each data point belongs to each cluster.

This idea can be simply implemented in the objective function represented by equation 3.26 by allowing the indicator variables $z_{i,j}$ to vary continuously, i.e. $z_{i,j} \in [0, 1]$. Since it is reasonable to decide cluster membership based on distances to cluster centroids a popular choice to achieve this transformation is to consider softmax functions [128]:

$$z_{i,j} = \frac{e^{-\beta \left|\left|\mathbf{x}_j - \boldsymbol{\mu}_i\right|\right|^2}}{\sum\limits_{l=1}^{K} e^{-\beta \left|\left|\mathbf{x}_j - \boldsymbol{\mu}_l\right|\right|^2}} \tag{3.27}$$

where β is called *stiffness parameter*. The indicator values $z_{i,j}$ thus defined can now be interpreted as probabilities $p(\boldsymbol{\mu}_i|\mathbf{x}_j)$.

3.6.2 Exemplar Based Clustering (EBC)

In this Section we present the background knowledge regarding exemplar based clustering through an affinity propagation algorithm. This technique has been used as a preprocessing operation in our contribution presented in Chapter 4.

A common approach in clustering is to use the data we hold to learn a set of centers such that the sum of squared errors between data points and their nearest centers is small (see Figure 3.7). This is the typical example encountered when approaching clustering with a technique such as k-means as we described in the previous section.

In contrast to this, when the centers are selected from actual data points, they are called “exemplars”. A popular technique in this context is the k-centers algorithm [133]. It begins with an initial set of randomly selected exemplars and iteratively refines this set so as to decrease the sum of squared errors. K-centers clustering (similarly to k-means) is quite sensitive to the initial selection of exemplars. The number of exemplars has to be determined beforehand and it has to be executed multiple times with different initializations to find a good solution. This algorithm works well only when two assumptions are correct, specifically the number of clusters (exemplars) is correct and when chances are good that at least one random initialization is close to a good solution.

Frey and Dueck [73] take a different approach and introduce a method that simultaneously considers all data points as potential exemplars. They proposed a technique where each data point is considered a node in a network, and devised an algorithm that recursively transmits messages along edges of the network until a good set of exemplars and the corresponding clusters emerges (points which have chosen the same exemplar are in the same cluster). Messages are updated on the basis of simple formulas that search for minima of an appropriately chosen energy function. The method is called affinity propagation because the magnitude of each message reflects the current "affinity" that one data point has for choosing another data point as its exemplar.

Affinity propagation takes as input a set of real-valued similarities between data points, where the similarity $s(i, k)$ specifies how well the data point with index k is suited to be the exemplar for data point i. For example, if the goal is to minimize the squared error, each similarity can be set to the negative Euclidean distance. Specifically, for points x_i and x_k, $s(i, k) = -||x_i - x_k||^2$. Alternatively, when appropriate for the application of interest, these similarities may be set by hand. As previously mentioned, one of the main advantages of this algorithm is that it does not require to define an explicit number of clusters beforehand. A suitable number is automatically determined and is influenced by the self similarities $s(k, k)$, referred to as "preferences", indicating how likely each point is to become an exemplar.

The efficiency and accuracy of this algorithm have been shown in different applications [73]. In this thesis, we are interested in this affinity propagation clustering technique as a preprocessing step to reduce the computation required by one of our algorithm later described in Chapter 4.

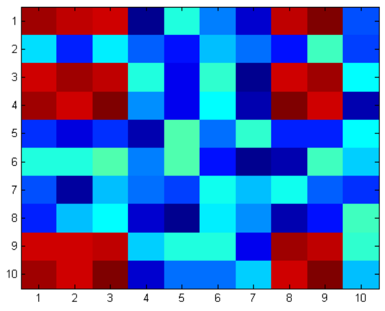
3.6.3 Biclustering

As previously mentioned in Section 1.5, one of the contribution presented in this thesis is a quadratic unconstrained binary optimization model for the biclustering problem.

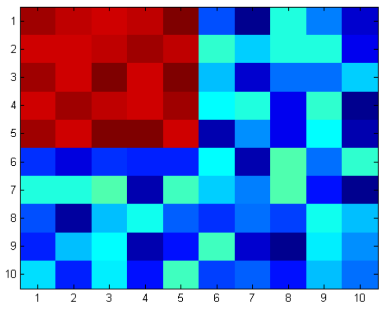
Biclustering, also known in other scenarios as sub-space clustering, is a term used to encompass a large set of data mining techniques generally aimed at "performing simultaneous row-column clustering" of a data matrix [120]. The biclustering problem appears in several different scenarios, such as document analysis [64], market segmentation [66], recommender systems [124], classification of medical data [140] and expression microarray data analysis [10, 72, 120, 134, 146].

More recently, biclustering has been employed for detecting, modeling and interpreting aquatic drone state in the context of water monitoring operations [44]. The use of this technique enables the analysis of large amounts of data collected by autonomous drones and enhance situation awareness which can be exploited by the platform to improve the decision making process.

The starting point of biclustering is a matrix whose rows and columns represent different aspects specific of the data analyzed. For example in the context of water monitoring [44] rows represent different properties of the drone (e.g. position, time, speed, physical phenomenas measured) and the columns represent samples over time. The analysis in this scenarios is to perform biclustering with the aim of discovering which properties show a coherent behavior over the time (for a graphical representation see Figure 3.8). In more detail, Castellini et al. [44] proposes the use of biclustering to identify interesting situations for the drones that can not be directly measured with specific sensors, such as whether the drone is moving upstream or downstream. The use of biclustering in this context is key to minimize the features that are considered to identify such situations. In fact, as mentioned before in contrast to clustering approaches, biclustering techniques can group coherent data over a subset of the features that define the data points. This is important in this context because it allows to retrieve situations more accurately and (more important) it can provide to a human operator a clear indication about which features are more relevant to determine the situation of interest. Results show the utility of such approach for environmental monitoring applications.



(a)



(b)

Figure 3.8: Example of a constant bicluster, before 3.8a and after permutations 3.8b.

Different biclustering techniques have been proposed in the past – e.g. [9, 24, 47, 62, 173], each one characterized by different features, such as computational complexity, effectiveness, interpretability and optimization criterion. For a general review we recommend [72, 84, 85, 120, 146].

Some of these approaches adapts a specific clustering technique to the biclustering problem, for example by repeatedly performing rows and columns clustering. However, the majority of recent works aim at proposing novel models for biclustering, where rows and columns are analyzed simultaneously (as opposed to clustering rows and columns separately) [173]. This approach has several advantages for what concerns the performance of the biclustering process that is significantly more accurate.

However, such accuracy comes at a price as such models typically involve a large amount of variables and relationships. In the context of biclustering data analysis, the typical candidate data are represented by a matrix with thousands of rows and columns. Moreover, the underlying optimization task required by the model is NP-hard leading to severe restrictions on the practical applicability of those approaches. In order to combat such complexity, recent works typically relax the model or use heuristic, greedy approaches, hence giving up optimality of the solution.

In this thesis we follow a standard technique where the problem of biclustering is formulated as a sequential search for the most coherent bicluster. This is a widely employed technique in the literature [16,47,61], and consists of the extraction of biclusters one by one from the data-matrix. Clearly, it is crucial how to "mask" the obtained bicluster before looking for the next one. In literature we can find different heuristics addressing this problem: for example, one way to address this problem is to replace the obtained bicluster with background noise in the original data matrix [47], so that the next bicluster can be looked for.

3.7 Orienteering

In our contribution regarding the computation of informative trajectories for mobile sensors, and specifically in our SBOLSE algorithm presented in Chapter 4, we make use of an orienteering formulation.

The Orienteering Problem (OP) originates from the sport game of orienteering. In the orienteering game, the start and end locations are specified along with a set of other checkpoints which have an associated score. The players aim to visit checkpoints in order to maximize the total score and reaching the end point within a given time frame. This problem can be used to model several different contexts. For example, consider the problem in which a traveling salesperson has a set of cities which could be visited. Assuming that the salesperson knows the expected number of sales in each city, the goal is to plan a route so as to maximize the total number of sales while keeping the total length of such route within a given budget (i.e. the maximum distance that can be traveled in one day).

More formally, the orienteering problem can be formulated in the following way: given a set of N locations, each with a score $S_i \geq 0$, a starting location with index $i = 1$, an ending location with index $i = N$ and the travel time t_{ij} for all couples of locations i and j (with $i \neq j$), the goal is to plan a route, limited by a given budget T_{max}, that visits a subset of these locations in order to maximize the total collected score.

An orienteering instance can easily be described using a weighted undirected complete graph $G = (V, E)$ where $V = \{v_1, \ldots, v_N\}$ is the set of locations (nodes) and E is the set of edges between every pair of nodes. In this formulation the nonnegative score S_i of location i is associated with a vertex $v_i \in V$, and the travel time t_{ij} between location i and j is associated with each edge $e_{ij} \in E$. See an example in figure 3.9.

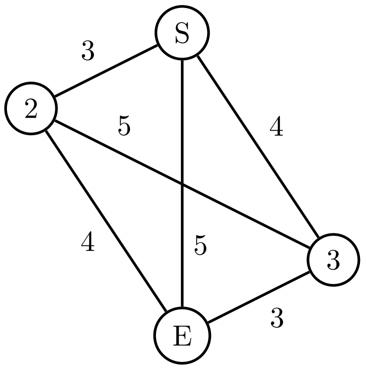


Figure 3.9: Example of the graphical representation of an orienteering instance. S represents the starting node and E represents the end node.

The orienteering problem consists of determining a Hamiltonian path over a subset of V, including the start node (v_1) and end node (v_N), having a total length that does not exceeds the bound T_{max}, in order to maximize the collected score. See example in figure 3.10.

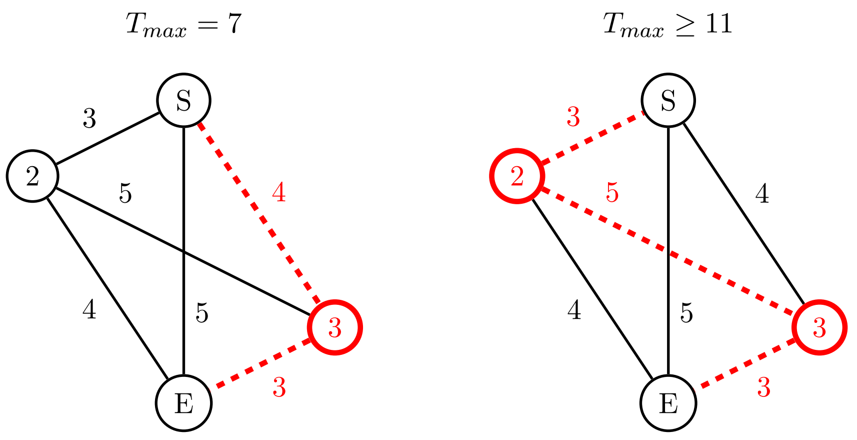


Figure 3.10: Example of the solutions of an orienteering instance by varying the total budget. The red (dashed) path represent the optimal solution with $T_{max} = 7$ on the left side and with any $T_{max} \geq 11$ on the right side.

The orienteering problem is a combination of node selection and shortest path computation between the graphs' nodes. As a consequence it can be cast as a combination of the Traveling Salesman Problem (TSP) problems [54] and the Knapsack Problem (KP), where the KP goal is to maximize the total score collected while the TSP aims at minimizing the travel distance. This formulation is also referred to as a generalized traveling salesman problem (GTSP) [77]. The orienteering problem is known to be an NP-hard, as it contains the well known traveling salesman problem as a special case.

An orienteering problem formulation arises in scheduling and routing applications, and it is also known as the selective traveling salesperson problem [113, 167] or the maximum collection problem [92]. A number of practical applications have been modeled as an orienteering problem and many heuristic approaches have been developed to combat the inherent complexity of the problem. In most cases, the orienteering problem is defined as a path to be found between distinct locations, rather than a circuit where $v_1 \equiv v_N$. However, in some applications v_1 can coincide with v_N but the difference between both formulations is not significant.

For a general review on the numerous variants and applications we suggest the surveys proposed by Vansteenwegen et al. [175] and Gunawan et al. [81].

3.8 Topological Skeletonization

In our SBOLSE algorithm presented in Chapter 4 we make use of a technique known as skeletonization as a preprocessing phase to reduce the number of points that we must consider when planning the path for a robotic platform.

In digital image processing and shape analysis the topological *skeletonization*, also known as thinning process or medial representation, is a process for reducing regions of an image to a thin (skeletal) representation while erasing most of the original pixels (see example in Figure 3.11). Further, algorithms has been developed and extended to extract a skeleton of 3D volumetric objects [185].

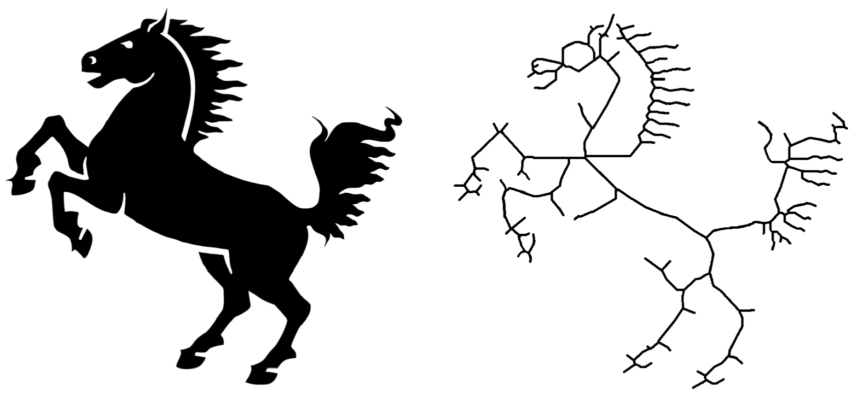


Figure 3.11: Example of a topological skeletonization applied to an image.

In general, the aim of the skeletonization is to extract a shape feature representing the general form of an object. The skeletonization preserves and usually emphasizes the geometrical properties of the shape, such as its topology, connectivity, direction and length.

Skeletonization was first introduced by using an intuitive model of fire propagation by [27]. If one “sets fire” at all points on the boundary of a shape, the skeleton forms at the points in the region where two or more “fires” meet. This intuitive description has different mathematical definitions and in the literature it is sometimes referred to as *medial axis* or *thinning* [78].

Skeletonization is an important step in pre-processing phase for many applications in digital image processing such as OCR (Optical Character Recognition), computer vision or path planning for a mobile robot among obstacles [65]. There are many techniques that are tailored for different application contexts. Such algorithms can vary in run time and properties of the produced skeleton, however they all significantly compress the input.

Part II

Informative path planning

Chapter 4

SBOLSE

4.1 Introduction

The use of robotic mobile sensors for environmental monitoring applications has gained increasing attention in recent years. In this context, a common application is to determine the region of space where the analyzed phenomena is above or below a given threshold level. This is the *level set estimation problem* as introduced in Section 3.2. One example is the analysis of water in a lake, where the operators might want to determine where the dissolved oxygen level is above a critical threshold value. Recent research proposes to model the spatial phenomena of interest using Gaussian processes, and then use an informative path planning procedure to determine where to gather data.

Our contribution presented in this part of the thesis is inserted in the aforementioned scenario, and aims at facing the problem of level set estimation. However, in contrast to previous works, we consider the case where a mobile platform with low computational power can continuously acquire measurements with a negligible energy cost. This scenario imposes a change in the perspective, since now efficiency is achieved by reducing the distance traveled by the mobile platform and the computation required by this path selection process.

Specifically, our techniques are motivated by the recent development of low-cost, small mobile platforms that can perform continuous-sampling in various body of waters (lakes, rivers and coastal areas). For example, consider the autonomous surface vessel shown in Figure 1.1. This platform is small (about 1 meter long and 50 cm wide) and it is equipped with probes that measure various parameters such as the dissolved oxygen, temperature, electrical conductivity and the pH level (see Figure 4.1) with sampling rate between 1 and 10 Hz while the platform is moving. In this setting the cost in terms of energy to perform a single measurement is negligible, and the most crucial issue for the data collection process is the energy consumed to move the vessel. In fact, to meet the payload constraint of this platform, batteries must have a limited capacity that results in constraints on total path length. In this scenario, our goal is then to minimize the total path length while collecting as much information as possible to correctly classify our domain of interest.

As a further constraint, we also want to take into account the low computational power of the hardware of this platform (composed of an Arduino Due board and an Android smartphone), which motivates the derivation of algorithms with reduced computational complexity.

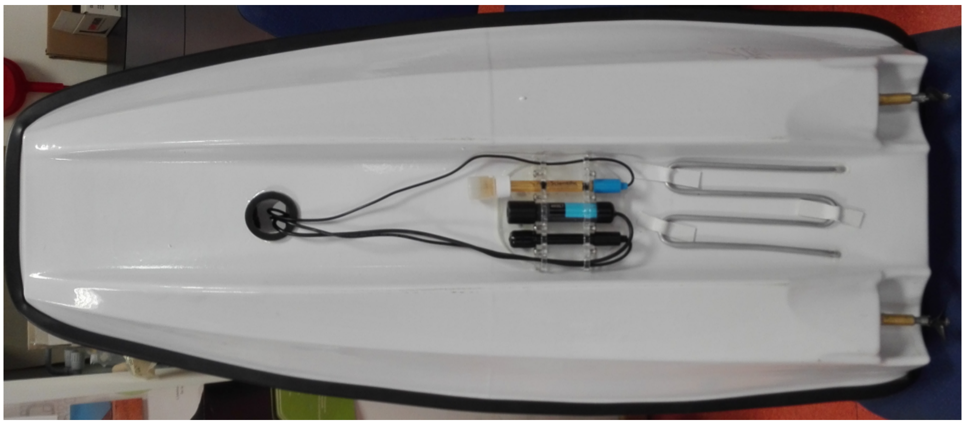


Figure 4.1: Picture of the bottom of the hull of the boat. We can observe that it is equipped with probes for different measurements capabilities.

Even if the LSE solutions proposed by Gotovos et al. [79] described in Section 3.2.1 proved to be effective and accurate, they are not suitable for our constrained scenario. Actually, with such methods the mobile sensor is guided toward the most informative locations without taking into account the path to reach such points. For example, the LSE algorithm assumes that the mobile sensor moves from one position to the next selected location following a straight line. Another issue is that the measure is collected only at the final location without considering all the points traversed by the sensor along its path. On the other hand, here we consider applications where sensors can provide data while the robotic platform is moving.

In general, when planning sampling paths, an important aspect is the trade-off between the quality of the paths and the timing of the decision. One can plan a high quality path, which can turn out to be worse then expected when it is followed and when the model of the environmental phenomenon is correspondingly updated.

In what follows we present our Skeleton-Based Orienteering for Level Set Estimation (SBOLSE) algorithm that makes use of an orienteering problem formulation for the level set estimation. It starts from the LSE framework but is specifically designed for continuous measuring sensors in which the cost (in terms of energy) required to take a measurement is negligible, but instead it is necessary to optimize the total path of the mobile platform to minimize battery consumption.

SBOLSE aims to obtain a high quality classification of the analyzed regions while optimizing the total path length required by the mobile agent, rather than the number of samples extracted during the executions (which is an important criteria for previous works in the level set domain). Moreover, to match the low computation power of mobile platforms, we introduce the use of several heuristics which significantly reduces the time required by the algorithm for the selection of an informative path.

Specifically, the main contributions to the state of the art are:

- We propose a novel algorithm called SBOLSE, that uses an orienteering formulation to solve the level set estimation problem. The algorithm is specifically designed for continuous-sampling mobile sensors.
- We propose four different heuristics with the aim to reduce the computation time required to determine an efficient path with the SBOLSE algorithm.
- We test our algorithms on a real world dataset of water pH level and on synthetic datasets (presented in Chapter 6). We show that our approaches are better in terms of computation time required and path length, while achieving a high quality classification when compared to the state of the art techniques for level set estimation.

Notice that, the SBOLSE algorithm is based on several methodologies derived from different areas of computer science: LSE from information gathering, skeletonization from image processing, orienteering from graph theory and clustering. Our work shows that a clever combination of such methodologies results in an effective approach for addressing level set estimation with continuous measurement sensors.

Although our techniques has been introduced for environmental monitoring operations, they can be generalized to different applications where mobile sensors are used to model the information of the environment. Specifically, applications where a mobile sensor has to take measurements from the environment with a battery constraint and hence it is required to compute an efficient path. Examples can span across different context such as search and rescue operations [156], precision agriculture [144, 169], sea-floor target localization [123] and radio signal source localization [158].

4.2 SBOLSE algorithm

The proposed SBOLSE algorithm is based on a Gaussian process model of the scalar field. It considers the knowledge about unclassified locations

Algorithm 1 SBOLSE algorithm

Input: set $\mathcal{X}$, threshold h, accuracy parameter ϵ,
prior known data $X \subset \mathcal{X}$, starting location x_{start}
Output: sets H and L

```
 1: $t \leftarrow 0$
 2: $\mathbf{x}_0 \leftarrow \mathbf{x}_{start}$
 3: $H_0 \leftarrow \emptyset, L_0 \leftarrow \emptyset, U_0 \leftarrow \mathcal{X}$
 4: while $H_t \cup L_t \neq \mathcal{X}$ do
 5:     $t \leftarrow t + 1$
 6:     Compute GP posterior $\mu(\mathbf{x})$ and $\sigma^2(\mathbf{x})$ for all $\mathbf{x} \in U_t$
 7:     Classify and update $H_t, L_t, U_t$ according to LSE [79]
 8:     $\mathbf{x}_c \leftarrow$ current position
 9:     $G \leftarrow buildGraph(\mathbf{x}_c, U_t)$
10:     $path \leftarrow orienteeringStep(G, budget)$
11:     Execute $path$
12: end while
13: $H \leftarrow H_t, L \leftarrow L_t$
```

$\mathbf{x}_i \in U_t$ (as done by Gotovos et al. [79] and previously described in Section 3.2.1) to build an orienteering problem instance and to select a sequence of locations to visit. The algorithm optimizes the information that can be acquired along the route while meeting the budget on the travel distance. Moreover, we propose a heuristic approach based on the topological skeletonization to combat the computational complexity associated with the orienteering problem, along with several heuristics to reduce the number of orienteering executions required. We empirically show that with these heuristics the classification accuracy does not suffer a significant degradation while greatly reducing the computation time.

The code of Algorithm 1 describes the steps of our SBOLSE approach. Our algorithm maintains three sets of points: the current sublevel L_t and superlevel H_t sets, as well as the set of unclassified points U_t. At each iteration t we update the GP posterior by integrating the new information gathered at the preceding iteration (line 6). Then we compute the confidence intervals $C_t(\mathbf{x})$ for each point $x \in U_t(\mathbf{x})$, classify them into one of the three sets, and then compute the sequence of locations to be visited. To compute such a path we consider the ambiguity defined by equation (3.14) of the unclassified points and build an orienteering problem instance. Specifically, in line 9 we create a graph from the unclassified points U_t (Algorithm 2) and then compute a path (line 10) using the orienteeringStep procedure of Algorithm 3. The algorithm terminates when $H_t \cup L_t = \mathcal{X}$, i.e. when all points are classified and thus $U_t = \emptyset$. Note that during the execution of the path (Algorithm 1, line 11) if the sensor moves over locations not yet analyzed but already classified according to LSE technique [79], these are evaluated and possibly re-classified considering newly acquired data.

4.2.1 Building the graph

Algorithm 2 buildGraph procedure

Input: current position x_c, unclassified elements U_t
Output: weighted graph G

```
1: $V \leftarrow v_1 \equiv \mathbf{x}_c$
2: $w(v_1) \leftarrow 0$
3: $n \leftarrow 1$
4: for all $\mathbf{x}_i \in U_t$ do
5:    $n \leftarrow n + 1$
6:    $V \leftarrow V \cup v_n \equiv \mathbf{x}_i$
7:    $w(v_n) \leftarrow a(\mathbf{x}_i)$
8: end for
9: $E \leftarrow \emptyset$
10: for all $v_i \in V$ do
11:    for all $v_j \in V$ do
12:       if $v_i \neq v_j$ then
13:          $E \leftarrow E \cup e_{ij}$
14:          $w(e_{ij}) \leftarrow ||v_i - v_j||$
15:       end if
16:    end for
17: end for
18: $G \leftarrow (V, E)$
```

In the buildGraph procedure we take all the unclassified locations U_t, and we build an undirected weighted graph, where all nodes are connected. This graph will then be used in the orienteering procedure.

As shown in Algorithm 2, the first node of the graph represents the current location of the mobile sensor (line 1). This location represents the starting position for the orienteering solver. Subsequently we build the nodes set V and the edges set E. The function $w(\cdot)$ denotes the weight of a node or the weight of an edge. The weight of a node $w(v_i)$ (line 7) corresponds to the ambiguity measure (equation 3.14) of the location that the node represents. The weight of the first node is an exception as this represents the current position of the mobile sensor, hence the location has been already visited and classified. The weight of the edges $w(e_{ij})$ (line 14) denotes the travel distance between the locations represented by the nodes v_i and v_j.

4.2.2 Orienteering Step

Algorithm 3 orienteeringStep procedure

```
Input: graph $G = (V, E)$, budget $B$
Output: $bestPath$
1: $bestPath \leftarrow \emptyset$
2: $bestPathValue \leftarrow 0$
3: for $i$ in range$(2, |V|)$ do
4:     if $||v_1 - v_i|| \leq budget$ then
5:         $path \leftarrow orienteeringHeuristic(G, v_1, v_i, B)$
6:         if $value(path) > bestPathValue$ then
7:             $bestPath \leftarrow path$
8:             $bestPathValue \leftarrow value(path)$
9:         end if
10:    end if
11: end for
```

In the orienteeringStep procedure we use the undirected weighted graph G previously built and consider this as the input to the orienteering problem. In particular we have a fixed starting point (i.e. the current location of the sensor), but we do not have an ending location (which is required in the classical formulation of the orienteering problem).

Please note that, in principle, it would clearly make sense to design an orienteering problem instance where the starting point is equal to the destination point. However in the classic orienteering problem the rewards of every node are fixed: in our case, rewards change during the execution of the algorithm since the information value of every point decreases while the sensor acquires new data. Hence, making a single run of orienteering with a budget equal to the total battery lifetime, and with a starting point equal to the ending point, would not take into account the dynamics of the information inherent in such a scenario. Therefore, we iterate the process for smaller segments, and this allows us to update the model (with a Gaussian process update) more frequently, considering the newly acquired data. When measuring at a point we also obtain information about nearby locations. Frequent updates allow the algorithm to make better decisions about future path choices. In other words, the choice of the budget (length) of these segments allows a trade-off between adaptivity and the horizon of our path planning procedure.

To choose the destination we perform an orienteering procedure multiple times (Algorithm 3, line 5), assuming as destination every unclassified point in the graph that is reachable with the given budget. The choice of repeating the orienteering step multiple times (one for every possible reachable destination) represents the simplest choice for formalizing this problem and this aspect is improved with the end-point heuristics described in Section

4.3.

Every time we solve an orienteering instance with a different destination point we obtain a new path. The procedure keeps track of the best discovered one and returns this as final route to be executed from the SBOLSE algorithm. Specifically with *value*(*path*) (line 6 and 8) we indicate the summation of the nodes' weights in that route, that is $value(path) = \sum_{v_i \in path} w(v_i)$. Since the orienteering problem aims at maximizing the score for a given travel budget, using this procedure we obtain a path that maximizes the information collected about the unclassified locations for the level set estimation problem.

4.2.3 Skeletonization

In practical applications of the level set estimation problem the input is a set of dense points that must be classified. Specifically, when the data acquisition process starts, we must consider the entire surface of the selected portion of the environment. These data are typically discretized and organized in a grid where each entry represents a small portion of the surface (i.e., a square of 50 centimeters or 1 meter in our experiments).

Now, given the smoothness property of the environmental phenomena, locations with high classification uncertainty usually cluster into areas where the unknown scalar field has higher probability to cross the threshold level. Considering all such points is redundant and this motivates the use of the topological skeletonization technique to compress the input.

Specifically, we consider the grid containing the information about the ambiguity measure (equation 3.14) of the unclassified points U_t as a binary image, where unclassified points are set to 1 and classified points are 0. We then apply a skeletonization technique (introduced in Section 3.8) to such image, and we maintain as interesting points to be classified only the points of the resulting skeleton. This greatly reduces the number of locations that we must consider in the *buildGraph* procedure previously presented in section 4.2.1 (see an example in Figures 4.2a and 4.2b). Note that in the *buildGraph* procedure each point that is maintained after the skeletonization represents a node of a complete graph.

4.3 Orienteering with end-point heuristics

The major computation bottleneck of the naive SBOLSE algorithm can be identified in the multiple executions of the orienteering step (Algorithm 3, line 5), assuming as its destination every unclassified location in the graph that is reachable with the given budget. Hence, we aim at reducing the number of the orienteering executions by selecting only a subset of the unclassified locations as potential end points.

The new orienteering step procedure is described in Algorithm 4. We can notice that the only differences with respect to Algorithm 3 are in line 3, where we determine a new set of nodes V' using a heuristic, and line 4 that loops on the newly created V' instead of V.

Algorithm 4 orienteeringStep procedure with heuristics

```
Input: graph $G = (V, E)$, budget $B$
Output: $bestPath$
1: $bestPath \leftarrow \emptyset$
2: $bestPathValue \leftarrow 0$
3: $V' \leftarrow heuristic(V)$
4: for $i$ in range(2, $|V'|$) do
5:   if $||v_1 - v_i|| \leq budget$ then
6:     $path \leftarrow orienteeringHeuristic(G, v_1, v_i, B)$
7:     if $value(path) > bestPathValue$ then
8:       $bestPath \leftarrow path$
9:       $bestPathValue \leftarrow value(path)$
10:    end if
11:  end if
12: end for
```

In what follows we propose the main heuristic (EBC in Section 4.3.1) that we implemented and used in order to determine the new locations set V'. Additionally, three baseline heuristics that we used for comparisons are described in Section 4.3.2.

4.3.1 EBC heuristic

The main heuristic that we propose is based on the Exemplar Based Clustering (EBC) performed with the Affinity Propagation technique [73] introduced in Section 3.6.2. The main idea behind this technique is to exploit the Affinity Propagation algorithm on the unclassified locations and use the selected exemplars as the set of valid end points for the orienteering procedure.

As described in Section 3.6.2 the affinity propagation procedure takes as input a set of real-valued similarities between data points, where the similarity $s(i, k)$ specifies how well the data point with index k is suited to be the exemplar for data point i, and a set of real numbers referred to as “preferences” which identify the preference of a location to become an exemplar.

In our application what we want to obtain is a set of points reasonably scattered in space and with high information content. Similarities and preferences have to be set accordingly. Specifically, the similarity between two points is related to their proximity, and preferences are related to the informativeness. Specifically values has been set as follows:

- **Similarity:** We want to associate the proximity of two points to their similarity. Moreover all the similarity measures have to be positive. To do so we compute the maximum distance possible between any two locations (that we identify as $maxDist$) and we set the similarity between point $\mathbf{x}_i$ and point $\mathbf{x}_j$ as $s(\mathbf{x}_i, \mathbf{x}_j) = (maxDist - ||\mathbf{x}_i - \mathbf{x}_j||)$. The set of similarity thus obtained is normalized so as to have all values between 0 and 1.

- **Preference:** We want to associate the informativeness of a location with the preference to become an exemplar for other neighboring points. To do so we simply set the preference for a point $\mathbf{x}_i$ with the ambiguity measure of that location, that is $s(\mathbf{x}_i, \mathbf{x}_i) = a_t(\mathbf{x}_i)$ with $a_t(\mathbf{x}_i)$ computed as in equation 3.14. The set of preferences is then normalized so as to have all values between 0 and 1.

An example of the effects of the Exemplar Base Clustering phase on the real dataset is shown in Figures 4.2c and 4.2d. Specifically, we can observe in Figure 4.2c that the exemplar points selected by the procedure are not uniformly distributed in space but rather they are more concentrated in areas with high information content as we would expect.

4.3.2 Orienteering end-point baseline heuristics

Here we describe three different baseline end-point heuristics for SBOLSE that we implemented. Even if they represent common straightforward approaches to select elements from a set, with the reduction of the number of orienteering executions, their impact in the reduction of computational complexity of our algorithm is substantial. Specifically, we tested the following methods:

- **Random(p):** With this simple heuristic, we randomly select a specified percentage p of the unclassified locations, in order to become the set of valid orienteering end points.

- **Sparse(p):** With this second heuristic, we select from the set of unclassified locations the specified percentage p of points with the higher ambiguity value (i.e. the locations with the higher amount of information).

- **Sample(p):** With this last heuristic we perform a discrete random sampling, where the probability of a point to be selected is weighted by the ambiguity value of that location.

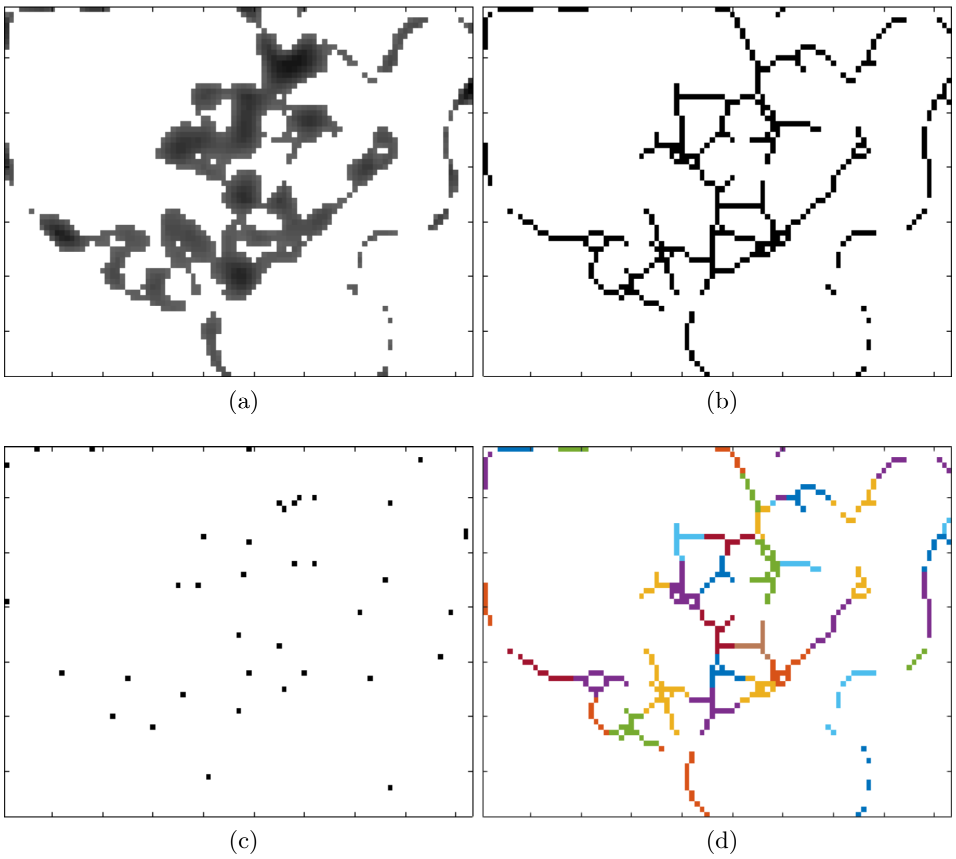


Figure 4.2: Example of the topological skeletonization and Exemplar Based Clustering heuristic applied to the data matrix containing the ambiguity measure for the unclassified points U_t. 4.2a data matrix before the skeletonization, a darker color corresponds to an higher value of ambiguity. 4.2b matrix after the skeletonization operation. 4.2c Exemplars selected with the EBC heuristic and in 4.2d the corresponding clusters identified with different colors.

4.4 Theoretical analysis

For what concerns the theoretical analysis of our approach, notice that Gotovos et al. [79] with Theorem 1 prove the convergence of the LSE algorithm. Even though the selection procedure of our SBOLSE algorithm differs from LSE, we used the same classification rules (Algorithm 1, lines 6-7). As in LSE, our technique iterates with the while loop until every point is classified. Hence we can ensure the convergence of the SBOLSE algorithm with a high quality classification as they do.

The computational complexity of the technique can be described as follows. Let's consider the worst case scenario; this scenario is represented by the case where at each iteration of the algorithm we move the sensor in a location adjacent to the current position and we are able to classify only this new measured location. In this case the while loop of the algorithm has to be performed $|\mathcal{X}|$ times.

The complexity of the body of the loop is the sum of four main components:

1. The computation of the Gaussian Process that requires $\mathcal{O}(|\mathcal{X}|^3)$ due to the need to invert a $|\mathcal{X}| \times |\mathcal{X}|$ matrix.

2. The execution of the *buildGraph* sub-procedure which has a complexity $\mathcal{O}(|U_t|^2)$. The cardinality of set U_t is $|\mathcal{X}|$ at the first iteration and decreases over time according to how many points have been classified.

3. The classification according to the LSE algorithm [79] that requires $\mathcal{O}(|\mathcal{X}|)$.

4. The execution of the *orienteeringStep* sub-procedure.

The complexity of the fourth point depends on the actual heuristics used and very efficient solutions can be found, $\mathcal{O}(log^2 OPT)$ where OPT is the number of nodes visited by an optimal solution [45]. As previously mentioned in Section 4.3 the major bottleneck is the multiple executions of the orienteering step that, in the worst case scenario, are $|\mathcal{X}|-t$. Even with a less efficient implementation of the orienteering heuristic (e.g. a $\mathcal{O}(|\mathcal{X}|^3)$ the execution of the orienteeringStep sub-procedure is on the order of $\mathcal{O}(|\mathcal{X}|^4)$.

To conclude, the combined complexity of the algorithm is on the order of $\mathcal{O}(|\mathcal{X}|^5)$. Although the complexity seems very high, this is by far the worst case scenario. Consider that, in practical applications the algorithm does not classifies a single point at each iteration but rather a set of new locations. Moreover, with the use of the skeletonization and the end-point heuristics the computational effort associated to the orienteeringStep is significantly reduced, hence, the algorithm can run much faster. In our empirical evaluation proposed in Chapter 6, we also detail the actual computation time required to perform the technique using two different datasets.

Chapter 5

PULSE

5.1 Introduction

As previously introduced in Chapter 4, our contribution presented in this part of the thesis is inserted in the framework of the level set estimation problem for continuous sampling mobile sensors. In this chapter we focus in particular on the need to develop procedures that take into account the low computation power of the processing units used on the platforms.

Specifically, in what follows we present another algorithm, which we call Path-Update LSE (PULSE) algorithm. Similarly to the SBOLSE algorithm presented in Chapter 4, this is specifically designed for continuous sampling sensors where:

- The cost in terms of energy consumption required to perform an individual measurement is negligible.
- It is necessary to optimize the total path of the agent in order to reduce the battery consumption.
- We need a computationally efficient path selection procedure.

The proposed technique determines an informative path in order to reach the most interesting location (i.e. the point in space with the highest ambiguity $a(\mathbf{x}_i)$ about its classification), moving from the current position through points that still have to be classified.

In contrast to SBOLSE, in which the orienteering routine considers the amount of information in each location, this PULSE technique is a greedy approach that builds a path using only the presence of the information in a location without taking into account the amount. Moreover, in this case we do not have a budget that gives us a trade-off between adaptivity and path planning horizon. The purpose of this algorithm is to develop a fast baseline technique for continuous sampling sensors that ignores the amount of information.

Specifically, the main contributions of this chapter are:

- We propose a novel greedy algorithm called PULSE for selecting measurement paths that exploits a less accurate but computationally faster path selection procedure. It is used as a baseline strategy for comparisons in the continuous-sampling setting.
- We derive a batch variant of PULSE to allow a further trade-off between the computation time required and the path efficiency.
- As done for the SBOLSE technique, we test our algorithms on a real world dataset of water pH level and on synthetic datasets (presented in Chapter 6).

Also in this case the techniques have been introduced for environmental monitoring operations but they can be generalized to different applications where mobile sensors are used to model the information content of the environment. Specifically, applications where a mobile sensor has to take measurements from the environment with a battery constraint and a low computation capacity.

5.2 PULSE algorithm

The pseudo-code of Algorithm 5 describes the steps of our PULSE approach. It is very similar to SBOLSE (differences in lines 8-9-10). The algorithm maintains at any iteration t three sets of points: the current superlevel H_t and sublevel L_t sets, as well as the set of unclassified points U_t. At each iteration t we update the Gaussian Process posterior by integrating the new information gathered at the preceding iteration (line 6). Then we compute the confidence intervals $C_t(\mathbf{x})$ for each point $\mathbf{x} \in U_t(\mathbf{x})$, classify them in one of the three sets (line 7) and then compute the next sample to be evaluated using the ambiguity defined by equation (3.14) (line 9). We then compute a path between the current location $\mathbf{x}_{t-1}$ and the selected point $\mathbf{x}_t$ (line 10) using the path selection procedure presented in Algorithm 6. The algorithm terminates when $H_t \cup L_t = \mathcal{X}$, i.e. when all points are classified and thus $U_t = \emptyset$. Note that during the execution of the path (Algorithm 5, line 11) if an agent moves through locations that are already classified, these are re-evaluated and re-classified considering newly acquired data.

5.2.1 Path Selection

Here we describe in detail the pseudo-code of Algorithm 6. At each time step t the algorithm keeps track of the starting position $\mathbf{x}_{t-1}$ of the platform (i.e. the last position) and the destination point assigned by the sample selection criteria, i.e. the most interesting point $\mathbf{x}_t$. In order to select

Algorithm 5 PULSE algorithm

```
Input: set $\mathcal{X}$, threshold $h$, accuracy parameter $\epsilon$,
prior known data $X \subset \mathcal{X}$, starting location $x_{start}$
Output: sets $H$ and $L$
 1: $t \leftarrow 0$
 2: $\mathbf{x}_0 \leftarrow \mathbf{x}_{start}$
 3: $H_0 \leftarrow \emptyset, L_0 \leftarrow \emptyset, U_0 \leftarrow \mathcal{X}$
 4: while $H_t \cup L_t \neq \mathcal{X}$ do
 5:     $t \leftarrow t + 1$
 6:     Compute GP posterior $\mu(\mathbf{x})$ and $\sigma^2(\mathbf{x})$ for all $\mathbf{x} \in U_t$
 7:     Classify and update $H_t, L_t, U_t$ according to LSE [79]
 8:     $\mathbf{x}_{t-1} \leftarrow \mathbf{x}_t$
 9:     $\mathbf{x}_t \leftarrow$next location according to LSE [79]
10:     $path \leftarrow$ pathSelection($\mathbf{x}_{t-1}, \mathbf{x}_t, U$)
11:     Execute $path$
12: end while
13: $H \leftarrow H_t, L \leftarrow L_t$
```

an informative path towards the destination, the path selection procedure analyzes each point $\mathbf{x} \in U_t$, i.e. locations that still have to be classified and therefore potentially carrying some useful information, selecting a path $\{\mathbf{x}_{t-1} = \mathbf{x}_{next_0}, \mathbf{x}_{next_1}, \cdots, \mathbf{x}_{next_n} = \mathbf{x}_t\}$ with $n \geq 1$. Note that the number of points touched by the agent, n, is automatically determined by the procedure. In the case of $n = 1$ the path corresponds to the straight line from the current position to the selected destination.

Each $\mathbf{x}_{next_i}$ point determined by the procedure meets the condition to always approach the destination point, i.e.

$$||\mathbf{x}_{next_i} - \mathbf{x}_t|| < ||\mathbf{x}_{next_{i-1}} - \mathbf{x}_t|| \tag{5.1}$$

where $||\mathbf{x}' - \mathbf{x}''||$ is the Euclidean distance between locations $\mathbf{x}'$ and $\mathbf{x}''$. In more detail, given the two points $\mathbf{x}_{next_{i-1}}$ and $\mathbf{x}_t$, the region of the space which contains points meeting this condition defines a convex area (see example in figure 5.1) and we call this area A_{t_i} (Algorithm 6, lines 8-10). The procedure analyzes all points $\mathbf{x}_i \in U_t \cap A_{t_i}$ and selects as $\mathbf{x}_{next_i}$ the closest point from the previous location, generating the path (Algorithm 6, line 11).

Note that Algorithm 5 differs from the LSE Algorithm proposed by [79] in the path selection procedure we employ. Specifically, our path selection procedure selects only points that meet the condition in equation (5.1) (Algorithm 6, lines 6 and 9). As we build the path from $\mathbf{x}_{t-1}$ to $\mathbf{x}_t$ the area A_{t_i} shrinks and converges towards the destination point $\mathbf{x}_t$. This allows the path to include informative points that lie inside this area (example in Figure 5.1), while ensuring that the path is going towards the most interesting

Algorithm 6 pathSelection procedure

```
Input: last position $\mathbf{x}_{t-1}$, next location $\mathbf{x}_t$, unclassified elements $U_t$
Output: $path$
1: $i \leftarrow 0$
2: $\mathbf{x}_{next_0} \leftarrow \mathbf{x}_{t-1}$
3: $path \leftarrow \mathbf{x}_{t-1}$
4: while $\mathbf{x}_{next_i} \neq \mathbf{x}_t$ do
5:     $i \leftarrow i + 1$
6:     $d \leftarrow ||\mathbf{x}_{next_{i-1}} - \mathbf{x}_t||$
7:     $A \leftarrow \emptyset$
8:     for all $\mathbf{x} \in U_t$ do
9:         if $||\mathbf{x} - \mathbf{x}_t|| < d$ then
10:            $A \leftarrow A \cup \mathbf{x}$
11:        end if
12:    end for
13:    $\mathbf{x}_{next_i} \leftarrow min_{\mathbf{x} \in A}\Big(||\mathbf{x}_i - \mathbf{x}_{next_{i-1}}||\Big)$
14:    $path \leftarrow path \cup \mathbf{x}_{next_i}$
15: end while
```

point defined by the ambiguity measure $a_t(x)$ presented in equation 3.14.

5.3 Theoretical analysis

For what concerns the convergence analysis of this approach the same argument made for the SBOLSE algorithm in Section 4.4 is valid. Notice that Gotovos et. al. with Theorem 1 in [79] proves the convergence of their LSE algorithm. Considering that PULSE uses the same classification and selection rules of LSE, the path selection procedure builds a path that terminates in the selected location, and that the information gathered along the path is never less than 0, the convergence is also valid for the technique proposed in this chapter.

As previously done for the SBOLSE algorithm in Chapter 4, to describe the computational complexity of the technique let us consider the worst case scenario. This scenario is represented by the case where at each iteration of Algorithm 5 we move the sensor in a location adjacent to the current position and we are able to classify only this new measured location. In this case the while loop of the algorithm has to be performed $|\mathcal{X}|$ times.

The complexity of the body of the loop is the sum of three main components:

1. The computation of the Gaussian Process that requires $\mathcal{O}(|\mathcal{X}|^3)$ due to the need to invert a $|\mathcal{X}| \times |\mathcal{X}|$ matrix.

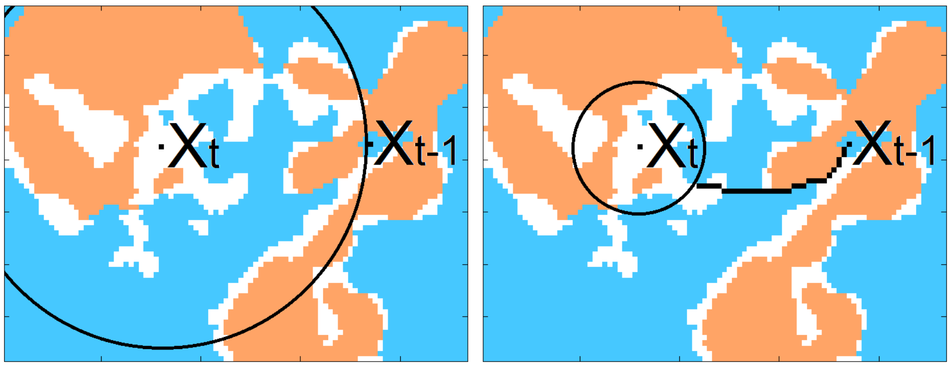


Figure 5.1: Example of runtime execution of the path selection procedure. The white areas represent location that are still unclassified. The circle represents the area A. On the left side the beginning of the procedure and on the right side we can observe the path that has been built after some iterations.

2. The classification according to the LSE algorithm [79] that requires $\mathcal{O}(|\mathcal{X}|)$.
3. The execution of the *pathSelection* sub-procedure described in Algorithm 6.

As we described in the previous section, our path selection procedure selects only points that meet the condition in equation (5.1) (Algorithm 6, lines 6 and 9). As we build the path from the current position to the next point defined by the ambiguity measure $a_t(x)$ the procedure selects a set of informative locations. However, in the worst case scenario that we are analyzing this path selection procedure has a constant complexity.

To conclude, the overall combined complexity of the PULSE algorithm is in the order of $\mathcal{O}(|\mathcal{X}|^4)$. Also in this case, although the complexity seems very high, this is by far the worst case scenario. Consider that, in practical applications the algorithm does not classifies a single point at each iteration but rather a set of new locations. In our empirical evaluation proposed in Chapter 6, we also detail the actual computation time required to perform the technique using two different datasets.

5.4 Batch variant

In active learning techniques the “batch” concept is a simple variant where instead of selecting a single point to be labeled a new batch (set) of points is selected before updating the model.

This same concept has been used by Gotovos et al. [79] in the context of the level set estimation problem. Specifically, in addition to the LSE algorithm, authors in [79] discuss the batch version where multiple locations are selected by taking the mutual information into account. The Gaussian process is then updated only after every location of the batch has been measured. Although the main goal of the LSE Batch approach is to select multiple locations and to compute a path between them, their algorithm is far in spirit from what we propose in this thesis: actually, as in the regular LSE algorithm, also in the batch variant the length of the path is not a variable explicitly considered in the choice of the points to be sampled next. The main reason for this is that in both cases their main goal is to minimize the number of sampling locations. Moreover, during the movement of the mobile platform from one location to next one, the monitoring process does not acquire any further data.

Following this idea, besides the PULSE algorithm we provided a simple batch variant that aims at selecting *a set* of informative locations in a single iteration (i.e. after a single Gaussian process update). This will act as a trade-off between the path's efficiency and the computation time required since we can greatly reduce the number of Gaussian processes updates.

Specifically, we exploit the property introduced in Section 3.1 that the predictive variance of Gaussian processes (equation 3.7) depends only on the location of a measurement and not on the measured value itself. Assuming we will obtain a new sample at some location, it is possible to evaluate the updated predictive variance, and thus the new ambiguity value, of every point $\mathbf{x}_i \in U_t$. This process is repeated adding the location with the new highest ambiguity to a set.

We can observe the pseudo-code of the batch procedure in Algorithm 7. After the batch is selected (line 8) it is possible to compute an efficient path that visits all the locations in such a set. The order in which those locations should be visited is determined by solving a Traveling Salesman Problem (TSP) [7] (line 9). Once we have the order of locations to be visited, the path selection procedure (Algorithm 6) is applied to all pairs of consecutive locations in order to obtain the final informative path (lines 10-13). This algorithm allows us to trade-off the adaptivity we would have by updating the Gaussian process more often in favor of a reduction of the computation required to compute the path.

Algorithm 7 PULSE batch algorithm

Input: set $\mathcal{X}$, threshold h, accuracy parameter ϵ, size of the batch b prior known data $X \subset \mathcal{X}$, starting location x_{start}
Output: sets H and L

```
 1: $t \leftarrow 0$
 2: $\mathbf{x}_0 \leftarrow \mathbf{x}_{start}$
 3: $H_0 \leftarrow \emptyset, L_0 \leftarrow \emptyset, U_0 \leftarrow \mathcal{X}$
 4: while $H_t \cup L_t \neq \mathcal{X}$ do
 5:     $t \leftarrow t + 1$
 6:     Compute GP posterior $\mu(\mathbf{x})$ and $\sigma^2(\mathbf{x})$ for all $\mathbf{x} \in U_t$
 7:     Classify and update $H_t, L_t, U_t$ according to LSE [79]
 8:     $\mathbf{X}_t \leftarrow$set of $b$ locations according to LSE [79]
 9:     $\hat{\mathbf{X}}_t = \langle \mathbf{x}_{t^0} = \mathbf{x}_{t-1}, \mathbf{x}_{t^1}, \dots \mathbf{x}_{t^b} \rangle \leftarrow TSP(\mathbf{X}_t)$
10:     for $j = 0$ to $b - 1$ do
11:         $path \leftarrow$ pathSelection($\mathbf{x}_{t^j}, \mathbf{x}_{t^{j+1}}, U$)
12:         Execute $path$
13:     end for
14:     $\mathbf{x}_t \leftarrow \mathbf{x}_{t^b}$
15: end while
16: $H \leftarrow H_t, L \leftarrow L_t$
```

Chapter 6

Empirical evaluation

6.1 Introduction

In this chapter we present the empirical evaluation of our proposed techniques previously presented in Chapters 4 and 5, comparing them with literature alternatives on two datasets and analyzing different aspects of the approaches. More in detail, in Section 6.2 we describe the two datasets that we used in our experiments; in Section 6.3 we present the main comparison between our algorithms and other state-of-the-art competitors and in Section 6.4 we specifically evaluate the SBOLSE orienteering end-point heuristics described in Section 4.3. Finally, in Sections 6.5 and 6.6 we present the complete comparisons among all the experiments performed and draw conclusions.

The set of algorithms that we compared in our experiments are the following:

- **SBOLSE:** Our algorithm described in Chapter 4.
- **SB-EBC:** SBOLSE algorithm with Exemplar Based Clustering heuristic as described in Section 4.3.1.
- **PULSE:** Our algorithm as explained in Chapter 5.
- **PULSE$_{bXX}$:** This algorithm is the batch variant of PULSE as described in Section 5.4.
- **CS:** This is a variant of LSE as described by [79] for the continuous measuring setting. Locations on the straight line between the last position and the next selected point are analyzed, simulating a continuous sampling sensor.
- **CS$_{bXX}$:** Similar to CS, this is a variant of LSE batch as described by [79] for the continuous measuring setting.

- **ARS-CIPP$_n$**: This a recent adaptive re-planning scheme algorithm proposed by Hitz et al. in [87]. The number n represents the number of control points of the B-spline that is optimized by the algorithm.

6.1.1 Setting of the algorithms and aims

Regarding our SBOLSE algorithm and its heuristics variants, we implemented a simple orienteering algorithm inspired by the *center of gravity* technique as proposed by Golden et al. in [77]. Notice that the performance of the SBOLSE technique presented in the following sections depends on the performance of the orienteering heuristic implemented and this can be substantially improved (e.g., by using a more advanced heuristic available in literature). Nevertheless, even with such a simple orienteering heuristic our techniques can compute informative paths that achieve a high quality classification while significantly reducing the travel distance required by the mobile sensor.

For the SBOLSE algorithm with the EBC heuristic we set the measures required by the affinity propagation algorithm as previously explained in Section 4.3.1. We performed the skeletonization with a basic technique, based on morphological operators, as implemented in the MATLAB function `bwmorph`.

In the two batch versions, XX identifies the cardinality of the batch set, i.e. the number of locations in a TSP.

The aims of this empirical evaluation are to assess the quality of the selected paths, showing that our techniques are competitive in terms of total traveled distance required to obtain a high quality classification and the gain in terms of computation time required by our techniques with respect to the state of the art for the level set estimation problem. We assess the accuracy of the classification using the F$_1$-score. This is typically used in information retrieval to measure the accuracy of binary classification. Here we consider the locations in the superlevel set as positives and the locations in the sublevel set as negatives. All the described algorithms have been implemented and tested using MATLAB R2016a on a AMD FX 6300 processor with 16GB RAM.

All the results presented in this chapters refers to experiments where the starting location has been assumed to be on the top left corner of the dataset for all the algorithms. However, the variation on the performance given different starting locations is sufficiently small such that the statistics presented are representative for any possible starting point.

6.2 Dataset

As mentioned before we performed our experiments on two different datasets, a real-world and a synthetic dataset.

The real-world dataset consists of measurements of the pH level extracted from waters of the Persian Gulf near Doha, Qatar using the boat in Figure 1.1. The data forms a 68×93 grid where each element represents a sampling location $\mathbf{x}_i$ that must be classified with respect to a given threshold. Each point of the grid represents 0.5 square meters of the surface that has been analyzed. The value associated with that location is the average of all the samples extracted by the sensors while moving the boat in that portion of the surface. In our experiments we applied three different thresholds (7.40, 7.42 and 7.44) to classify the scalar field. We then assessed the results starting from ten random initial priors composed by 10% of the points in the grid, for a total of 30 tests with every algorithm. These priors were used to fit the hyper-parameters of an isometric Matérn-3 [149] kernel function. We can observe the ground truth of this dataset in Figure 6.1.

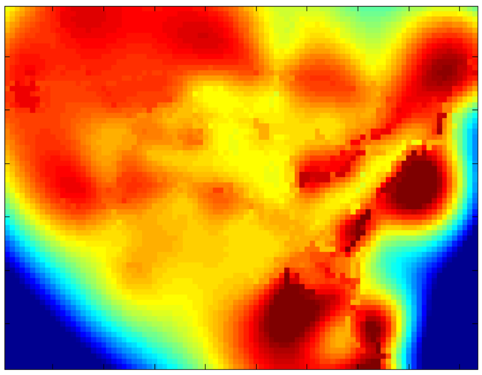


Figure 6.1: Scalar field of the real-world dataset, i.e. the pH level of waters extracted in the Persian Gulf near Doha (Qatar).

The synthetic dataset consists of ten 60×179 grids. The motivation for using this dataset is to test the techniques with more than 10,000 locations to classify hence comparing the algorithms on larger instances of the problem. The dataset has been extracted from portions of CO_2 maps[1] in order to obtain a scalar field with a topology consistent with typical environmental phenomena. We assume that each location represents 1 square meter of surface to analyze, and we used a threshold value equal to 85% of the maximum value in the scalar field. We assessed the results with five random initial priors for the Gaussian process composed of 10% of the points in the grid. The priors were used to fit the hyperparameters of an isometric Matérn-3 [149] kernel function. With five priors per grid and ten grids, we performed a total of 50 tests with each algorithm. We can observe the ground truth of one of this dataset in Figure 6.2.

We assume that there are no natural or artificial barriers inside both datasets that would prevent a mobile sensor to move and that would gen-

[1]http://oco.jpl.nasa.gov/galleries/gallerydataproducts/

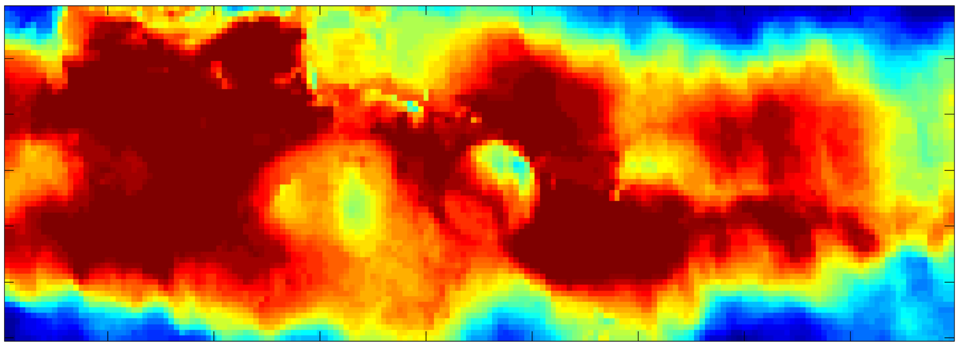


Figure 6.2: Example of one scalar field of the CO_2 synthetic dataset.

erate abrupt changes in the phenomena. In cases were a barrier is present, the kernel used in the Gaussian process should be adapted accordingly. Although this is an interesting scenario, it falls outside of the scope of the current empirical evaluation.

6.3 Results

6.3.1 Real-world dataset experiments

For what concerns the real-world dataset, as done in previous approaches [79] we performed tests to determine the β and ϵ parameter values that allow a high accuracy for all the algorithms. The parameter β represents a scaling factor for the interval described by Equation 3.9 whereas ϵ is an accuracy parameter used to relax the classification conditions (Equations 3.11 and 3.12).

Specifically, the scaling parameter β has been set to ensure that the resulting confidence intervals $C_t(\mathbf{x})$ (Equation 3.10) for every point $\mathbf{x}$ contain the scalar field $f(\mathbf{x})$ with high probability. The accuracy parameter ϵ has been chosen after testing different values that vary between 2% and 20% of the maximum value of the dataset.

For the batch algorithms we performed tests with batches of different sizes. We did not observe a significant reduction of the total traveled distance with batches of size larger than 30. Thus, we carried out the comparisons with batches of 30 points. For the ARS-CIPP$_n$ we performed tests with 3-7-11-15-19 control points. In the following tables we report the results with 7 control points since this configuration has obtained the best results both in terms of total traveled distance and runtime. The additional results can be found in Section 6.5.

As we can observe in Table 6.1 the F_1-score is consistently higher than 97% for all the algorithms. Regarding the total traveled distance our SBOLSE algorithm performs very well, with a traveled distance that is lower than all

Table 6.1: F_1-score, total traveled distance (meters) and computation time (seconds) using the real world pH dataset. $\overline{x}$ is the average of all 30 experiments and $SE_{\overline{x}}$ is the standard error of the mean.

	F_1-score		**Traveled dist. (m)**		**Comp. time (s)**	
	$\overline{x}$	$SE_{\overline{x}}$	$\overline{x}$	$SE_{\overline{x}}$	$\overline{x}$	$SE_{\overline{x}}$
PULSE	97.46	0.063	587.8	10.82	**11.1**	0.27
PULSE_{b30}	97.43	0.060	518.7	6.68	63.5	0.89
CS	**98.22**	0.039	1560.8	18.58	38.1	0.49
CS_{b30}	97.47	0.055	671.7	13.71	82.4	1.74
SBOLSE	97.23	0.066	**473.6**	6.20	1006.2	45.99
SB-EBC	97.25	0.064	495.1	8.08	124.4	4.19
ARS-CIPP_7	97.57	0.045	736.1	10.30	114.5	1.70

other techniques but with a significantly higher computation time. SBOLSE with our EBC heuristics represents the best trade-off between total path and computation required, but with a time that is one order of magnitude lower than the basic version without the heuristic. Moreover, notice that the performance of SBOLSE depends on the orienteering algorithm implemented. As previously stated in Section 6.1.1 these can be substantially improved.

We can observe that PULSE_{b30} represents a good trade-off as well, however the average path required from the SB-EBC is lower and statistically significant according to a t-test with $\alpha = 0.05$. It is possible to observe a graphical representation of the different paths chosen by CS, SBOLSE, PULSE and ARS-CIPP in Figure 6.3.

Notice that, results reported in Table 6.1 represents a full execution of the algorithms until convergence is reached. In case of a limited budget (i.e., a limited total travel distance that the mobile sensor can run) such that it is not possible to classify all points, our SBOLSE and SB-EBC obtain a clear advantage in terms of F_1-score. In Fig. 6.4a it is possible to observe the F_1-score as a function of the traveled distance. We can notice that the F_1-scores of SBOLSE and SBOLSE with the EBC heuristic outperform the other techniques. This translates directly in an advantage for our techniques in case with a limited budget. If we would have to interrupt the techniques before the convergence, due to a limited battery capacity of the mobile sensor, our techniques would have reached a better classification accuracy. For example, if we stop after 300 meters in the real world dataset SBOLSE would have an F_1-score of 95.29 with a gain of 1.39 with respect to the next best competitor ARS-CIPP_7 that obtains an F_1-score of 93.9.

6.3.2 Synthetic CO_2 dataset experiments

As previously done with the real-world dataset, we determined a parameter setting that allowed a high accuracy with all the algorithms. Result of

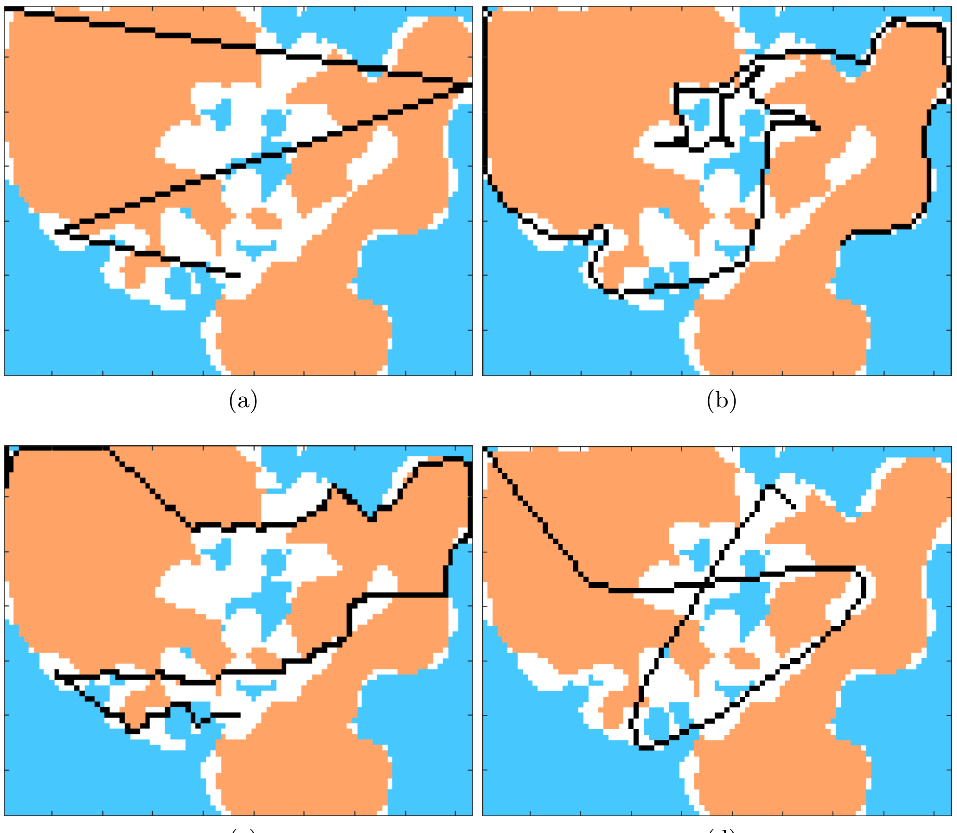


Figure 6.3: Real dataset experiments. The white areas represent location that are still unclassified and black lines display a portion of the path selected by the algorithms: (a) CS, (b) SBOLSE, (c) PULSE and (d) ARS-CIPP. The starting point in the top left corner is the same for all the algorithms.

experiments on the synthetic dataset are shown in Table 6.2. As obtained in the real world dataset, also in the synthetic one the SBOLSE algorithm shows the best performance in term of traveled distance required to obtain a high quality classification. However in this case the advantage is minimal (i.e., 1355.6 meters instead of 1356.4 performed by PULSE$_{b30}$). These results lead to the conclusion that the advantage of the SBOLSE technique is dataset dependent. Moreover, notice that the performance of SBOLSE depends on the orienteering algorithm implemented. As previously stated in Section 6.1.1 this can be substantially improved.

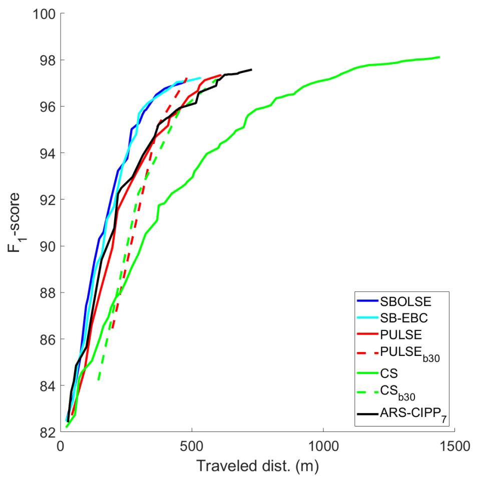
Also in this synthetic dataset the advantage of SBOLSE comes with a prohibitive computation time. The best trade-off is obtained using either the exemplar based clustering heuristic with a minor increase in the path length (+7.7%) but with a substantial reduction of the computation time (-

Table 6.2: F_1-score, total traveled distance (meters) and computation time (seconds) using the synthetic CO_2 dataset, $\overline{x}$ is the average of all 50 experiments and $SE_{\overline{x}}$ is the standard error of the mean.

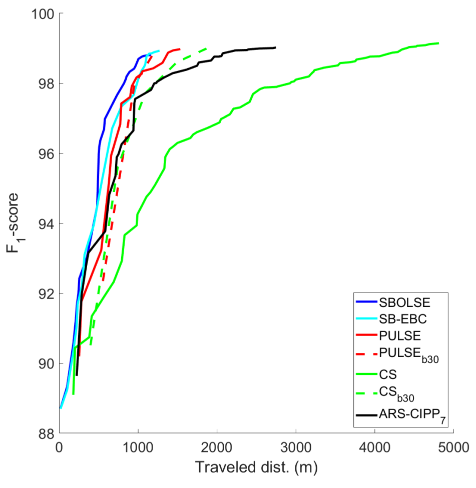
	F_1-score		**Traveled dist. (m)**		**Comp. time (s)**	
	$\overline{x}$	$SE_{\overline{x}}$	$\overline{x}$	$SE_{\overline{x}}$	$\overline{x}$	$SE_{\overline{x}}$
PULSE	98.22	0.090	1709.4	35.37	**23.9**	0.75
$PULSE_{b30}$	98.23	0.092	1356.4	23.08	163.0	4.11
CS	**98.66**	0.071	5588.1	136.86	99.4	2.91
CS_{b30}	98.25	0.089	1782.7	34.05	223.5	5.08
SBOLSE	97.99	0.100	**1355.6**	26.16	3663.8	265.22
SB-EBC	98.03	0.096	1460.8	25.89	168.5	7.95
$ARS\text{-}CIPP_7$	98.25	0.089	2616.0	63.44	192.9	4.75

95.5%), or using the PULSE batch algorithm with a comparable path length and computation time.

Similarly to what explained in section 6.3.1, also by using the synthetic dataset we can notice a clear advantage of SBOLSE and SB-EBC in terms of F_1-score in case of a limited budget. We can notice in Fig. 6.4b that the F_1-scores of our techniques outperform the other, that is, for a smaller budget constraint with SBOLSE and SB-EBC it is possible to obtain a better classification accuracy. For example, if we stop after 1000 meters in the synthetic dataset SBOLSE would have an F_1-score of 98.7 with a gain of 1.15 with respect to the next best competitor $ARS\text{-}CIPP_7$ that obtains an F_1-score of 97.55.



(a)



(b)

Figure 6.4: Evolution of the F_1-score as a function of the distance traveled: (a) on a real-world instance; (b) on a synthetic instance.

6.4 End-point heuristics experiments

In this section we present the empirical evaluation using the orienteering end-point heuristics with respect to the standard SBOLSE algorithm. We performed tests with the four heuristics (EBC heuristic presented in Section 4.3.1 and orienteering end-point baseline heuristics presented in Section 4.3.2) on both the real world dataset and the synthetic datasets previously described in Section 6.2. We identify the different techniques as follows:

- **SBOLSE-EBC:** SBOLSE algorithm with the exemplar based clustering heuristic as described in Section 4.3.1.
- **SB-Random(p):** SBOLSE algorithm with random sparsification heuristic as described in Section 4.3.2.
- **SB-Sparse(p):** SBOLSE algorithm with lowest data point sparsification heuristic as described in Section 4.3.2.
- **SB-Sample(p):** SBOLSE algorithm with discrete random sampling heuristic as described in Section 4.3.2.

In these heuristics, (p) identifies the heuristic parameter (percentage of points). Specifically, we performed tests varying the percentage parameter from 90% down to 10% of the points.

6.4.1 End-point heuristics results

Table 6.3: Average time (seconds) on the real-world and synthetic datasets varying the Random, Sparse and Sample heuristics' parameter

% of points	100	90	80	70	60	50	40	30	20	10
Real dataset										
SBOLSE	1006.2	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a
SBOLSE-EBC	124.4	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a
SB-Random	n/a	858.3	762.1	681.9	582.5	479.8	396.1	306.5	200.3	117.6
SB-Sparse	n/a	857.1	763.4	676.8	576.2	488.8	389.7	301.0	207.6	111.7
SB-Sample	n/a	563.7	533.6	482.8	439.2	390.8	329.2	259.2	192.7	110.4
Synthetic dataset										
SBOLSE	3663.8	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a
SBOLSE-EBC	168.5	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a
SB-Random	n/a	1544.2	1378.8	1225.1	1038.2	880.2	692.7	537.8	389.5	227.7
SB-Sparse	n/a	1555.8	1411.7	1212.3	1047.0	868.8	693.8	537.4	381.0	222,1
SB-Sample	n/a	1004.0	924.9	850.1	768.9	663.1	593.0	472.5	349.5	212.4

We obtained a significant reduction in computation time on the real world dataset. As we can observe on Table 6.3, even with a selection of 90% of the points we obtained a significant reduction on the computation time of roughly 15% with random and sparse. We obtained up to a reduction of roughly 88% when we select only 10% of the points. With the sample heuristic we obtain a reduction of 44% up to 89% with the same parameters. While obtaining a significant reduction on the computation time, we can observe in Table 6.4 that we have a small increase in the total path length required. Specifically, with just 10% of the points selected we obtained an increase in path length of roughly 20-22% with the three random, sparse and sample heuristics. The trend of the F_1-score as a function of the traveled distance on a real world instance is shown in Fig. 6.5. The trend of SBOLSE, SB-EBC, and the three other heuristics is very similar, with a distance traveled that is slightly longer for the heuristics compared to the standard SBOLSE algorithm.

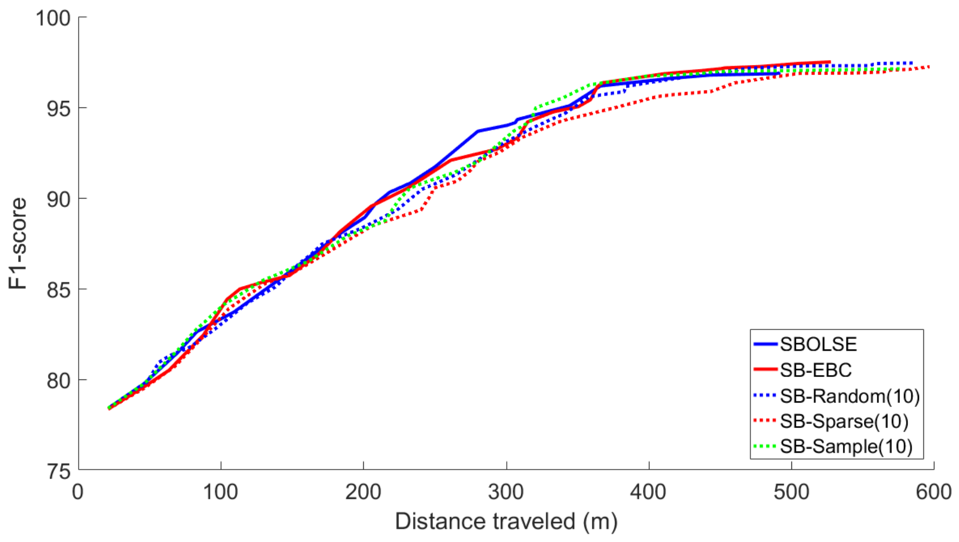


Figure 6.5: Runtime F_1-score comparison on the typical example instance of the real dataset, varying the path length between SBOLSE, SB-EBC, SB-Random, SB-Sparse, and SB-Sample algorithms.

For the synthetic dataset we obtained an even greater reduction in computation time. As we can observe on Table 6.3 with a selection of 90% of the points we obtained a reduction of roughly 58% up to 93% with a selection of 10% of the points. With the sample heuristic we obtained a reduction of 72% up to 94% with the same parameters. As previously discussed for the real-world dataset, the reduction in computation time implies a small increase in the path length (see Table 6.4). Specifically, with only 10% of the points selected we obtained an increase of roughly 21-23% with the three heuristics.

Table 6.4: Average traveled distance (meters) on the real-world and synthetic datasets varying the Random, Sparse and Sample heuristics' parameter

% of points	100	90	80	70	60	50	40	30	20	10
Real dataset										
SBOLSE	473.6	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a
SBOLSE-EBC	495.1	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a
SB-Random	n/a	472.7	486.4	486.6	494.4	494.3	509.9	515.6	530.8	572.7
SB-Sparse	n/a	478.5	473.3	494.0	487.8	503.5	500.5	501.7	529.8	578.3
SB-Sample	n/a	498.6	498.2	501.4	490.0	503.5	513.1	521.1	530.5	569.8
Synthetic dataset										
SBOLSE	1355.6	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a
SBOLSE-EBC	1460.8	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a
SB-Random	n/a	1386.1	1396.4	1407.3	1379.7	1438.5	1451.6	1534.8	1517.7	1645.1
SB-Sparse	n/a	1368.9	1367.6	1381.8	1367.8	1420.0	1425.7	1443.0	1508.4	1650.6
SB-Sample	n/a	1440.2	1401.0	1414.3	1424.3	1469.0	1479.5	1489.9	1546.1	1670.9

6.5 Complete results

For completeness, in Tables 6.5 and 6.6 we report the same results of Tables 6.1 and 6.2 with the additional heuristics presented in Section 4.3.2. Regarding the SB-Random, SB-Sparse and SB-Sample we report results with parameter $p = 10$ because it represents a good trade-off between time and path length.

Moreover, in the following tables we also present the complete experiments performed with the technique recently proposed by Hitz et al. in [87]. As mentioned before, in this technique authors use an adaptive re-planning algorithm that optimizes a B-spline by varying n control points. In our experiments with this technique we have varied the number n of control points, in particular using $n \in \{3, 7, 11, 15, 19\}$.

6.6 Conclusions

In this part of the thesis we proposed a novel set of algorithms for a specific environmental monitoring application called the level set estimation problem. Our algorithms are specifically designed for continuous-measuring mobile sensors where the cost to perform a measurement is negligible. In this context we aim at optimizing the total path length required from the platform and reduce time required to compute an informative path. As the results of our empirical evaluation presented in this chapter shows, the different variants of our techniques are able to obtain a high quality classification with a shorter path and a lower computation time compared to state-of-the-art algorithms in literature for the level set estimation problem.

Specifically, the SBOLSE algorithm presented in Chapter 4 implements an orienteering heuristic solution as a subroutine to select an informative path that meets a given travel budget. Results show that our approach

Table 6.5: F_1-score, total traveled distance (meters) and computation time (seconds) using the real world pH dataset. $\overline{x}$ is the average of all 30 experiments and $SE_{\overline{x}}$ is the standard error of the mean.

	F_1-score		**Traveled dist. (m)**		**Comp. time (s)**	
	$\overline{x}$	$SE_{\overline{x}}$	$\overline{x}$	$SE_{\overline{x}}$	$\overline{x}$	$SE_{\overline{x}}$
PULSE	97.46	0.063	587.8	10.82	**11.1**	0.27
PULSE_{b30}	97.43	0.060	518.7	6.68	63.5	0.89
CS	**98.22**	0.039	1560.8	18.58	38.1	0.49
CS_{b30}	97.47	0.055	671.7	13.71	82.4	1.74
SBOLSE	97.23	0.066	**473.6**	6.20	1006.2	45.99
SB-EBC	97.25	0.064	495.1	8.08	124.4	4.19
SB-Random(10)	97.35	0.057	572.7	10.16	117.6	5.07
SB-Sparse(10)	97.36	0.068	578.3	11.45	111.7	4.32
SB-Sample(10)	97.32	0.066	569.8	9.57	110.4	3.95
ARS-CIPP_3	97.60	0.058	899.2	13.44	147.7	2.57
ARS-CIPP_7	97.57	0.045	736.1	10.30	114.5	1.70
ARS-CIPP_{11}	97.64	0.054	812.9	11.95	150.4	2.69
ARS-CIPP_{15}	97.68	0.050	895.8	15.36	187.4	4.18
ARS-CIPP_{19}	97.75	0.056	962.1	11.05	227.7	3.70

significantly outperforms the state-of-the-art algorithms for the level set estimation problem in terms of total travel distance, while maintaining a near-optimal classification quality.

On the other hand the PULSE algorithm presented in Chapter 5 reduces the computation time compared to the other techniques, while remaining competitive in terms of distance traveled by the mobile sensor and the classification accuracy obtained.

In general, for applications where the mobile sensor is equipped with sufficient computation power and the optimization of the traveled distance is crucial (e.g. limited battery capacity) it is preferable to use the SBOLSE algorithm. Instead, for application where the computation power is limited PULSE constitutes the preferable choice. Overall SB-EBC (i.e SBOLSE with exemplar based clustering heuristic) represents a good trade-off for most applications.

Table 6.6: F_1-score, total traveled distance (meters) and computation time (seconds) using the synthetic CO_2 dataset, $\overline{x}$ is the average of all 50 experiments and $SE_{\overline{x}}$ is the standard error of the mean.

	F_1-score		**Traveled dist. (m)**		**Comp. time (s)**	
	$\overline{x}$	$SE_{\overline{x}}$	$\overline{x}$	$SE_{\overline{x}}$	$\overline{x}$	$SE_{\overline{x}}$
PULSE	98.22	0.090	1709.4	35.37	**23.9**	0.75
PULSE$_{b30}$	98.23	0.092	1356.4	23.08	163.0	4.11
CS	**98.66**	0.071	5588.1	136.86	99.4	2.91
CS$_{b30}$	98.25	0.089	1782.7	34.05	223.5	5.08
SBOLSE	97.99	0.100	**1355.6**	26.16	3663.8	265.22
SB-EBC	98.03	0.096	1460.8	25.89	168.5	7.95
SB-Random(10)	98.05	0.094	1645.1	32.11	227.7	14.28
SB-Sparse(10)	98.05	0.095	1650.6	34.22	222.1	14.98
SB-Sample(10)	98.04	0.093	1670.9	41.70	212.4	13.47
ARS-CIPP$_3$	98.23	0.082	3059.0	75.11	323.7	9.96
ARS-CIPP$_7$	98.25	0.089	2616.0	63.44	192.9	4.75
ARS-CIPP$_{11}$	98.28	0.089	2777.4	62.30	218.2	5.02
ARS-CIPP$_{15}$	98.32	0.082	3052.7	61.33	265.0	5.89
ARS-CIPP$_{19}$	98.39	0.075	3433.2	74.52	319.6	7.43

Part III

Optimizing sampling locations

Chapter 7

Gradient descent for Gaussian processes variance reduction

7.1 Introduction

As we are describing thorough the thesis, in many analyses we are dealing with a spatial phenomena of interest that is modeled using Gaussian processes [149]. When tackling the analysis of a spatial phenomena in a data-driven manner, a key issue, before setting up the actual model, is to decide on the locations where measurements are going to be taken. The better our choice of locations, the better the Gaussian process will approximate the true underlying functional relationship or the fewer measurements we need to build a model that provides a prespecified level of performance.

In contrast to what we presented in the previous part of the thesis instead of optimizing paths for mobile sensors here we tackle the problem of optimizing a given set of sensing locations. One example where this is a common problem to solve is of course environmental monitoring. In environmental monitoring it is necessary to choose a set of locations in space in which to measure the specific phenomenon of interest (e.g. the temperature of the environment or the pH value of water in rivers or in lakes [88, 160]) or similarly it is necessary to chose the displacement positions of fixed sensors [80, 105, 108]. In both cases however, the process is usually costly and one wants to select observations that are especially informative with respect to some objective function.

Recent research in the context of optimizing sampling locations has focused on selecting a set of measurement so as to minimize the posterior variance of the Gaussian process [107]. This selection of measurement locations is basically performed through the use of greedy procedures. In particular, submodularity is often exploited [103, 105, 145]. Submodularity is an intuitive diminishing returns property (as previously described in Section 3.3), and the idea behind this technique is that adding a new measurement to a small set helps more than adding it to a large set of measurements.

Although submodular objective functions allows for a greedy optimization with bound guarantees [129], the solution of this technique is inherently discrete since we can select the sensing points between a given set of possible locations. Hence, the solution of such a technique can deviate considerably from the optimum in cases where the sensing location can be arbitrarily adapted in a continuous domain of interest. In other words, there is definitely room for improvement, which is the main goal of the work proposed in this chapter.

Specifically, in what follows we propose a Gradient descent (GD) procedure to minimize the posterior variance of the Gaussian process by adapting sampling locations and we evaluate its performance. Essentially, we use a gradient descent algorithm to adapt the sensing locations starting from a set of initial positions that can be given from another algorithm. The proposed study falls within the context of continuous optimization motivated by the fact that in some cases, such as some environmental monitoring applications, the locations where measurements are performed does not have to be confined in predetermined set of points whereas approaching this problem by exploiting submodularity requires a discretization of the space.

More in detail, the main contributions of this chapter are:

- We provide a Gradient descent technique that minimizes the posterior variance of a Gaussian process.
- We perform an extensive empirical evaluation of the procedure under different conditions by varying:
  - The hyperparameters of the Gaussian process.
  - The number of sampling points K.
  - The method of initialization of the points.
  - The dimensionality of the dataset to prove the validity of the technique in different contexts besides environmental monitoring applications.
- We present the results and discuss the applicability and the improvements that our technique offers under the different settings we have tested. In particular, we show how submodular greedy solutions can be further improved.

7.1.1 Problem definition

Given a continuous domain $\mathcal{D} \subset \mathbb{R}^d$ that represents a region of the environment, we want to select a set of K points where to perform measurements in order to minimize the total posterior variance of the Gaussian process that we are using to model the phenomena in that domain.

Specifically, we want to select a set of K measurements taken at locations $\mathcal{K} = \{\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \cdots, \boldsymbol{\mu}_K\} \in \mathcal{D}$ such that we minimize the following objective function:

$$J(\{\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \cdots, \boldsymbol{\mu}_K\}) = \sum_{\mathbf{x}_i \in \mathcal{X}} \sigma^2(\mathbf{x}_i) \tag{7.1}$$

where $\sigma^2(\mathbf{x}_i)$ is the Gaussian process variance computed in point $\mathbf{x}_i$ as defined by equation 3.7 and $\mathcal{X}$ is a set of points where we test the Gaussian process. For example, in a typical environmental monitoring application we are interested in generating a model of the phenomena of interest in a specific region $\mathcal{D}$ of the environment and we generate this model by testing the Gaussian process in a matrix of uniformly distributed point in space. Notice that the variance computed with equation 3.7 in locations $\mathbf{x}_i \in \mathcal{X}$ is dependent on the set of sampling points $\mathcal{K}$ as explained in Section 3.1.1.

$J(\{\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \cdots, \boldsymbol{\mu}_K\})$ is a multi-dimensional function that represents the total posterior variance of the Gaussian process. Specifically, given a set of K measurements and a domain $\mathcal{D}$ in d dimensions, the objective function J is Kd multi-dimensional.

7.2 Implementation

Rather than exploiting the submodularity property of the objective function in equation 7.1 to come to a greedy subset selection in a discrete domain, we decide to rely on Gradient descent to optimize the sensing locations in a continuous manner. Specifically, starting from an initial configuration of measurement points in the domain, we perform a Gradient descent procedure to minimize the total posterior variance of the Gaussian process.

The main idea behind our algorithm is to exploit the gradient of the objective function in equation 7.1 to iteratively re-adapt the locations of the measurement points across the domain. As mentioned before, the value of the multi-dimensional objective function $J(\{\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \cdots, \boldsymbol{\mu}_K\})$ represents the total posterior variance of the Gaussian process given the K observations in a d-dimensional space. Following the gradient of the objective function corresponds to a simultaneous update of all the measurement points in the domain space. Notice that, submodular greedy approach does not consider all these points simultaneously and this is what gives our technique an edge over that approach. To facilitate reading we present the calculation of the gradient of the objective function separately in Appendix A.

In the direction of the negative gradient we have, in principle, a better solution and in our algorithm we take all the necessary precautions to avoid that the iterative step produces a displacement that would lead to a worse solution. With this, at every iteration the algorithm is guaranteed to obtain an improvement. The full pseudo-code of the procedure is listed in Algorithm 1.

Algorithm 1 Gradient descent procedure
input: set of initial sampling locations $\mathcal{K}^0 = \{\boldsymbol{\mu}_1^0, \boldsymbol{\mu}_2^0, \dots, \boldsymbol{\mu}_K^0\}$, set $\mathcal{X}$, convergence factor cf

```
1:  $i \leftarrow 0$
2:  $step \leftarrow 0$
3:  compute $maxD$
4:  $converged \leftarrow false$
5:  compute GP given $\mathcal{K}^0$
6:  compute $J(\mathcal{K}^0)$
7:  while !converged do
8:    $i \leftarrow i + 1$
9:    $step \leftarrow step + 1$
10:   compute $\nabla J(\mathcal{K}^{i-1})$
11:   $improved \leftarrow false$
12:   while !improved && !converged do
13:     for all $\boldsymbol{\mu} \in \mathcal{K}^{i-1}$ do
14:       $displacement(\boldsymbol{\mu}) \leftarrow \boldsymbol{\mu}^{i-1} - \nabla J(\boldsymbol{\mu}^{i-1})/step$
15:       $\boldsymbol{\mu}^i \leftarrow \boldsymbol{\mu}^{i-1} + displacement(\boldsymbol{\mu})$
16:     end for
17:     compute GP given $\mathcal{K}^i$
18:     compute $J(\mathcal{K}^i)$
19:     if $J(\mathcal{K}^i) < J(\mathcal{K}^{i-1})$ then
20:       $improved \leftarrow true$
21:     else
22:       $step \leftarrow step + 1$
23:       $\mathcal{K}^i \leftarrow \mathcal{K}^{i-1}$
24:     end if
25:     $converged \leftarrow true$
26:     for all $\boldsymbol{\mu} \in \mathcal{K}^i$ do
27:       if $|displacement(\boldsymbol{\mu})| > cf \cdot maxD$ then
28:         $converged \leftarrow false$
29:       end if
30:     end for
31:   end while
32: end while
33: return $\mathcal{K}^i$
```

Let us go through the procedure, starting out by describing the inputs and output that it considers. One of the inputs is the set of initial sampling points $\mathcal{K}^0$ that can be initialized using different choices. For example they can be chosen randomly or through use of a different techniques, a detailed description regarding our choices can be found in the experimental phase in Section 7.3. The input $\mathcal{X}$ represent the set of locations where we want to evaluate our Gaussian process in order to compute the posterior variance using equation 3.7. The remaining input (cf) is used to determine the convergence of the procedure and its use will be clearer in the following description. The output of the procedure is represented by the final set $\mathcal{K}^i$ of sampling locations after i iterations of the algorithm.

The procedure begins by initializing the required variables (lines 1-4) to manage the main loop and by computing the total posterior variance given the initial set of sampling locations $\mathcal{K}^0$ (lines 5-6). The main loop (lines 7-32) iterates until the convergence is reached and it is made up of three main components: i) the computation of the gradient of our objective function (line 10); ii) the gradient descent iterative step that allows to minimize the objective function (lines 11-31), that we better describe in Section 7.2.1; iii) the check of convergence (lines 25-30) whose function is described in section 7.2.2.

7.2.1 Gradient descent iterative step

Here we describe in details the iterative step (lines 11-31) that allows our procedure to minimize the objective function. We have previously computed (line 10) the derivative of the objective function given the current position of the points in our set $\mathcal{K}^{i-1}$. The iterative step computes for all the current measurement locations $\boldsymbol{\mu} \in \mathcal{K}^{i-1}$ (lines 13-16) what is the displacement given the derivative. However, as any gradient descent procedure, we have to keep into account situations where the iterative step would "jump" over the current basin of attraction.

As noted earlier, in the direction of the negative gradient the objective function is decreasing in value and we want to guarantee that our algorithm at every iteration improves the solution. A simple method is to check whether the current step would make us improve the current solution or not. To this aim we recompute the value of the objective function (lines 17-18) and verify that this corresponds to a net improvement with respect to the previous configuration (lines 19-20). Otherwise we roll-back to the previous configuration (line 23) and recompute a smaller displacement. To this aim we make use of the additional variable *step*. We can observe that this variable is used to compute the amplitude of the displacement of a point $\boldsymbol{\mu}$ (in line 14) with the formula $\boldsymbol{\mu}^{i-1} - \nabla J(\boldsymbol{\mu}^{i-1})/step$.

The *step* is increased at each iteration of the algorithm at least once (in line 9) to guarantee a slowdown, and an additional number of times (line 22) to guarantee that at each iteration we obtain an improvement (i.e. we minimize our objective function).

7.2.2 Convergence

As shown in Algorithm 1, as part of the inputs we have cf which is used to determine the convergence of the algorithm. This parameter is intended as a threshold to determine whether the procedure has to terminate or not. cf specifies what is the lowest percentage (with respect to the dataset diameter) of displacement that any points we are adapting can move. At the beginning of the procedure we compute the diameter of the dataset $maxD$ (line 3). Later, inside the main loop of the procedure we check the convergence (lines 25-30). When all the points in $\mathcal{K}^i$ received a displacement that is lower than $cf \cdot maxD$ we consider the procedure terminated.

The cf parameter acts as a trade-off between the precision of the solution and the computation (number of iterations) required to converge. For small values the algorithm is allowed to go through its iterations as long as at least one of the points in space is moving by a small amount. Larger values will make the procedure stop earlier with a solution that may of course be further from an optimum than when small values are used.

7.3 Empirical evaluation

Since we want to test the performance of our procedure under different conditions we generated datasets with domains from 1 up to 5 dimensions. Specifically, we have generated cubic datasets with equally distributed domain points $\mathcal{X}$. The cardinality of the domain $|\mathcal{X}|$, that is the number of points on which we evaluate the Gaussian process, has been adapted to be at least 1000 points. The two dimensional dataset is simply a set of equally distributed points on a grid, while the three dimensional dataset is a set of equally distributed points on a cube, etc. (see Figure 7.1). Notice that in particular the two and three dimensional datasets are particularly suited to represent a typical instance in case of environmental monitoring application. In fact, usually in environmental analysis we are interested in modeling the property of a phenomena in a specific sector of the environment.

One of the most widely used kernel in many contexts even besides spatial phenomena modeling, is the Gaussian one, also known as squared exponential kernel:

$$k(\mathbf{x}, \mathbf{x}') = \sigma_f^2 \exp\left(-\frac{(\mathbf{x}-\mathbf{x}')^T(\mathbf{x}-\mathbf{x}')}{2l^2} \right) \tag{7.2}$$

which is therefore the obvious choice in our experiments. The hyperparameters of the kernel can vary considerably however. Hence, to generally study the performance of our gradient descent procedure we varied these in our experiments. Specifically we used 20 different length-scale l and 15 different σ_f. The former describes the smoothness property of the true underlying function while the latter the standard deviation of the modeled function. As

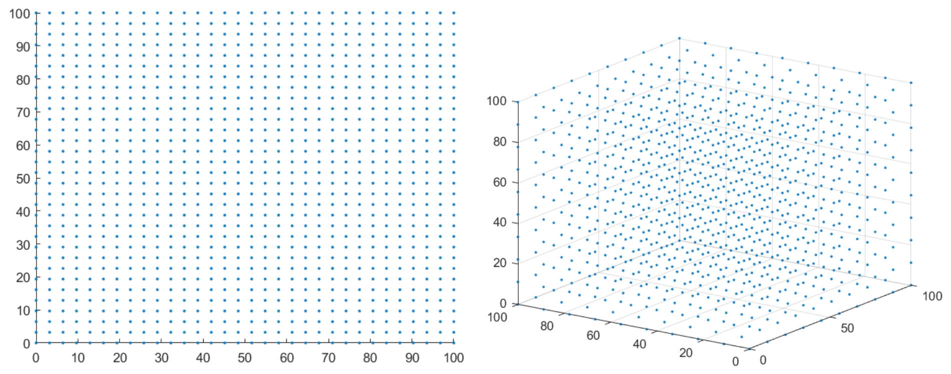


Figure 7.1: Left: 2-d dataset (1024 points). Right: 3-d dataset (1000 points)

we can observe in equation 3.7 these are fundamental to determine the variance of the Gaussian process. Moreover, as mentioned in the background Section 3.1.1 we assume that predictions are made using noisy observations, hence in our experimental phase we also used 10 different σ_n.

In addition to the different number of dimensions of the datasets and the hyperparameters previously described, we have tested the procedure by adapting a different number K of measurement points which varies from 2 up to 7. The number of sampling points is reasonable for applications with datasets of comparable size. The case of a single point has been excluded since the submodular greedy technique is optimal by definition.

Some starting locations of the sampling points are required to initialize our gradient descent algorithm. Here we initialized them using the submodular greedy procedure in order to measure the magnitude of the possible improvements and to see under what conditions we can obtain them. The additional input of the procedure as described in Section 7.2.2 is $cf = 1/1000$.

To summarize, by considering the different hyperparameters, dimensionality of the datasets and number of measurement points, we have performed 90,000 different experiments that allows us to characterize and study the improvement obtainable with the gradient descent procedure with respect to the widely used submodular greedy technique.

Moreover, we have also performed the 90,000 experiments by initializing the points randomly instead of using a submodular solution, this allows us to study the average improvement obtainable without the needs to previously perform a different algorithm. In addition we have selected a subset of the hyperparameters and datasets to perform a test with many different random initializations on the same instances.

The results and study of the experiments are described in the next section.

7.4 Results

We describe the results from different points of view and comment on the applicability of the technique we proposed. Specifically:

- We present the results of the Gradient descent procedure as a function of the hyperparameters of the Gaussian process (section 7.4.1).
- We discuss the results and applicability of the technique varying the number of sampling points and the dimensionality of the domain (section 7.4.2).
- We speculate about the initialization of the sampling points by comparing the results of Gradient descent staring from a configuration of points as a submodular greedy procedure would select and a random initialization (section 7.4.3).

7.4.1 Results as a function of the hyperparameters

To explain the performance of gradient descent as a function of the hyperparameters of the Gaussian process, we take as example the two plots in Figure 7.2. In these pictures we can observe the % of improvement that gradient descent obtains with respect to the submodular solution by varying the hyperparameters in the two dimensional dataset when using 5 sampling points: the vertical axis reports the length-scale l of the kernel and the horizontal axis reports the standard deviation σ_f of the function. The two pictures represent these improvements by fixing a single standard deviation of the noise measurement σ_n; the one to the right with a σ_n that is almost three times the one to the left.

Based on these we can make the following observations:

- Independently of σ_f and σ_n when we use very small length-scales (top rows of the two pictures in Figure 7.2) the advantage we can obtain with gradient descent is very low. The reason why this happens is that with small length-scales the contribution in variance reduction given by an observations is mostly concentrated in a very narrow position. To observe this phenomena take as an example Figure 7.3a. Consider that we are trying to estimate where to make two observations, as long as they are a little separated one another we are already obtaining most of the variance reduction possible. With very small length-scale the position where we make observations influences little to nothing the final amount of posterior variance. Hence with gradient descent in these cases we cannot obtain an advantage with respect to the submodular greedy technique.

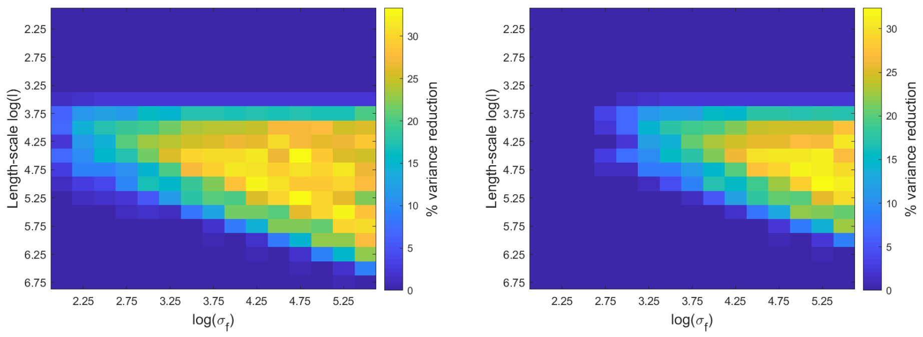


Figure 7.2: Results as a function of the hyperparameters. Horizontally are variations in the standard deviation σ_f and on the vertical axis in the length-scale l. Colors represent the % of variance reduction of GD relative to the submodular greedy solution. These results refer to the adaptation of 5 points in the 2-dimensional dataset for a fixed σ_n. Specifically in the right image σ_n is about three times higher then in the left one.

- When the length-scale of the kernel becomes bigger (as an example Figures 7.3b and 7.3c) the reduction in variance given by a measurement point has an effect on a larger portion of the domain, hence the location where the measurements are taken affect the total amount of posterior variance reduction. In this case we observe that the locations selected by the gradient descent procedure obtain an advantage with respect to the submodular greedy technique.

- When the length-scale becomes bigger we notice that the σ_f and σ_n parameters affect the results differently. Consider, for instance, the left picture in Figure 7.2. We can observe that for small values of σ_f we obtain a small advantage and vice versa. These results are shifted to the right when the σ_n parameter increases (right picture in Figure 7.2). This show that the ratio σ_f/σ_n affects the quality of the results: the higher the ratio the higher the improvements we can obtain.

7.4.2 Varying the number of points and dimensionality

In this section we study the performance of gradient descent with respect to the submodular greedy solution by varying the number of sampling points K and the number of dimensions of the domain. In tables 7.1 and 7.2 we report the percentage of variance reduction that the gradient descent procedure allows to obtain with respect to the total posterior variance of the Gaussian process when the measurement locations are selected by the submodular greedy technique. Specifically, each entry of the table reflects

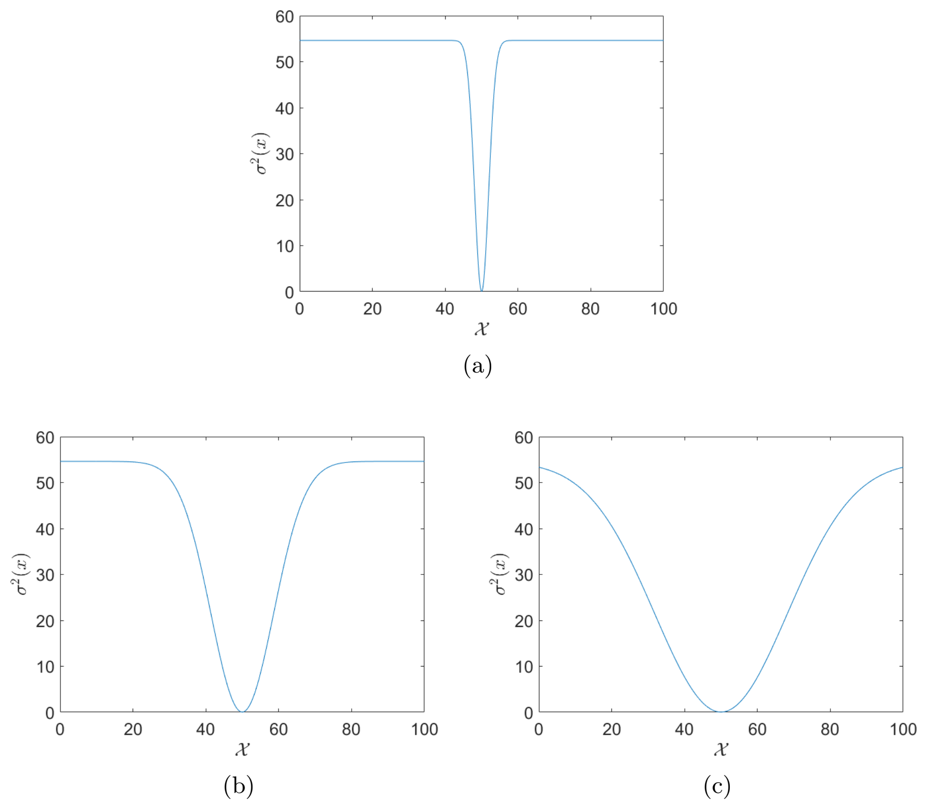


Figure 7.3: Plots of the variance of a Gaussian process with a single sampling point in the middle of the domain using the squared exponential kernel with three different length-scales.

the improvement obtained for a specific combination of number of points and dimensionality of the domain.

Table 7.1 represents the average % gain of gradient descent with respect to the submodular greedy solution. Specifically each entry represents the average over all the 3000 hyperparameters for a specific combination of dimensionality of the domain and number of measurement points. As we can observe, in general the gradient descent procedure allows us to improve significantly the sobmodular solutions for small dimensionality and number of points.

Table 7.2 instead represent the maximum improvement that gradient descent can obtain with respect to the submodular greedy solution on a specific combination of dimensionality of the domain and number of measurement points. Specifically the improvement reported in each entry of the table is the maximum value encountered between all the possible 3000 combination of hyperparameters for the specific number of sampling points K and dimensionality of the domain. Also in this case we can observe that

Table 7.1: Average % gain of gradient descent with respect to the submodular greedy solution.

	Number of points (K)					
	2	3	4	5	6	7
1-D	32.83	18.23	17.60	17.05	14.76	8.50
2-D	4.14	16.93	19.68	9.16	13.66	14.48
3-D	1.03	2.83	8.79	8.00	10.55	8.20
4-D	0.32	0.97	1.94	5.06	3.46	4.91
5-D	0.01	0.60	1.07	1.67	3.89	2.20

Table 7.2: Maximum % gain of gradient descent with respect to the submodular greedy solution.

	Number of points (K)					
	2	3	4	5	6	7
1-D	59.91	86.77	89.80	89.15	71.57	71.72
2-D	21.11	60.27	54.91	33.36	76.67	72.27
3-D	6.23	15.84	52.09	29.91	41.23	31.00
4-D	6.59	11.48	12.17	31.11	20.73	22.57
5-D	2.99	8.80	8.23	17.54	40.09	22.64

in general gradient descent produces better results for small dimensionality and number of points. For a graphical representation of the execution of the gradient descent procedure on an example instance see Figure 8.3.

7.4.3 Random initialization

Here we report the results similarly to the previous section. In this case the gradient descent procedure has been initialized with points in randomly selected locations across the domain.

Table 7.3: Average % gain of gradient descent with respect to a single random configuration.

	Number of points (K)					
	2	3	4	5	6	7
1-D	38.81	44.97	45.57	46.62	47.14	46.59
2-D	19.69	34.98	36.39	35.80	36.96	38.59
3-D	18.32	17.95	32.27	30.13	30.87	30.73
4-D	17.16	14.61	16.92	30.32	27.37	25.94
5-D	15.90	13.43	12.92	15.92	28.02	25.11

Table 7.3 represents the average and table 7.4 the maximum improvement of gradient descent with respect to the random initial collocation of

Table 7.4: Maximum % gain of gradient descent with respect to a single random configuration.

	Number of points (K)					
	2	3	4	5	6	7
1-D	99.38	99.33	99.59	99.78	99.74	99.59
2-D	78.28	99.09	97.36	96.85	94.41	96.48
3-D	69.95	81.10	98.44	96.63	94.12	88.85
4-D	62.85	66.11	76.19	96.71	94.23	94.37
5-D	59.92	58.77	62.33	75.27	95.58	97.10

points. These results represent the gain in terms of percentage of variance reduction with respect to the variance of the Gaussian process with the measurement points in random locations. As we can observe, since the random collocation of point can represent a very bad quality solution compared to the submodular greedy procedure, results show much bigger improvements.

A more interesting point of view is offered in table 7.5. Here we compare the total posterior variance of the Gaussian process after the gradient descent adaptation from a random initialization with the total posterior variance after the gradient descent adaptation starting from the submodular greedy solution.

Table 7.5: Maximum % gain of gradient descent starting from a random configuration with respect to gradient descent starting from the submodular greedy solution.

	Number of points (K)					
	2	3	4	5	6	7
1-D	43.37	75.97	74.02	39.10	53.18	36.90
2-D	14.15	34.63	31.88	35.27	52.05	52.13
3-D	9.68	15.80	30.15	16.44	35.86	21.94
4-D	4.90	7.74	14.09	26.64	15.33	15.28
5-D	1.23	7.00	7.03	7.20	26.68	21.44

Specifically, table 7.5 reports the maximum improvements that has been encountered by varying the 3000 hyperparameters. Although, the result can vary considerably across the hyperparameters, results show that from a random initialization of points we can obtain in some cases better results than using a submodular greedy procedure to select the starting configuration.

Notice that the aforementioned tables (7.3, 7.4 and 7.5) report results considering a single random initialization per instance. Since the selection of the initial measurement points is subject to a great variance we also performed a more detailed test on a small subset of cases. Specifically, we have selected the two dimensional dataset and we use gradient descent to adapt the location of two points and the three dimensional dataset with six points. By fixing also a specific σ_n parameter, we performed experiments by using 100 randomly initialization for each of the 300 combinations of σ_f and l. Results are presented in Figure 7.4. As we can observe, when we perform multiple randomly initialized executions on average we obtain a spectrum of improvements similar as what shown in previous Figure 7.2.

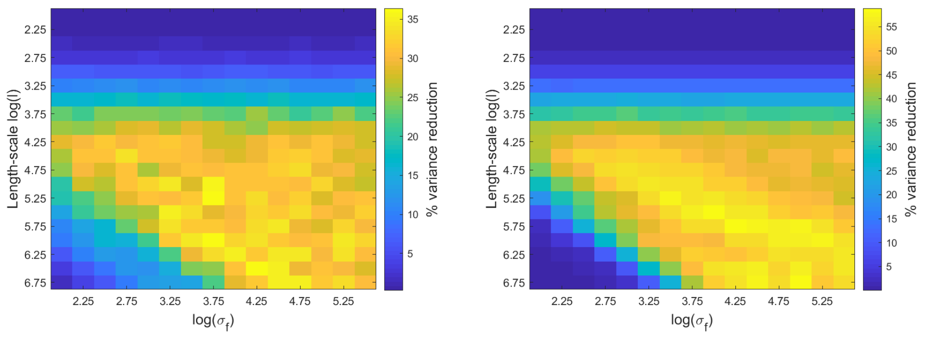


Figure 7.4: Average gain over 100 randomly initialized execution of gradient descent. Left with 2 points in the 2-dimensional dataset and right 6 points in the 3-dimensional dataset.

7.5 Conclusions

In this chapter we proposed a Gradient descent procedure to minimize the posterior variance of a Gaussian process and the performance of the technique has been analyzed under different settings in order to show the generality of the approach. Results show that in many cases it is possible to obtain a significant improvement with respect to a random initialization or the well-known submodular greedy procedure. Although with a random initialization the performance can vary considerably, results show that in some cases it is possible to obtain better solutions than with a submodular greedy initialization.

The proposed study falls within the context of continuous optimization motivated by the fact that in some applications, such as environmental monitoring, the locations where measurements are performed does not have to be confined in predetermined points in space, but rather the domain is continuous. Approaching this context by exploiting submodularity requires a discretization of the space. On the other hand Gradient descent does not requires the domain to be discrete and it can iteratively improve the solution by freely move the measurement points in a continuous manner. A natural evolution of these contributions is represented by the use of different kernels (e.g. anisotropic kernels), allowing to study and define the properties and the performances in different scenarios.

Chapter 8

Heuristic for Gaussian processes variance reduction

8.1 Introduction

Similarly to what discussed in Chapter 7, here we present an additional contribution in the context of optimizing a set of sampling locations in order to minimize the Gaussian process posterior variance.

As previously described, the selection of measurement locations is usually performed through the use of greedy procedures (such as submodular optimization) that operates in a discrete space. Specifically they optimize the sensing locations by selecting them from a given set of feasible points. We showed by proposing a gradient descent procedure that the variance reduction can be widely improved if we can optimize the sensing locations in domain of interest that is continuous and not discrete.

Although the contribution proposed in the previous chapter allows us to study how the sensing selection can be improved in a continuous setting under different conditions (hyperparameters of the Gaussian process, dimensionality of the domain and number of sampling locations), as any gradient descent procedure the selection of a parameter or the implementation of some mechanism that controls the learning rate is required as we described in Section 7.2.1. Moreover, the convergence of a gradient descent procedure usually requires a significant computational effort and time.

For this reason, in this chapter we propose a novel iterative heuristic approach to reduce the posterior variance of a Gaussian process but specifically designed to solves the issues described above. Our technique does not require the tuning of any parameter to determine how the sensing locations evolve at each iteration (learning rate) and represents a trade-off between the quality of solutions and the associated computation time.

More in detail, our contributions presented in this chapter of the thesis can be summarize as follows:

- We propose an iterative heuristic approach to optimize sensing locations by minimizing the Gaussian process posterior variance.
- We prove the approximation of the heuristic with respect to the real objective function.
- We perform an empirical evaluation on the same datasets presented in the previous chapter, showing the performance of the technique in terms of gain in variance reduction and in computation time. Also in this case we performed our tests by varying:
  - The hyperparameters of the Gaussian process.
  - The number of sampling points K.
  - The method of initialization of the points.
  - The dimensionality of the dataset to prove the validity of the technique in different contexts besides environmental monitoring applications.

8.1.1 Problem definition

The problem definition that we report here for convenience is the same as presented in Section 7.1.1.

Given a continuous domain $\mathcal{D} \subset \mathbb{R}^d$ that represents a region of the environment, we want to select a set of K points where to perform measurements in order to minimize the total posterior variance of the Gaussian process that we are using to model the phenomena in that domain.

Specifically, we want to select a set of K measurements taken at locations $\mathcal{K} = \{\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \cdots, \boldsymbol{\mu}_K\} \in \mathcal{D}$ such that we minimize the following objective function:

$$J(\{\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \cdots, \boldsymbol{\mu}_K\}) = \sum_{\mathbf{x}_i \in \mathcal{X}} \sigma^2(\mathbf{x}_i) \tag{8.1}$$

where $\sigma^2(\mathbf{x}_i)$ is the Gaussian process variance computed in point $\mathbf{x}_i$ as defined by equation 3.7 and $\mathcal{X}$ is a set of points where we test the Gaussian process. For example, in a typical environmental monitoring application we are interested in generating a model of the phenomena of interest in a specific region $\mathcal{D}$ of the environment and we generate this model by testing the Gaussian process in a matrix of uniformly distributed point in space. Notice that the variance computed with equation 3.7 in locations $\mathbf{x}_i \in \mathcal{X}$ is dependent on the set of sampling points $\mathcal{K}$ as explained in Section 3.1.1.

$J(\{\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \cdots, \boldsymbol{\mu}_K\})$ is a multi-dimensional function that represents the total posterior variance of the Gaussian process. Specifically, given a set of K measurements and a domain $\mathcal{D}$ in d dimensions, the objective function J is Kd multi-dimensional.

8.2 Implementation

Rather than relying on a gradient descent procedure to optimize equation 8.1, here we propose a technique that is inspired by the Lloyd's algorithm for K-means [116]. Specifically, starting from an initial configuration of measurement points in the domain, we perform an iterative procedure to minimize the total posterior variance of the Gaussian process.

The main idea behind our heuristic is to represent the sampling locations as the centroid of a procedure inspired by soft K-means clustering and to iteratively re-adapt the locations of the measurements points across the domain.

The basic notions about soft K-means have been previously described in Section 3.6.1. Here we start by describing the classical Lloyd's procedure for K-means re-adapted to the case of soft clustering [13]. Its pseudo code is reported in Algorithm 2.

Algorithm 2 Pseudo code for soft K-means clustering
input: set of K randomly initialized centroids $\{\boldsymbol{\mu}_1^0, \boldsymbol{\mu}_2^0, \ldots, \boldsymbol{\mu}_K^0\}$

```
1: $t \leftarrow 0$
2: repeat
3:   // compute all indicator variables
4:   for $i = 1$ to $K$ do
5:     for $j = 1$ to $|\mathcal{X}|$ do
6:       $z_{i,j} = \frac{e^{-\beta||\mathbf{x}_j - \boldsymbol{\mu}_i^t||^2}}{\sum_{l=1}^{K} e^{-\beta||\mathbf{x}_j - \boldsymbol{\mu}_l^t||^2}}$
7:     end for
8:   end for
9:   // update all centroids
10:  for $i = 1$ to $K$ do
11:    $\boldsymbol{\mu}_i^{t+1} = \frac{\sum_j z_{i,j}\mathbf{x}_j}{\sum_j z_{i,j}}$
12:  end for
13:  $t \leftarrow t + 1$
14: until centroids stabilize
```

As we can observe in Algorithm 2 the procedure iteratively computes the indicator variables $z_{i,j}$ (which represent the degree of belonging of point j to the cluster i) and subsequently updates the centroids of the clusters.

The heuristic that we propose is similar in spirit to this soft K-means procedure since we also iteratively compute a set of indicator variables and subsequently the new position of centroids. In our case however, a 'centroid' $\boldsymbol{\mu}$ represents a sampling location, hence the indicator variable $z_{i,j}$ will represent the degree of which the variance of point $\mathbf{x}_j$ is reduced by sampling in position $\boldsymbol{\mu}_j$.

The pseudo code of our technique is listed in Algorithm 3.

Algorithm 3 Pseudo code of our heuristic
input: set of K initial sampling locations $\mathcal{K}^0 = \{\boldsymbol{\mu}_1^0, \boldsymbol{\mu}_2^0, \ldots, \boldsymbol{\mu}_K^0\}$, convergence criteria

```
1: $t \leftarrow 0$
2: **repeat**
3:   **for** $i = 1$ **to** $K$ **do**
4:     // compute GP variance given sampling locations $\mathcal{K}^t \setminus \boldsymbol{\mu}_i^t$
5:     $\Gamma = GP|_{\mathcal{K}^t \setminus \boldsymbol{\mu}_i^t}$
6:     // compute indicator variables
7:     **for** $j = 1$ **to** $|\mathcal{X}|$ **do**
8:       $z_{i,j} = \Gamma(\mathbf{x}_j) \frac{\sigma_f^2}{\sigma_f^2 + \sigma_n^2} e^{-\frac{||\mathbf{x}_j - \boldsymbol{\mu}_i^t||^2}{l^2}}$
9:     **end for**
10:    // update sampling location
11:    $\boldsymbol{\mu}_i^{t+1} = \frac{\sum_j z_{i,j}\mathbf{x}_j}{\sum_j z_{i,j}}$
12:  **end for**
13:  $t \leftarrow t + 1$
14: **until** convergence criteria are satisfied
```

Please note that our heuristic listed in Algorithm 3 has been specifically designed for the squared exponential kernel:

$$k(\mathbf{x}, \mathbf{x}') = \sigma_f^2 \exp\left(- \frac{(\mathbf{x} - \mathbf{x}')^T(\mathbf{x} - \mathbf{x}')}{2l^2} \right) \tag{8.2}$$

We will better detail how the iterative procedure of the heuristic updates the sampling locations in the next section.

8.2.1 Description of the heuristic

During iteration t for each sampling point i (for loop in lines 3-12) the iterative step of the heuristic is composed of three components:

1. Line 5 - We compute Γ which represent the Gaussian process variance (using equation 3.7) at every location $\mathbf{x}_j \in \mathcal{X}$, given $\mathcal{K}^t \setminus \boldsymbol{\mu}_i^t$, that is the set of sampling location at time t excluded $\boldsymbol{\mu}_i^t$.

2. Line 8 - We compute the soft indicator variables $z_{i,j}$ for each $\mathbf{x}_j \in \mathcal{X}$.

3. Line 11 - We compute the location of sampling point $\boldsymbol{\mu}_i^{t+1}$ for the next iteration $t+1$.

On an abstract level we can think of the execution of the algorithm in this terms: an indicator variable $z_{i,j}$ (computed as in line 8) represent the attractive force that a domain location $\mathbf{x}_j \in \mathcal{X}$ exert on sampling point $\boldsymbol{\mu}_i$. As mentioned before $z_{i,j}$ represent the degree of which the variance of point $\mathbf{x}_j$ is reduced by sampling in position $\boldsymbol{\mu}_j$. We compute this force by taking into account how the variance of the Gaussian process is already effected by the presence of the other sampling locations $\mathcal{K}^t \setminus \boldsymbol{\mu}_i^t$.

The choice of using:

$$\frac{\sigma_f^2}{\sigma_f^2 + \sigma_n^2} e^{-\frac{\left|\left|\mathbf{x}_j - \boldsymbol{\mu}_i^t\right|\right|^2}{l^2}} \tag{8.3}$$

as scalar factor for the variance $\Gamma(\mathbf{x}_j)$ in location $\mathbf{x}_j$ represent a good choice as dictated by Theorem 1.

Theorem 1. *Using a squared exponential kernel, in a Gaussian process with a single sampling point in a generic location $\boldsymbol{\mu}$, the variance of this Gaussian process at any location $\boldsymbol{x}$ is equivalent to the multiplication of the variance of the Gaussian process without any sampling point and $1 - f(\boldsymbol{x})$, where $f(\boldsymbol{x})$ is a Gaussian function. Specifically:*

$$\sigma^2(\boldsymbol{x}) = \sigma_f^2\big(1 - f(\boldsymbol{x})\big) \tag{8.4}$$

with $f(\boldsymbol{x}) = ae^{-\frac{||\boldsymbol{x}-\boldsymbol{b}||^2}{2c^2}}$ for some a, $\boldsymbol{b}$ and c that are dependent on the hyper-parameters of the kernel.

Proof. Given the equation of the variance of a Gaussian process:

$$\sigma^2(\mathbf{x}) = k(\mathbf{x}, \mathbf{x}) - \mathbf{k}_*^T(\mathbf{K} + \sigma_n^2\mathbf{I})^{-1}\mathbf{k}_* \tag{8.5}$$

and the equation of the squared exponential kernel:

$$k(\mathbf{x}, \mathbf{x}') = \sigma_f^2 \exp\left(- \frac{(\mathbf{x} - \mathbf{x}')^T(\mathbf{x} - \mathbf{x}')}{2l^2}\right) = \sigma_f^2 \exp\left(- \frac{||\mathbf{x} - \mathbf{x}'||^2}{2l^2}\right) \tag{8.6}$$

for a single sampling point in a generic location $\boldsymbol{\mu}$ we obtain:

$$\begin{aligned}
\sigma^2(\mathbf{x}) &= k(\mathbf{x},\mathbf{x}) - k(\mathbf{x},\boldsymbol{\mu})\Big(k(\boldsymbol{\mu},\boldsymbol{\mu}) + \sigma_n^2\Big)^{-1} k(\mathbf{x},\boldsymbol{\mu}) \\
&= k(\mathbf{x},\mathbf{x}) - \frac{k(\mathbf{x},\boldsymbol{\mu})^2}{k(\boldsymbol{\mu},\boldsymbol{\mu}) + \sigma_n^2} \\
&= \sigma_f^2 - \frac{\left(\sigma_f^2 e^{-\frac{||\mathbf{x}-\boldsymbol{\mu}||^2}{2l^2}}\right)^2}{\sigma_f^2 + \sigma_n^2} \\
&= \sigma_f^2 - \frac{\sigma_f^4 e^{-\frac{||\mathbf{x}-\boldsymbol{\mu}||^2}{l^2}}}{\sigma_f^2 + \sigma_n^2} \\
&= \sigma_f^2\left(1 - \frac{\sigma_f^2}{\sigma_f^2 + \sigma_n^2} e^{-\frac{||\mathbf{x}-\boldsymbol{\mu}||^2}{l^2}}\right) \\
&= \sigma_f^2\big(1 - a e^{-\frac{||\mathbf{x}-\mathbf{b}||^2}{2c^2}}\big) \\
\text{With } a &= \frac{\sigma_f^2}{\sigma_f^2 + \sigma_n^2},\ \mathbf{b} = \boldsymbol{\mu} \text{ and } c = \frac{l}{\sqrt{2}}
\end{aligned}$$

□

Since the indicator variable $z_{i,j}$ represents an attractive force proportional to the amount of variance reduction obtainable in location $\mathbf{x}_j$ by sampling in position $\boldsymbol{\mu}_j$, consequently, the update step of the sampling locations (line 11) represents the sum of the contributions from all the attractive forces from the domain locations $\mathbf{x}_j \in \mathcal{X}$.

The main loop of the procedure (lines 2-14) iterates until some convergence criteria of the procedure are met. These can be defined in different ways. In our implementation we used two criteria. Specifically, one condition is that the procedure has to always improve the solution at each iteration, hence it terminates when a new iteration produces a configuration that is not reducing the variance of the Gaussian process further. The second condition is based on a convergence factor cf that is used in the same way as presented in the gradient descent procedure (Section 7.2.2). Specifically, this parameter is intended as a threshold to determine whether the procedure has to terminate or not. cf specifies what is the lowest percentage (with respect to the dataset diameter) of displacement that any points we are adapting can move. As described in the previous chapter, the cf parameter acts as a trade-off between the precision of the solution and the computation (number of iterations) required to converge. For small values the algorithm is allowed to go through its iterations as long as at least one of the points in space is moving by a small amount. Larger values will make the procedure stop earlier with a solution that may of course be further from an optimum than when small values are used.

In the next section we describe the approximation that our heuristic is performing by comparing how its iterative procedure adapts the sampling locations with respect to what a gradient descent algorithm (as the one we analyzed in Chapter 7) would do under the same conditions.

8.2.2 Approximation performed by our heuristic

First of all, observe once again the objective function in equation 8.1. As previously mentioned given K sampling location in a d-dimensional space this function is Kd-dimensional. Specifically, each dimension of the objective function represent the position of a sampling point in one of the dimensions of the domain. Observe for example Figure 8.1a, here we can see a bi-dimensional objective function that refers to the problem of adapting two sampling points in a mono-dimensional domain (see Figure 8.1b).

As we described in the previous chapter, the idea behind a gradient descent algorithm is that given a point in the objective function at iteration t (that represent a specific configuration of sampling locations $\mathcal{K}^t$ in the domain) we compute the gradient in that point with respect to any dimension. Computing the derivative for a specific dimensionality of the objective function corresponds to computing the direction in which a single sampling point $\boldsymbol{\mu}_i$ has to move along a single dimension d of the domain given the current configuration of the other sampling points $\mathcal{K}^t \setminus \boldsymbol{\mu}_i^t$.

For a graphical representation of this concept let's take look again at the images in Figure 8.1. The current configuration of sampling points is $\mathcal{K} = \{\boldsymbol{\mu}_1 = 25, \boldsymbol{\mu}_2 = 50\}$. The value of the objective function (Figure 8.1a) in the point $J(25, 50)$ represent the sum of the variance (Figure 8.1b) along the entire domain. Computing the derivative in $J(25, 50)$ along the first dimension of the objective function correspond to computing the derivative of the section of the objective function given the second dimension, i.e. $\boldsymbol{\mu}_2 = 50$ (Figure 8.1c). Respectively, computing the derivative along the second dimension of the objective function correspond to computing the derivative of the section of J given the first dimension, i.e. $\boldsymbol{\mu}_1 = 25$ (Figure 8.1d).

The heuristic proposed in this chapter iterates in a similar way. However, the objective function optimized is an approximation of the original one in equation 8.1. Specifically, as described in Theorem 2, at each iteration the point in the objective function that represents the current configuration of sampling locations $\mathcal{K}^t$ moves along each axis in the same direction as gradient descent would do on an approximated objective function.

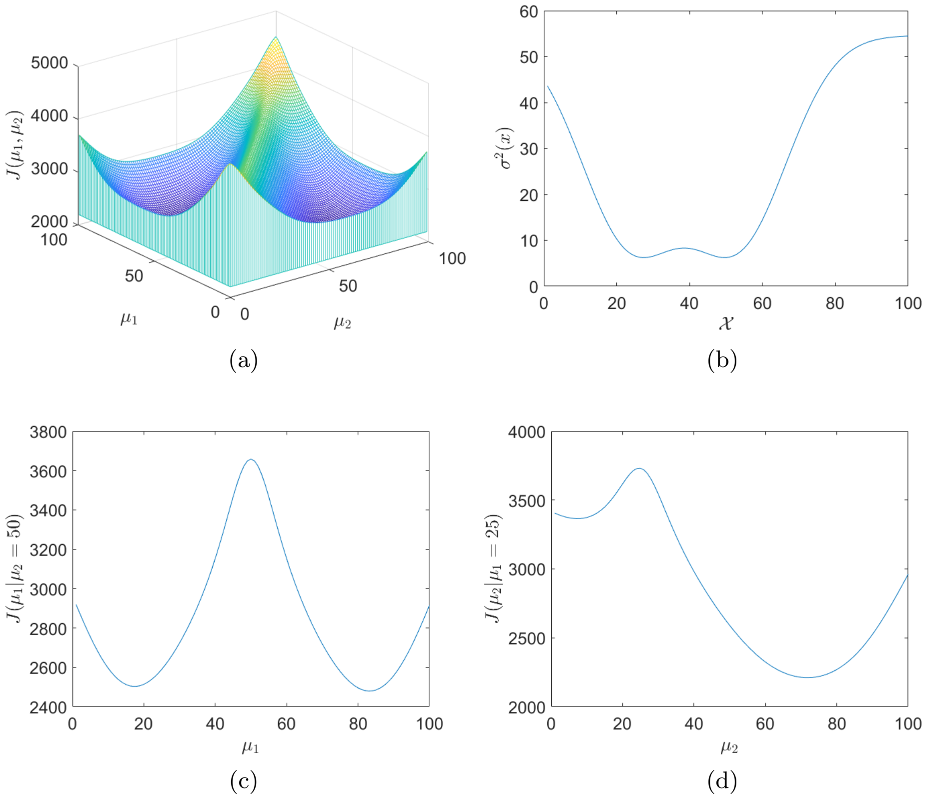


Figure 8.1: (a) Example of the objective function given 2 sampling points in a mono-dimensional domain. (b) Plot of the variance of the Gaussian process with sampling points $\mathcal{K} = \{\boldsymbol{\mu}_1 = 25, \boldsymbol{\mu}_2 = 50\}$. (c) Section of the objective function given $\mu_2 = 50$ and varying μ_1. (d) Section of the objective function given $\mu_1 = 25$ and varying μ_2.

Theorem 2. *Given that $\Gamma > 0$, the direction of movement of a sampling point μ is in the same direction as gradient descent would compute in the location μ on the objective function $\Gamma * \left(1 - \frac{\sigma_f^2}{\sigma_f^2+\sigma_n^2} e^{-\frac{(x-\mu)^2}{l^2}}\right)$. That is:*

$$sign\left(\mu - \frac{\int \Gamma(x) \frac{\sigma_f^2}{\sigma_f^2+\sigma_n^2} e^{-\frac{(x-\mu)^2}{l^2}} x \, dx}{\int \Gamma(x) \frac{\sigma_f^2}{\sigma_f^2+\sigma_n^2} e^{-\frac{(x-\mu)^2}{l^2}} \, dx}\right) = sign\left(\frac{d}{dx}\left(\Gamma * \left(1 - \frac{\sigma_f^2}{\sigma_f^2 + \sigma_n^2} e^{-\frac{(x-\mu)^2}{l^2}}\right)\right)(\mu)\right)$$

Proof.

To simplify the notation let $a = \dfrac{\sigma_f^2}{\sigma_f^2 + \sigma_n^2}$

$$sign\left(\frac{d}{dx}\left(\Gamma * \left(1 - ae^{-\frac{(x-\mu)^2}{l^2}}\right)\right)(\mu)\right)$$

For the differentiation property of the convolution $\dfrac{d}{dx}(f * g) = \dfrac{df}{dx} * g = f * \dfrac{dg}{dx}$

$$= sign\left(\left(\Gamma * \frac{2ax}{l^2}e^{-\frac{x^2}{l^2}}\right)(\mu)\right)$$

$$= sign\left(\left(\int \Gamma(\tau)\frac{2a(x-\tau)}{l^2}e^{-\frac{(x-\tau)^2}{l^2}}d\tau\right)(\mu)\right)$$

$$= sign\left(\int \Gamma(\tau)\frac{2a(\mu-\tau)}{l^2}e^{-\frac{(\mu-\tau)^2}{l^2}}d\tau\right)$$

$$= sign\left(\frac{2}{l^2}\int \Gamma(\tau)ae^{-\frac{(\mu-\tau)^2}{l^2}}\mu - \Gamma(\tau)ae^{-\frac{(\mu-\tau)^2}{l^2}}\tau \, d\tau\right)$$

Since $\dfrac{2}{l^2} > 0$

$$= sign\left(\mu\int \Gamma(\tau)ae^{-\frac{(\mu-\tau)^2}{l^2}}\,d\tau - \int \Gamma(\tau)ae^{-\frac{(\mu-\tau)^2}{l^2}}\tau\,d\tau\right)$$

$$= sign\left(\mu - \frac{\int \Gamma(\tau)ae^{-\frac{(\mu-\tau)^2}{l^2}}\,\tau\,d\tau}{\int \Gamma(\tau)ae^{-\frac{(\mu-\tau)^2}{l^2}}\,d\tau}\right)$$

□

In Theorem 2 the left side of the equation represents the direction of movement of a sampling point $\boldsymbol{\mu}$ along an axis as computed by our heuristic (line 11). This movement correspond to the movement that gradient descent would compute in the location $\boldsymbol{\mu}$ of the function:

$$\Gamma * \left(1 - \frac{\sigma_f^2}{\sigma_f^2 + \sigma_n^2}e^{-\frac{(x-\mu)^2}{l^2}}\right) \tag{8.7}$$

We previously showed in theorem 1 that the variance at any location $\mathbf{x}$ of a Gaussian process with a single sampling point in a generic location $\boldsymbol{\mu}$ is equivalent to the multiplication of the variance of the Gaussian process without any sampling point and $1 - \frac{\sigma_f^2}{\sigma_f^2+\sigma_n^2}e^{-\frac{(x-\mu)^2}{l^2}}$.

By definition of the convolution, equation 8.7 is computing the integral of the multiplication between Γ and $1 - \frac{\sigma_f^2}{\sigma_f^2+\sigma_n^2}e^{-\frac{(x-\mu)^2}{l^2}}$ by shifting one over the other. This represents an approximation of the section of the objective function $J(\boldsymbol{\mu}|\mathcal{K} \setminus \boldsymbol{\mu})$, that is, the section of the objective function varying $\boldsymbol{\mu}$ given the other sampling points $\mathcal{K} \setminus \boldsymbol{\mu}$.

If we look to the plots in Figure 8.1, we are for example approximating the section $J(\mu_1|\mu_2 = 50)$ in Figure 8.1c by computing Equation 8.7 where Γ represent the variance of the Gaussian process given μ_2.

8.3 Empirical evaluation

Here we present the empirical evaluation of our heuristic approach. Similarly to what we have done for the gradient descent procedure presented in Chapter 7, here we want to test the performance of our heuristic approach under different conditions. To this aim we used the same datasets and settings as before.

To sum up, we have datasets with domains that ranges from 1 up to 5 dimensions with cardinality $|\mathcal{X}|$ of at least 1000 points. We used 20 different length-scale l, 15 different σ_f and 10 different σ_n to set the hyperparameters of the square exponential kernel and the noise of measurement respectively. The procedure has been tested by adapting K sensing locations with $K \in \{2, 3, 4, 5, 6, 7\}$, the number of sampling points is reasonable for applications with datasets of comparable size. Since the heuristic requires a set of initial sampling locations, also in this case they have been initialized using the solutions provided by the submodular greedy procedure.

These empirical evaluations allows us to determine the degree of improvement that the heuristic obtains with respect to the submodular greedy procedure along with a comparison with the results previously obtained with gradient descent. Moreover, we also analyze the performance in terms of computation time proving that the heuristic approach represents a good trade-off between quality of the solution and computation time with respect to gradient descent.

In order to obtain a fair comparison the cf parameter used in the convergence criteria (as explained in Section 8.2.1) has been set to the same value as in the previous experiments with gradient descent, i.e. $cf = 1/1000$ and all the described algorithms have been tested using the same machine equipped with an AMD FX 6300 processor with 16GB RAM.

To summarize, by considering different combinations of hyperparameters, dimensionality of the dataset and number of sampling points, we have performed 90,000 experiments that allows to characterize and study the improvement obtainable with the heuristic presented in this chapter. Moreover, we also performed the 90,000 experiments by initializing the points randomly instead of using the submodular solutions. We detail the result of such experiments in the next section.

8.4 Results

As previously done in Section 7.4 for the gradient descent technique, here we analyze the results of the empirical evaluation of the heuristic from different points of view. Specifically:

- We present the advantage of the heuristic with respect to the submodular initialization as a function of the hyperparameters used in the Gaussian process (Section 8.4.1).
- We analyze the performance of the technique varying the number of sampling points and the dimensionality of the domain (Section 8.4.2).
- We provide the results using an initial configuration of sampling points randomly generated (Section 8.4.3).
- We compare our techniques (heuristic and gradient descent) in terms of the computation time required (Section 8.4.4).

8.4.1 Results as a function of the hyperparameters

To explain the performance of the heuristic as a function of the hyperparameters of the Gaussian process, we take as example the two plots in Figure 8.2. These plots specific for the heuristic are the counterpart of the plots in Figure 7.2 for gradient descent. More in detail, in this pictures we can observe the % of improvement that the heuristic obtains with respect to the submodular solution by varying the hyperparameters in the two dimensional dataset by adapting 5 points: vertically the length-scale l of the kernel and horizontally the standard deviation σ_f of the function. The two pictures represent these improvements by fixing a single standard deviation of the noise measurement σ_n; the one to the right with a σ_n that is almost three times the one to the left.

The discussion presented in Section 7.4.1 still holds also for the heuristic approach. In fact, we can observe that in general the advantage obtainable with the heuristic as a function of the hyperparameters follows the same pattern as in gradient descent.

With very small length-scale the position where we make observations influences little to nothing the final amount of posterior variance. Hence, in these cases we cannot obtain an advantage with respect to the submodular greedy technique. On the other hand, when the length-scale becomes bigger we notice that the σ_f and σ_n parameters plays a role in the quality of the variance reduction obtainable. Specifically, the ratio σ_f/σ_n affects the quality of the results: the higher the ratio the higher the improvements we can obtain.

If we compare the images for the heuristic (Figure 8.2) with their counterpart for gradient descent (Figure 7.2), we notice that the improvements

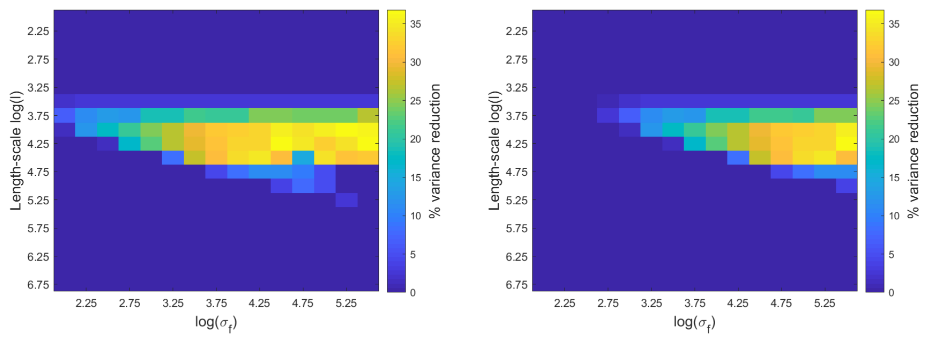


Figure 8.2: Results as a function of the hyperparameters. Horizontally are variations in the standard deviation σ_f and on the vertical axis in the length-scale l. Colors represent the % of variance reduction of the heuristic relative to the submodular greedy solution. These results refer to the adaptation of 5 points in the 2-dimensional dataset for a fixed σ_n. Specifically in the right image σ_n is about three times higher then in the left one.

offered by the heuristic are confined to a smaller area, i.e, to a smaller portion of hyperparameter combinations. This is indeed induced by the fact that the heuristic (as demonstrated in Theorem 2) operates on an approximation of the objective function.

8.4.2 Varying the number of points and dimensionality

Here we study the performance of the heuristic with respect to the submodular greedy solution depending on the number K of sampling locations and the dimensionality of the domain. In tables 8.1 and 8.2 we report the average and maximum percentage gain of variance reduction that the heuristic obtains with respect to the total posterior variance given by the submodular greedy solution. Specifically, each entry of the tables reflect the improvement obtained in a specific combination of dimensionality of domain and number of sampling points.

Table 8.1: Average % gain of the heuristic with respect to the submodular greedy solution.

	Number of points (K)					
	2	3	4	5	6	7
1-D	24.02	12.32	14.55	14.95	14.58	6.87
2-D	1.76	5.58	7.57	4.81	4.80	5.65
3-D	0.16	0.43	1.48	0.86	0.93	1.54
4-D	0.01	0.03	0.16	0.50	0.11	0.17
5-D	0.00	0.03	0.19	0.43	1.01	0.23

Table 8.2: Maximum % gain of the heuristic with respect to the submodular greedy solution.

	Number of points (K)					
	2	3	4	5	6	7
1-D	69.59	82.38	80.84	75.82	86.43	89.13
2-D	8.44	27.13	46.69	43.43	52.72	60.15
3-D	1.15	3.23	8.98	8.85	11.32	14.69
4-D	0.10	0.25	1.03	4.67	1.18	7.01
5-D	0.08	0.52	2.24	5.00	14.22	6.82

In more details, Table 8.1 represent the average % gain of the heuristic with respect to the submodular greedy solution. This average is computed over all the 3000 combination of hyperparameters for a given configuration of dimensionality of the domain and number of sampling points. In Table 8.2 we report the maximum % gain encountered between all the possible 3000 combination of hyperparameters.

In general the results obtained follow the same trend of the those proposed for the gradient descent approach. If we compare the average improvements of the heuristic in Tables 8.1 with those of gradient descent in Table 7.1, we notice that the advantage of the heuristic are lower for the reasons explained in the previous section. However, if we compare the maximum improvements of the heuristic in Tables 8.2 with those of gradient descent in Table 7.2 we notice that in some cases (e.g. 2 points in one dimension) the heuristic is able to achieve a better result, meaning that in some restricted cases the heuristic converges to a better solution than gradient descent even with the same convergence conditions.

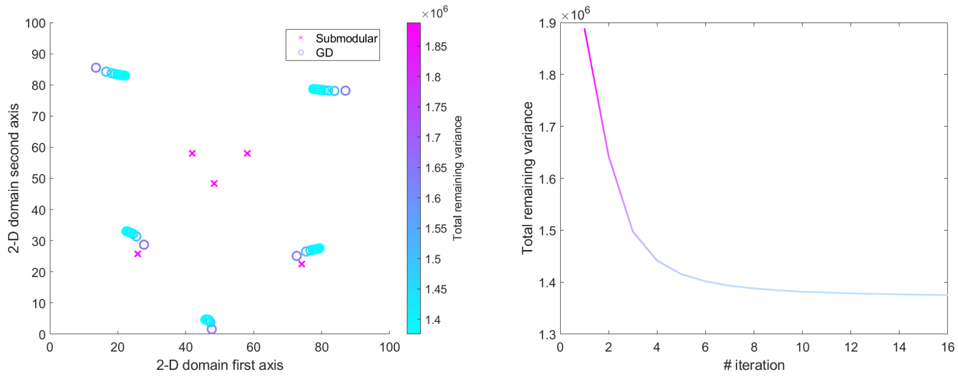
For a graphical comparison of the execution of both gradient descent and the heuristic algorithms on an example instance see Figure 8.3.

8.4.3 Random initialization

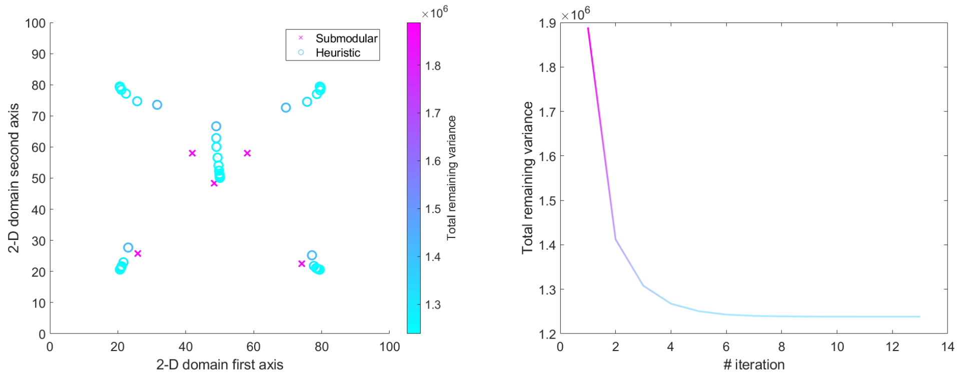
Similarly to the discussion above, here we report results of the heuristic empirical evaluation when the procedure has been initialized with a set of randomly chosen sampling locations across the domain.

Tables 8.3 and 8.4 refers respectively to the average and maximum improvements of the heuristic with respect to the random initial collocation of points. These results represent the gain in terms of percentage of variance reduction with respect to the variance of the Gaussian process with the measurement points in random locations. As we expected, since the random collocation of point can represent a very bad quality solution compared to the submodular greedy procedure, results show much bigger improvements.

However, similarly to what observed for the gradient descent results, also for the heuristic approach a more interesting point of view is showed in Table



(a)



(b)

Figure 8.3: Experiments on the 2-dimensional dataset with 5 sampling points. (a) Execution of the gradient descent technique and (b) execution of the heuristic.

8.5. Here we present the comparison of the total posterior variance obtained with the heuristic when initialized with a random configuration of sampling location with respect to the total posterior variance when initialized with the submodular solution.

Specifically, in Table 8.5 we show the maximum improvements encountered among all the 3000 combinations of hyperparameters. Since the randomly generated sampling locations can vary considerably this is reflected in the quality of the results. However what Table 8.5 shows is that in some cases the heuristic is able to obtain better results with a random initialization rather then using a submodular greedy procedure to select the starting configuration.

The aforementioned tables present results considering only a single randomly generated initialization of sampling points per instance (i.e. per combination of hyperparameters, dimensionality of domain and number of sam-

Table 8.3: Average % gain of the heuristic with respect to a single random configuration.

	Number of points (K)					
	2	3	4	5	6	7
1-D	25.43	22.65	23.23	23.90	24.44	25.18
2-D	15.83	15.31	14.15	12.77	13.22	13.79
3-D	15.57	9.83	12.52	9.79	8.23	7.99
4-D	14.85	9.42	8.00	10.71	9.21	7.71
5-D	14.27	9.27	7.05	7.16	9.65	8.74

Table 8.4: Maximum % gain of the heuristic with respect to a single random configuration.

	Number of points (K)					
	2	3	4	5	6	7
1-D	97.54	98.15	99.19	99.80	99.78	99.95
2-D	74.91	94.43	87.57	84.49	85.30	89.82
3-D	65.43	64.66	88.37	75.29	55.94	67.26
4-D	64.93	58.35	61.02	90.83	74.40	70.95
5-D	58.81	53.68	47.70	54.30	83.79	75.78

pling points). Since the selection of the initial configuration of sampling points can vary considerably along the domain, we also performed a more detailed test on a small subset of cases.

In Figure 8.4 we show results varying the length-scale l and σ_f and by fixing a specific σ_n parameter on two specific cases. In particular, we have selected the two dimensional dataset and we used the heuristic by randomly initializing two sampling locations 100 times. The same for the case of six points in the three dimensional dataset. We can observe that by performing multiple randomly initialized executions, on average we obtain a spectrum of improvements similar as what previously shown in Figure 8.2 and comparable to the related results for the gradient descent technique in Figure 7.4.

8.4.4 Computation time comparison

On top of not having to tune a parameter, or to implemented a mechanism that controls the learning rate of the procedure, the main advantage offered by our heuristic is a fast computation time. In what follows, we report the results comparing the run time required by our heuristic with the gradient descent technique presented in the previous chapter.

In Table 8.6 we report the computation time for the two procedures. Also in this case, each entry of the table represents the average over the

Table 8.5: Maximum % gain of the heuristic starting from a random configuration with respect to the heuristic itself starting from the submodular greedy solution.

	Number of points (K)					
	2	3	4	5	6	7
1-D	68.81	91.85	94.42	89.69	51.40	74.83
2-D	22.34	48.10	49.40	35.12	67.00	68.31
3-D	9.38	19.08	31.85	20.79	12.46	7.59
4-D	8.22	9.91	10.28	15.38	8.24	3.62
5-D	0.40	5.69	10.36	9.39	9.84	3.72

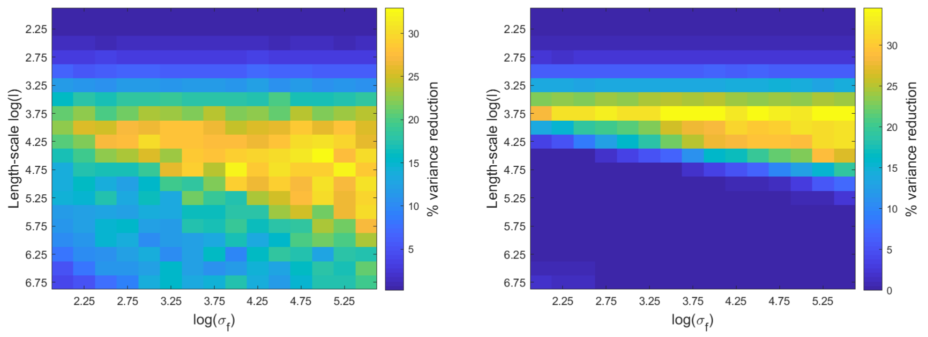


Figure 8.4: Average gain over 100 randomly initialized execution of the heuristic. Left with 2 points in the 2-dimensional dataset and right 6 points in the 3-dimensional dataset.

3000 combinations of hyperparameters for a specific configuration of dimensionality of the domain and number of sampling points. As we can observe the computation times for the heuristic algorithm constitute a fraction of those for the gradient descent procedure.

Given that the heuristic is much faster but obtains a smaller advantage with respect to gradient descent, a more interesting comparison is offered in Table 8.7.

As we can observe in this last table, in most cases the heuristic is able to obtain a high percentage of improvement with a fraction of the time. For example if we look at the case of two points in a one dimensional domain, we observe that with only 11.6% of computation time it is able to obtain 73.1% of the variance improvement.

Although results reported in this chapter and in the previous one refers to experiments that vary considerably from one to another (given the wide variability of hyperparameters and datasets) we can give an even more concise description. If we average the computation times in Table 8.6 we see that the heuristic takes 0.40 seconds per instance whereas gradient descent

Table 8.6: Computation times in seconds of the heuristic and gradient descent (both with submodular initialization). Each entry represent the average over the 3000 hyperparameters combinations for a specific dimensionality of the domain and number of sampling locations.

	Computation time (s)											
	Gradient descent						**Heuristic**					
	2	3	4	5	6	7	2	3	4	5	6	7
1-D	1.12	1.70	1.87	2.55	2.53	3.13	0.13	0.05	0.17	0.19	0.21	0.13
2-D	2.13	2.81	6.53	4.08	7.27	10.92	0.24	0.29	0.40	0.20	0.31	0.66
3-D	2.05	2.84	4.27	7.15	11.07	12.63	0.13	0.21	0.77	0.83	0.61	0.86
4-D	3.00	6.10	7.88	9.46	18.27	17.32	0.14	0.24	0.69	0.92	0.96	0.89
5-D	3.15	7.29	14.96	20.95	20.59	25.76	0.03	0.10	0.31	0.40	0.32	0.60

Table 8.7: Percentage of time and percentage of improvement in variance reduction of the heuristic with respect to gradient descent.

	% of Time						**% Improvement**					
	2	3	4	5	6	7	2	3	4	5	6	7
1-D	11.6	2.8	9.3	7.6	8.3	4.3	73.1	67.6	82.7	87.7	98.8	80.8
2-D	11.2	10.4	6.2	4.9	4.3	6.0	42.5	33.0	38.5	52.5	35.1	39.0
3-D	6.4	7.5	18.1	11.6	5.5	6.8	15.5	15.1	16.8	10.7	8.8	18.7
4-D	4.5	4.0	8.8	9.7	5.2	5.1	4.2	2.6	8.3	10.0	3.3	3.5
5-D	0.8	1.3	2.0	1.9	1.5	2.3	0.5	5.7	17.3	26.0	25.9	10.3

takes 8.05. Now, if we average the improvement results in Table 8.1 for the heuristic and in Table 7.1 for gradient descent we obtain 4.19 and 8.42 respectively. We can therefore conclude observing that the heuristic obtains 50% of the improvement that gradient descent can achieve but with only 5% of the computation time, thus constituting an excellent compromise between quality of the results and computation time required.

8.5 Conclusions

In this chapter we presented a novel heuristic algorithm to minimize the posterior variance of a Gaussian process. The proposed technique falls within the same context as the gradient descent approach proposed in the previous chapter. These contributions are motivated by the fact that in some applications, such as environmental monitoring, the locations where measurements are performed does not have to be confined in predetermined points in space, but rather the domain is continuous. The heuristic does not requires the domain to be discrete and it can iteratively improve the solution by freely moving the measurement points in a continuous manner.

The proposed contribution has been analyzed under different settings in order to show the advantages and the generality of the approach. Although the improvements are lower than the one that gradient descent can achieve, the heuristic with a fraction of the computation time should be able to better scale to larger instances while still improving with respect to submodular in a context where sampling locations can be adapted in a continuous domain.

Part IV

QUBO for environmental monitoring applications

Chapter 9

QUBO for sensor placement

9.1 Introduction

Several problems in artificial intelligence and pattern recognition, such as the ones discussed in the previous parts of this thesis, are computationally intractable due to their inherent complexity and the exponential size of the solution space. Hence, the development of approximated and heuristic approaches.

Quantum information processing could provide a viable alternative to combat such a complexity. As discussed in the introduction chapter (Section 1.4.1) a notable work in this direction is due to the recent development of the D-Wave computer, whose processor has been designed to solve Quadratic Unconstrained Binary Optimization (QUBO) problems. We presented in Section 2.3 different works in literature that investigate the use of a QUBO formulation to address typical artificial intelligence and pattern recognition problems.

In this part of the thesis we also investigate this possibility by encoding optimization problems related to environmental monitoring applications into QUBO models. More in detail, in this chapter we start by tackling a problem that falls within the same context as what presented in Part III of the thesis. Specifically, we want to optimize a set of sensing locations in order to minimize the posterior variance of a Gaussian process model.

However, in contrast to what previously done in Part III where we optimize in a continuous manner, here we consider the scenario where the sampling locations can be selected from a finite discrete set of predetermined available locations. Namely, the domain of interest $\mathcal{X}$ is discretized. In this context our direct competitors are discrete (combinatorial) techniques such as the greedy submodular optimization [103, 105, 145] (see Section 2.2 for related work in this context).

We can summarize the contributions of this chapter as follows:

- We propose a QUBO model to optimize the sensing locations and minimize the posterior variance of a Gaussian process.

- We provide a mathematical demonstration that the optimum of our QUBO model satisfies the constraint of the problem.
- We study the performance of the proposed QUBO model with respect to submodular greedy and random sampling selections.

9.1.1 Problem definition

The definition of the problem described in this chapter is very similar to what was presented in the two contributions of Part III of the thesis. Given a Gaussian process and a discretized domain $\mathcal{X}$, we want to select a set of K points within $\mathcal{X}$ where to perform measurements in order to minimize the total posterior variance of the Gaussian process. Specifically we want to select a set of K measurements taken at locations $\mathcal{K} = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_K\}$ such that we minimize the following objective function:

$$J(\{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_K\}) = \sum_{\mathbf{x}_i \in \mathcal{X}} \sigma^2(\mathbf{x}_i) \tag{9.1}$$

where $\sigma^2(\mathbf{x}_i)$ is the Gaussian process variance computed in point $\mathbf{x}_i$ as defined by Equation 3.7. Notice that the variance computed with Equation 3.7 in locations $\mathbf{x}_i \in \mathcal{X}$ is dependent on the set of sampling points $\mathcal{K}$ as explained in Section 3.1.1.

However in this case, our goal is to model a QUBO objective function (Equation 3.20) that approximates our problem:

$$O(z_1, \dots, z_n) \approx J(\{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_K\}) \tag{9.2}$$

9.2 QUBO formulation of our problem

In this section we describe our QUBO model for the Gaussian process posterior variance minimization problem with a discrete domain. For a reference on QUBO models see Section 3.5.

Given a discrete domain $\mathcal{X}$ the starting point to build our QUBO model is represented by a complete graph composed of exactly $|\mathcal{X}|$ nodes. Specifically, in a complete graph each node is connected to all the others, hence the total number of edges is $\dfrac{|\mathcal{X}|(|\mathcal{X}|-1)}{2}$.

In order to explain how we set the values of such graph, we decompose the study and the description of the model into two parts:

1. We describe how to set values directly related to the variance of the Gaussian process.
2. We describe how to set values in order to implement the constraint of the problem (i.e, the number of sampling points must be exactly K).

9.2.1 Variance values

As previously mentioned, given a domain $\mathcal{X}$ which represents the set of candidate sampling locations, we build a graph composed by $|\mathcal{X}|$ nodes where each node corresponds to a specific location of the domain. Since our problem requires to select a set of sampling points such that the posterior variance of the Gaussian process is minimized, a natural representation is to assign to each node of the graph the amount of variance reduction obtainable by sampling the Gaussian process in the location represented by that node. Specifically, the value of node i is:

$$\alpha_i \triangleq J(\{\mathbf{x}_i\}) - J(\emptyset) \tag{9.3}$$

Since the objective function of a QUBO instance has to be minimized and the objective of our problem is to minimize the posterior variance of a Gaussian process, with Equation 9.3 we set α_i values as the negative amount of variance reduction. This is guaranteed by the properties of Gaussian processes. Selecting a node gives us an improvement (i.e, a lower value of the QUBO objective function) that is equivalent to the amount of variance reduction obtainable by adding a sampling point in the location represented by node i of the graph.

We can imagine these α_i as "selecting forces", in fact the amount of variance reduction obtainable is a force that attracts a specific location of the domain to be selected as a candidate sampling point. However, when we select two sampling points, the total amount of variance reduction is lower than the sum of the reduction obtainable from the two sampling points individually. Formally:

$$J(\emptyset) - J(\{\mathbf{x}_i, \mathbf{x}_j\}) < \big(J(\emptyset) - J(\{\mathbf{x}_i\})\big) + \big(J(\emptyset) - J(\{\mathbf{x}_j\})\big) \tag{9.4}$$

As a consequence, in our QUBO objective function we have to keep into account the mutual interaction between sampling points. To this aim in our QUBO graph we set edges values as follows:

$$\begin{aligned}\beta_{i,j} &\triangleq J(\{\mathbf{x}_i, \mathbf{x}_j\}) - J(\emptyset) - \alpha_i - \alpha_j \\ &= J(\{\mathbf{x}_i, \mathbf{x}_j\}) - J(\emptyset) - J(\{\mathbf{x}_i\}) + J(\emptyset) - J(\{\mathbf{x}_j\}) + J(\emptyset) \\ &= J(\{\mathbf{x}_i, \mathbf{x}_j\}) - J(\{\mathbf{x}_i\}) - J(\{\mathbf{x}_j\}) + J(\emptyset)\end{aligned} \tag{9.5}$$

The total variance reduction obtainable with two sampling points is proportional to the distance between the two locations. Points close to each other would have a high mutual interaction and as a consequence the value $\beta_{i,j}$ between these locations would be high. We can imagine these $\beta_{i,j}$ almost as the opposite of a "selecting force", discouraging the simultaneous selection of points close to each other. Please note that this is dependent on the length-scale of the process encoded in the hyperparameters of the kernel used.

Values $\beta_{i,j}$ are guaranteed to be always positive as we prove in the following proposition.

Proposition 1. *Values $\beta_{i,j}$ are guaranteed to be higher that 0.*

Proof. Given Equation 9.4:

$$\begin{aligned}
&J(\emptyset) - J(\{\mathbf{x}_i, \mathbf{x}_j\}) < \big(J(\emptyset) - J(\{\mathbf{x}_i\})\big) + \big(J(\emptyset) - J(\{\mathbf{x}_j\})\big) \\
&J(\emptyset) - J(\{\mathbf{x}_i, \mathbf{x}_j\}) < 2J(\emptyset) - J(\{\mathbf{x}_i\}) - J(\{\mathbf{x}_j\}) \\
&- J(\emptyset) - J(\{\mathbf{x}_i, \mathbf{x}_j\}) + J(\{\mathbf{x}_i\}) + J(\{\mathbf{x}_j\}) < 0 \\
&J(\emptyset) + J(\{\mathbf{x}_i, \mathbf{x}_j\}) - J(\{\mathbf{x}_i\}) - J(\{\mathbf{x}_j\}) > 0
\end{aligned}$$

$\square$

Values α_i and $\beta_{i,j}$ as described above depends exclusively from the Gaussian process. However, this is not enough to guarantee that the minimum of the QUBO objective function represents a solution where exactly K sampling points are selected. To overcome this problem in the next section we describe how to implement such constraint.

9.2.2 Implementation of the constraint

In general, to implement a constraint into an unconstrained problem we have to encode it as penalties in the objective function. Since a QUBO model has to be minimized, any combination that does not satisfy the constraint must have a higher value to ensure that it does not represent the optimal solution.

In what follows we describe how to implement the constraint of our problem into a complete graph. Specifically, a feasible solution has to represent a set of exactly K sampling locations. This is known as the *cardinality constraint* and is common in QUBO problems. However, in what follow we present our derivation that is useful to implement it in relation with values α_i and $\beta_{i,j}$ computed from the variance of a Gaussian process.

Specifically, we further divide the description of our implementation of this constraint in two separate parts:

1. We describe how to guarantee that exactly K nodes are selected in an zero-value graph (i.e. a weighted graph with zero values on every node and every edge).

2. We describe how to guarantee that the constraint is strong enough given that we want to implement it in a graph with values computed using Equations 9.3 and 9.5.

Selecting K nodes

In order to implement our constraint, we have to guarantee that the objective function is minimized if and only if the solution represents a configuration with exactly K nodes selected as candidate sampling points. For any other configuration the objective function must have a higher value.

Starting from a zero-value graph, we need to add some values to the nodes and edges to implement this constraint. Let assume that we assign the same value $A \in \mathbb{R}$ to every node and the same value $B \in \mathbb{R}$ to every edge. In a complete graph, a configuration where n nodes are selected leads to the following summation of values:

$$\frac{Bn(n-1)}{2} + An \tag{9.6}$$

If we want to guarantee that exactly K nodes are selected Equation 9.6 must have its minimum value when $n = K$.

Proposition 2. *Given the function* $\frac{Bn(n-1)}{2} + An$*, for any* $B > 0$ *if* $A = -BK + \frac{B}{2}$ *the minimum is in* $n = K$*.*

Proof.

$$\frac{Bn(n-1)}{2} + An = \frac{Bn^2}{2} - \frac{Bn}{2} + An$$

With $B > 0$ the quadratic function is a parabola that opens upwards and the minimum of the function is where the derivative is equal to 0.

$$\frac{\partial}{\partial n}\Big(\frac{Bn^2}{2} - \frac{Bn}{2} + An\Big) = 0$$
$$Bn - \frac{B}{2} + A = 0$$

Now we want to fix the derivative to be 0 when $n = K$.

$$BK - \frac{B}{2} + A = 0$$
$$A = -BK + \frac{B}{2}$$

□

With Proposition 2 we have shown that it is possible to implement the constraint for any K, hence also in the restricted case of our problem when $K \in \mathbb{N}^{[2,|X|-1]}$. This restriction is an inherent property of the problem since the cases for $K = 1$ or $K = |X|$ are trivial.

Guarantee the strength of the constraint

In the discussion above we have shown how to implement the constraint in a zero-value graph. However, in our problem we have to guarantee that it is satisfied in a complete graph that is already populated with values as shown in Section 9.2.1. Now we show how values A and B must be set such that the energy penalties are strong enough to guarantee that the constraint is always satisfied.

Given $A = -BK + \frac{B}{2}$ Equation 9.6 becomes:

$$\frac{Bn(n-1)}{2} - BKn + \frac{Bn}{2} = \frac{Bn^2}{2} - BKn \tag{9.7}$$

We want to analyze how this function increases as we move away from $n = K$ that should represent the minimum. Specifically, given $l \in \mathbb{N}$ we analyze how this function increases when $n = K + l$ and when $n = K - l$.

$$\begin{gathered}\left[\frac{Bn^2}{2} - BKn\right]_{n=K+l} - \left[\frac{Bn^2}{2} - BKn\right]_{n=K} = \\ \frac{B(K+l)^2}{2} - BK(K+l) - \left(\frac{BK^2}{2} - BK^2\right) = \frac{Bl^2}{2}\end{gathered} \tag{9.8}$$

$$\begin{gathered}\left[\frac{Bn^2}{2} - BKn\right]_{n=K-l} - \left[\frac{Bn^2}{2} - BKn\right]_{n=K} = \\ \frac{B(K-l)^2}{2} - BK(K-l) - \left(\frac{BK^2}{2} - BK^2\right) = \frac{Bl^2}{2}\end{gathered} \tag{9.9}$$

From Equations 9.8 and 9.9 we can observe that the constraint act as an energy penalty by increasing the value of the objective function quadratically with the difference of number of nodes selected with respect to a given K.

Note that this constraint is generic and can be applied to any other problem where a specific given number of nodes has to be selected. From the mathematical point of view of a QUBO function the constraint built so far guarantees that exactly K binary variables must have value 1 in order to minimize the function. Moreover, any feasible solution that satisfies the constraint starts with the same energy value as expressed by Equation 9.10, leaving the feasible solutions to 'compete' for which one is the optimal given a specific instance of the problem.

$$\frac{BK(K-1)}{2} - BK^2 + \frac{BK}{2} = -\frac{BK^2}{2} \tag{9.10}$$

The strength of the energy penalty is directly dependent on value B. In order to guarantee that the constraint is satisfied in a non zero-value graph (i.e. a weighted graph with non-zero weights) we have to selected B big enough to overcome additional 'forces'.

From a practical point of view, since $B \in \mathbb{R}^+$ we can just select a very big number to guarantee that the constraint is satisfied. For our specific problem (and objective function) we show in the next section that it is possible to compute a bound for B.

9.2.3 Ensuring a lower bound for the constraint

We want to compute a lower bound for B such that the constraint is satisfied in a generic weighted graph with values computed as presented in Section 9.2.1. Let start by making the following considerations:

- The constraint is satisfied if and only if the minimum of the QUBO objective function corresponds to a solution with exactly K nodes selected.
- The strength of the constraint (i.e. the energy penalty) as computed in Equations 9.8 and 9.9 is $\frac{Bl^2}{2}$, where $l \in \mathbb{N}$ represents the difference from K on the number of selected points.
- Values α_i computed with Equation 9.3 are always negative.
- Values $\beta_{i,j}$ computed with Equation 9.5 are always positive.

Given these considerations, we can compute the strongest energy value that a non feasible solution can have to deviate from a feasible one. In other terms, we can compute what is the highest contribution to the QUBO objective function that a configuration that does not satisfy the constraint can have. Once we compute this value, we can select B such that the energy penalty overcomes the worst case scenario. In what follow we analyze the two possible cases.

Configuration with more nodes selected

Let $\mathcal{A}$ be the set of α_i values computed with Equation 9.3. As a reminder, each of these α_i values represent the amount of variance reduction obtainable in a specific location of the domain. We can define $\mathcal{A}_n$ as the set of the n lowest values in $\mathcal{A}$:

$$\mathcal{A}_n \triangleq \begin{cases} \mathcal{A}_0 = \emptyset \\ \mathcal{A}_n = \mathcal{A}_{n-1} \cup \min(\mathcal{A} \setminus \mathcal{A}_{n-1}) \end{cases} \tag{9.11}$$

Since each α_i is a negative value and we want to minimize the QUBO objective function, the contribution of each node represents an improvement. Informally, we can imagine that the more sampling points we add in the domain the lower the total remaining variance would be. On the other hand $\beta_{i,j}$ values opposes to the selection of many sampling points.

In a configuration where $K + l$ nodes are selected the strongest contribution of values that are trying to deviate from a feasible solution with K sampling points is the following:

$$\underbrace{\sum_{\alpha_i \in \mathcal{A}_l} |\alpha_i|}_{\text{Contribution of additional nodes}} \quad \underbrace{- \left(\frac{(K+l)(K+l-1)}{2} - \frac{K(K-1)}{2} \right) \beta_{i,j}}_{\text{Contribution of additional edges}} \tag{9.12}$$

We can now compute an upper bound for this quantity as follows:

$$\begin{aligned} \sum_{\alpha_i \in \mathcal{A}_l} |\alpha_i| - \left(\frac{(K+l)(K+l-1)}{2} - \frac{K(K-1)}{2} \right) \beta_{i,j} &< \\ \sum_{\alpha_i \in \mathcal{A}_l} |\alpha_i| - 0 &\leq \\ l|\min(\mathcal{A})| &\leq \\ l^2|\min(\mathcal{A})| & \end{aligned} \tag{9.13}$$

To ensure that the energy penalties of the constraint are strong enough to overcome any configuration with more than K nodes selected, we need to impose Equations 9.8 and 9.9 to be higher than the upper bounded force applied by non-feasible configuration as computed in Equation 9.13, specifically:

$$\begin{aligned} \frac{Bl^2}{2} &> l^2|\min(\mathcal{A})| \\ B &> 2|\min(\mathcal{A})| \end{aligned} \tag{9.14}$$

Configuration with less nodes selected

Similarly to what we have done above, now we have to compute a bound for B such that the energy penalties of the constraint are strong enough to overcome any configuration with less than K sampling points.

Let $\mathcal{B}$ be the set of $\beta_{i,j}$ values computed with Equation 9.5. We can define $\mathcal{B}_n$ as the set of the n highest values in $\mathcal{B}$:

$$\mathcal{B}_n \triangleq \begin{cases} \mathcal{B}_0 = \emptyset \\ \mathcal{B}_n = \mathcal{B}_{n-1} \cup \max(\mathcal{B} \setminus \mathcal{B}_{n-1}) \end{cases} \tag{9.15}$$

Since each $\beta_{i,j}$ is a positive value and we want to minimize the QUBO objective function, in a configuration where $K - l$ nodes are selected the strongest contribution of values that are trying to deviate from a feasible solution with K sampling points is the following:

$$\underbrace{\sum_{\beta_{i,j}\in\mathcal{B}_{\left(\frac{K(K-1)}{2}-\frac{(K-l)(K-l-1)}{2}\right)}} \beta_{i,j}}_{\text{Contribution of removing edges}} - \underbrace{l|\alpha_i|}_{\text{contribution of removing nodes}} \tag{9.16}$$

where $\frac{K(K-1)}{2} - \frac{(K-l)(K-l-1)}{2}$ represents the number of extra edges if we select K sampling points as opposed to $K-l$. We can now compute an upper bound for this quantity as follow:

$$\begin{aligned}
\sum_{\beta_{i,j}\in\mathcal{B}_{\left(\frac{K(K-1)}{2}-\frac{(K-l)(K-l-1)}{2}\right)}} \beta_{i,j} - l|\alpha_i| &< \\
\sum_{\beta_{i,j}\in\mathcal{B}_{\left(\frac{K(K-1)}{2}-\frac{(K-l)(K-l-1)}{2}\right)}} \beta_{i,j} - 0 &\leq \\
\left(\frac{K(K-1)}{2} - \frac{(K-l)(K-l-1)}{2}\right)\max(\mathcal{B}) &= \\
\left(Kl - \frac{l^2}{2} - \frac{l}{2}\right)\max(\mathcal{B}) &= \\
\frac{l^2}{2}\left(\frac{2K}{l} - 1 - \frac{1}{l}\right)\max(\mathcal{B}) &< \\
\frac{2Kl^2}{2}\max(\mathcal{B}) & \qquad (9.17)
\end{aligned}$$

To ensure that the energy penalties of the constraint are strong enough to overcome any configuration with less than K nodes selected, we need to impose Equations 9.8 and 9.9 to be higher then the upper bounded force applied by non-feasible configuration as computed in Equation 9.18, specifically:

$$\begin{aligned}
\frac{Bl^2}{2} &> \frac{2Kl^2}{2}\max(\mathcal{B}) \\
B &> 2K\max(\mathcal{B}) \qquad (9.18)
\end{aligned}$$

Given the nature of the problem and the generality of the model that has to be valid for any possible hyperparameters combination of the Gaussian process, we cannot infer which one between Equations 9.14 and 9.18 imposes a bigger value of B. However, for any instance of the QUBO objective function we can easily compute a feasible value for B as follows:

$$B > \max\Big(2|\min(\mathcal{A})|\, ,\, 2K\max(\mathcal{B})\Big) \tag{9.19}$$

9.2.4 The complete model

The final QUBO model can easily be expressed as a complete weighted graph whose values are a combination of what presented in Section 9.2.1 and the constraint explained above. More in details, each node of the graph will have value:

$$a_i \triangleq \alpha_i - BK + \frac{B}{2} \tag{9.20}$$

and each edge of the graph will have value:

$$b_{i,j} \triangleq \beta_{i,j} + B \tag{9.21}$$

To conclude, the final QUBO model can be expressed as follows:

$$\begin{aligned} O(z_1, ..., z_n) = \sum_{i=1}^{|\mathcal{X}|} \Big(J(\{x_i\}) - J(\emptyset) - BK + \frac{B}{2}\Big) z_i + \\ \sum_{1 \le i < j \le |X|} \Big(J(\{x_i, x_j\}) - J(\{x_i\}) - J(\{x_j\}) + J(\emptyset) + B\Big) z_i z_j \end{aligned} \tag{9.22}$$

with B computed as described by Equation 9.19.

9.2.5 Optimized variant

Notice that values $\beta_{i,j}$ computed with Equation 9.5 represent the difference between the variance reduction obtainable with two sampling points acquired at the same time and the sum of the variance reduction obtainable with the two points acquired individually. However, $\beta_{i,j}$ is not taking into account how the real variance of a Gaussian process is affected by the presence of additional measurement points.

In general, the posterior variance of a Gaussian process is monotonically decreasing with the number of sampling points. Hence, to better represent the real variance reduction of a Gaussian process in case we have more than 2 sampling points, the $\beta_{i,j}$ values should be lower than what computed with Equation 9.5.

For this reason we now propose a variant of the model where we multiply by a weight value w each $\beta_{i,j}$ computed with Equation 9.5. This w multiplier is intended as a scaling factor for the difference in the variance reduction, specifically:

$$\begin{aligned} \beta_{i,j} &\triangleq w\big(J(\{\mathbf{x}_i, \mathbf{x}_j\}) - J(\emptyset) - \alpha_i - \alpha_j\big) \\ &= w\big(J(\{\mathbf{x}_i, \mathbf{x}_j\}) - J(\{\mathbf{x}_i\}) - J(\{\mathbf{x}_j\}) + J(\emptyset)\big) \end{aligned} \tag{9.23}$$

By setting $0 < w \leq 1$ we can better approximate the real variance of a Gaussian process for cases where we have 3 or more sampling points. We will show in the empirical evaluation (Section 9.3) that this additional parameter allows us to obtain better results.

9.3 Empirical Evaluation

In what follows we present the results of the empirical evaluation of our QUBO model for Gaussian process posterior variance reduction. Notice that we are not proposing an optimization method for quadratic unconstrained binary problems, instead we want to show that the QUBO objective function represented by our model is a good approximation for the problem of minimizing the posterior variance as explained in Section 9.1.1 Equation 9.2. The main objectives of this empirical evaluation are:

1. Test the model under different conditions, specifically:
   - Using different hyperparameters of the Gaussian process to show the generality of the approach (similarly to what we show in Chapters 7 and 8).
   - Varying the number of sampling points K to study the performance of the QUBO model.
2. Show a comparison between the optimal solution of the QUBO model with respect to submodular optimization.
3. Show a comparison between the optimal solution of the QUBO model with respect to a random sampling solution (used as a simple baseline technique).

9.3.1 Dataset and setup

In this empirical evaluation we use two different 2-dimensional cubic datasets with equally distributed domain points $\mathcal{X}$ similarly to what presented in Section 7.3 and Figure 7.1. However, since in this case we have to compute the optimum of the QUBO objective function, the cardinality $|\mathcal{X}|$ (number of points on which we evaluate the Gaussian process) of the datasets needs to be widely reduced. Specifically, our two datasets are composed by 25 and 36 domain points respectively.

We tested our model by training the Gaussian process using the squared exponential kernel reported here for convenience:

$$k(\mathbf{x}, \mathbf{x}') = \sigma_f^2 \exp\left(- \frac{(\mathbf{x} - \mathbf{x}')^T(\mathbf{x} - \mathbf{x}')}{2l^2} \right) \tag{9.24}$$

Regarding the hyperparameters of the kernel, in our tests we used two different length-scale l, two different σ_f and two different σ_n. As previously mentioned throughout the thesis, l describes the smoothness property of the true underlying function, σ_f describes the standard deviation of the modeled function and σ_n the standard deviation of the noise of the observations (see Section 3.1.1 for more details). These hyperparameters gives us a total of 8 different combinations to test the model under different conditions.

For the optimized variant of the model as presented in Section 9.2.5, we performed experiments using 19 different w parameters, specifically from 0.1 up to 1 with steps of 0.05. Notice that using $w = 1$ corresponds to the standard version of the model as described by Equation 9.22.

To run the experiments and compute the optimum value of the QUBO objective function we used CPLEX optimization library. For more details on the optimization using CPLEX please refer to Appendix B.

All the above mentioned combinations of hyperparameters and datasets have been tested by adapting a different number K of measurement points which varies from 2 up to 7. The case of a single point has been excluded since the submodular greedy technique is optimal by definition.

Moreover, regarding the comparison with a random procedure, for each of the described combination of hyperparameters, dataset and number of sampling points K, we have generated 100 randomly selected solutions.

9.3.2 Results

Figures 9.1 and 9.2 show the aggregated results of the experiments previously described for the dataset with 25 and 36 domain locations respectively. In these plots we can observe the total remaining variance of the Gaussian process (in logarithmic scale for the sake of representation) by varying the number K of sampling points. Lines in these charts represent the average over the eight combinations of hyperparameters used in the experiments and the error bars represent the standard errors of the mean. The only exception concerns the line that represent the random technique which is the average over 8×100 experiments (8 combination of hyperparameters and 100 randomly selected combinations of sampling points).

First of all we notice that in general the optimized variant of the QUBO model (presented in Section 9.2.5) with the strength of the quadratic terms modulated by the w parameter, allows us to obtain much better results compared to the 'standard' QUBO model (which corresponds to use $w = 1$), proving that indeed a good tuning of that parameter provides an advantage for our QUBO model.

Moreover, we can observe that the optimum solutions of our QUBO model (by tuning the w parameter) in the first dataset (Figure 9.1) are comparable with submodular selection technique when using 2 sampling points, worst with 3 and better in all the other cases. A similar situation

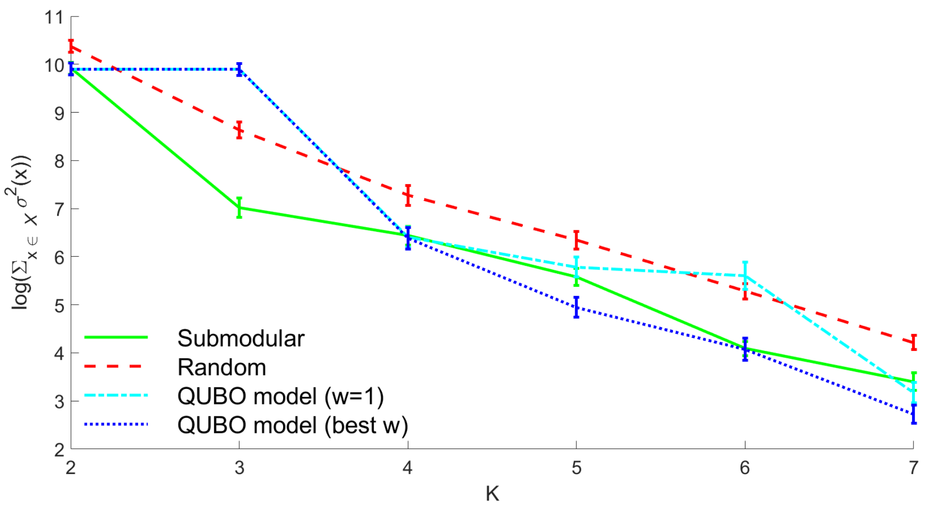


Figure 9.1: Posterior variance of the Gaussian process by varying the number K of sampling points on the dataset with a domain composed of 25 locations. The results represent the average over the 8 combination of hyperparameters used during the experiment and the error bars represent the standard error of the mean.

is true for the second dataset (Figure 9.2) where we have in some cases a comparable results, in some worst and in the remaining better results than submodular. On average the solutions obtained with a random selection of sampling points perform worst than both submodular and the QUBO model.

As we can observe for both the datasets (Figures 9.1 and 9.2) the trend for all the techniques tested is the same. As expected, by adding more measurement locations the variance of the Gaussian process decreases. However, the interleaving of the curve representing our QUBO model and the curve representing the submodular selection shows that the optimum of the QUBO model represents indeed a good approximation of the objective function of our problem.

9.4 Conclusions

In this chapter we proposed a novel QUBO model to tackle the problem of optimizing sampling locations in order to minimize the posterior variance of a Gaussian process. The strength of this contribution is the proposal of a completely alternative method that can be used by non-classical computing architectures (quantum annealer) and therefore benefit from research in this field.

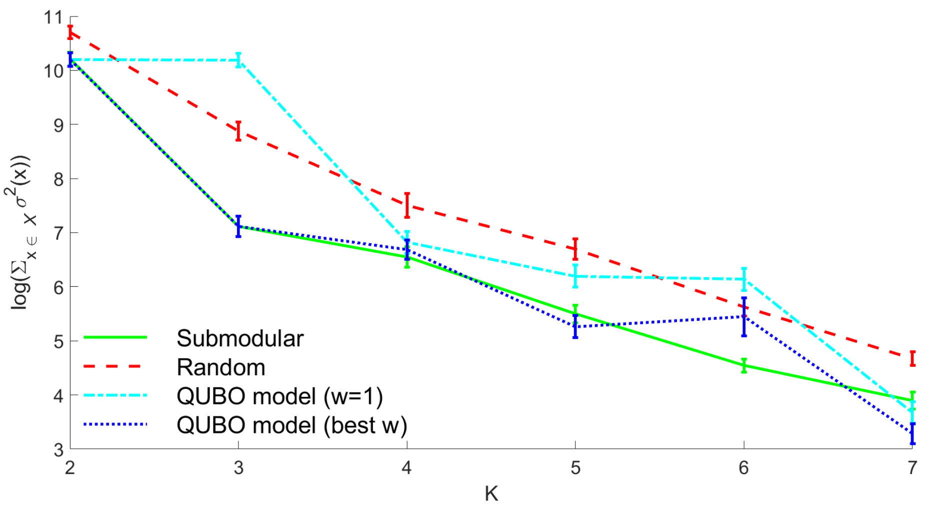


Figure 9.2: Posterior variance of the Gaussian process by varying the number K of sampling points on the dataset with a domain composed of 36 locations. The results represent the average over the 8 combination of hyperparameters used during the experiment and the error bars represent the standard error of the mean.

Although the w parameter of our model has to be determined empirically, results shows that the optimum of the QUBO objective function represents a good solution for the above mentioned problem, obtaining comparable and in some cases better results than the widely used submodular technique. Moreover, values $\beta_{i,j}$ as computed in Equation 9.5 could be replaced with values computed with different techniques such as the one proposed in our heuristic in Chapter 8.

Nevertheless, we believe that our contribution with this QUBO model takes an important first step towards the sampling optimization of Gaussian processes in the context of quantum computation.

Chapter 10

QUBO for biclustering

10.1 Introduction

As we discussed in the previous chapter, quantum information processing could provide a viable alternative to the optimization process in artificial intelligence and pattern recognition problems. In what follow we will tackle the problem of biclustering (introduced in Section 3.6.3).

While clustering has been widely used to find patterns in data, the use of biclustering (sometimes called co-clustering or sub-space clustering) is more recent and has been used mainly to analyze biological data and particularly microarray gene data. However, recently, biclustering has been applied to the analysis of data acquired by mobile platforms for environmental monitoring applications [44]. In this context, the goal is usually to identify situations of interest (e.g., mobile platform moving upstream or downstream) using a minimal set of features.

Similarly to what proposed in Chapter 9 here we formulate a quadratic unconstrained binary optimization (QUBO) model for the biclustering problem. The contribution of this work can be summarized as follows:

- We introduce the first QUBO model for the biclustering problem. More specifically, we formulate the biclustering problem as a repeated search for the most coherent biclusters following well-known approaches [16, 47] where biclusters are extracted one at a time from the datamatrix (one-bicluster problem).
- We analyze the model and give mathematical proofs of its correctness, i.e., that the optimal solution of the QUBO model is the optimal solution for the one-bicluster problem.
- We perform an empirical evaluation showing that our model outperforms in terms of quality the state-of-the-art biclustering approaches (i.e., BICRELS [172] and FLOC3 [184]).

- We discuss the practical applicability of our model by means of experiments performed on the the D-Wave 2X$^{\text{TM}}$ architecture.

Overall, our key contribution is a QUBO formulation for the biclustering problem that can be solved by quantum annealing machines. Our investigation on the D-Wave quantum annealer shows that such QUBO model is a viable approach for small-sized data matrices and the proposed principles might be used as a foundation to tackle larger datasets.

10.2 The QUBO model for biclustering

As already mentioned, biclustering has been used in various application domains with different techniques. In its most general form the biclustering problem can be defined as the simultaneous clustering of rows and columns of a given data-matrix [120]. The goal is then retrieving the subsets of rows and columns that have a coherent behavior, where "coherence" is defined according to the specific application domain (e.g., Euclidean distance, Pearson correlation).

In our contribution, we formulate the problem of biclustering as a sequential search for the most coherent bicluster. This is a widely employed technique in the literature [16, 47, 61], and consists in extracting biclusters one by one from the data-matrix (we call it one-bicluster problem). Once we extract a bicluster we then need to "mask" it in the data matrix before looking for the next one. There exist different heuristics in the literature addressing this problem: for example, one way to address this problem is to replace the obtained bicluster with background noise in the original data matrix [47].

In this section we focus only on the problem of extracting one bicluster and we detail our QUBO formulation for this problem. In particular, we first describe a binary model for the one-bicluster problem and then, we show how such a model can be encoded as a QUBO.

10.2.1 A binary model for one-bicluster

We now present the objective function for the binary one-bicluster problem and in what follows we explain how it is derived. Given a real-valued data matrix A with N rows and M columns, the objective function for the binary one-bicluster problem is the following:

$$\operatorname*{arg\,max}_{(c_{1,1},\dots,c_{N,M})} \left(\sum_{i,j} a_{i,j} c_{i,j} - \sum_{i,j,t,k} O_{i,j,t,k} c_{i,j} c_{t,k} + \sum_{i<t} B_{i,t} \right) \tag{10.1}$$

where $1 \le i, t \le N$; $1 \le j, k \le M$.

In the first two terms, we have NM binary variables $c_{i,j}$ that encode whether a given entry $a_{i,j}$ of the data matrix A belongs to the bicluster or not (where $c_{i,j} = 1$ indicates that the entry $a_{i,j}$ does belong to the bicluster). In this equation, we can identify two forces:

- One that encourages points to group together, namely the first term in (10.1).
- One that avoids points that are not coherent to be in the same group, namely the second term in (10.1). Such term is based on a value $O_{i,j,t,k}$ which measures the coherence between two points $a_{i,j}$ and $a_{t,k}$.

The function $O_{i,j,t,k}$ depends on which kind of biclusters we wish to analyze. In particular, following the relevant literature (e.g., [173]) we consider two types of coherence:

Constant: Which aims at penalizing points that have a different activation level and hence identifies biclusters that have a single coherent value.

$$O_{i,j,t,k} = w|a_{i,j} - a_{t,k}| \tag{10.2}$$

Additive: Which identifies biclusters that encode an evolution of the activation values over columns.

$$O_{i,j,t,k} = w(a_{i,j} - a_{t,j} + a_{t,k} - a_{i,k})^2 \tag{10.3}$$

In both equations (10.2) and (10.3), the weight w can be adjusted to balance such two forces: setting w to high values favors the coherence of the points inside the biclusters, while setting w to low values favors the creation of large biclusters. The set of valid values for such weight is $\mathbb{R}^+$; however, setting high values could lead as a result to biclusters composed of a single element. The appropriate value to set depends on the data context and must be determined experimentally as shown in [61].

In order to solve our problem, we need to restrict the feasible variable assignments so that only valid assignments correspond to a bicluster. In other words, we need to rule out assignments that do not correspond to a subset of rows and columns that have all entries selected (see Figure 10.1b for an example of a non-valid assignment). To do so, we add one constraint stating that, given two rows of the output matrix C, they have to share the same configurations or one of them must be zero. The constraint between rows i and t is expressed in equation (10.1) by the term:

$$B_{i,t} = \begin{cases} 0, & \text{if } (\sum_k c_{i,k} = 0) \vee (\sum_k c_{t,k} = 0) \\ & \quad \vee (\sum_k (c_{i,k} - c_{t,k}) = 0) \\ -\infty, & \text{otherwise} \end{cases} \tag{10.4}$$

Such constraint ensures that there is a permutation of rows and columns that forms a sub-matrix with all entries selected (i.e., visually a full rectangle of ones).

Another interesting way to look at an admissible configuration is that it can be described by fixing the same value for all the elements of a column with an exception for the elements that belong to a disabled row. For example, considering Figure 10.1a (before permutations), the configuration can be expressed as: Columns $\{1, 3, 4\}$ take value 1, columns $\{2, 5\}$ take value 0 and row 2 is disabled (all the element are 0). Hence, any admissible configuration can be uniquely identified by this type of description. This description is useful to better understand the QUBO model we describe next.

$$C = \begin{pmatrix} 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 \end{pmatrix} \longrightarrow \begin{pmatrix} 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

(a)

$$C = \begin{pmatrix} 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 \end{pmatrix}$$

(b)

Figure 10.1: Example of: a valid assignment and its permutation that results in a full rectangle of ones (10.1a); an invalid assignment, no permutation can result in a full rectangle of ones (10.1b)

10.2.2 The QUBO model for the one-bicluster problem

In this section we provide a QUBO formulation for the binary model described above. Similarly to the presentation proposed in Chapter 9, we detail the description of the model by providing initially a version that does not considers the constraints required by the problem and subsequently we describe how we add energy penalties that implements them. For this reason, let us start with a QUBO representation that does not consider the bicluster constraint (i.e., the $B_{i,t}$ elements in equation (10.1)).

To build such model by using the graph-based representation of QUBOs, we create a node $x_{i,j}$ for each variable $c_{i,j}$. Considering that the QUBO formulation has to be minimized, we then assign a coefficient $-a_{i,j}$ to each node. For each pair of nodes $(x_{i,j}, x_{t,k})$ we assign to the edge between them a positive value $O_{i,j,t,k}$ calculated according to the equations (10.2) or (10.3). Note that the latter has value 0 for points on the same row or the same column, hence for such measure, the horizontal and vertical edges are absent from the graph. The corresponding objective function for the QUBO problem will then be:

$$\underset{(x_{1,1},\ldots,x_{N,M})}{\arg\min}\left(\sum_{i,j} -a_{i,j}x_{i,j} + \sum_{i,j,t,k} O_{i,j,t,k}x_{i,j}x_{t,k}\right) \tag{10.5}$$

where $1 \leq i,t \leq N$; $1 \leq j,k \leq M$. It is easy to see that the assignment that maximizes function (10.1) without the bicluster constraint is the same that minimizes the QUBO objective function (10.5). Figure 10.2 shows a graphical representation of such a simplified QUBO model for a 2×2 input data matrix.

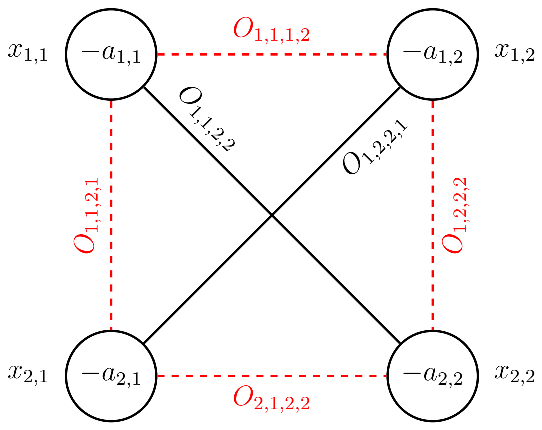


Figure 10.2: A graphical representation of our QUBO model for a 2×2 data-matrix, the (red) dotted edges are absent in case of additive coherence measure (10.3).

Now, in order to consider the bicluster constraint, we must add some extra nodes to the QUBO model so as to ensure that the assignments generated are valid (i.e., they represent a subset of rows and columns). As mentioned in Section 10.2.1 an admissible configuration should set all variables in the same column to the same value except for the variables that belong to disabled rows. To express this, we create two types of constraints: column constraints and row constraints.

A column constraint ensures that all variables in a column have the same value (either 0 or 1). To do so, we add to each node a positive coefficient V and we add a new node to the graph with a coefficient equal to $N(B-V)$ where $B > V$. We call this new node the *column switch* and we indicate with s_j the variable that corresponds to the node switch for column j. Finally, we set the coefficient of the edges between the column switch and the N nodes to $-B$ (see Figure 10.3 for a graphical representation). Intuitively, if k of the N nodes are selected and the switch is not active (i.e., $s_i = 0$), we add to the objective function a value kV. If we select the switch and the k nodes, we add $k(V - B) + N(B - V)$. Since we are minimizing the objective function the best configuration will be either selecting all nodes (with a contribution of $N(V - B) + N(B - V) = 0$) or not selecting any node (again with a contribution of zero). All other configurations will add a positive value to the objective function.

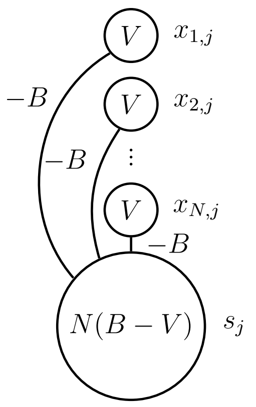


Figure 10.3: Graphical representation of a column constraint.

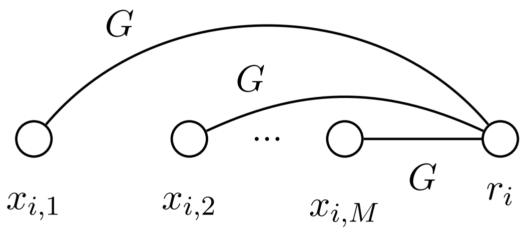


Figure 10.4: Graphical representation of a row constraint.

A row constraint should force all variables in a row to be zero when a specific condition holds (i.e., we decide to not consider that row). To enforce this, we add a new node to the graph with a coefficient 0 and we call this new node the *row switch*. We indicate with r_i the variable that corresponds to the node switch for row i (see Figure 10.4 for a graphical representation). Then, we set the edges between the row switch and the M nodes to a positive coefficient G. Intuitively, when the $r_i = 0$ any configuration for the M nodes contributes with a null value to the objective function; hence, they are equally desirable. However, if $r_j = 1$, then selecting any of the M nodes will increase the objective function of a value G. Hence, in this case, the best configuration is the one that does not select any of the M nodes.

Finally we combine the first graph (Figure 10.2) without the bicluster

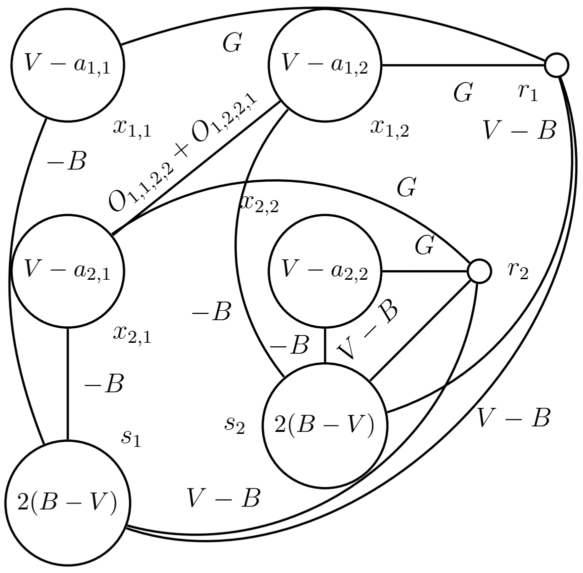


Figure 10.5: Graph of the complete model for $N = 2$ and $M = 2$ with the additive coherence similarity metric (equation (10.3)) and the simplification proposed at the end of this section.

constraint (from now on called the *inner graph*) with the row and column constraints and by adding from each row switch to every column switch an edge with coefficient $V - B$. The objective function has now the following form:

$$\begin{aligned}\underset{(x_{1,1},\ldots,x_{N,M})}{\arg\min} \sum_{i,j} \Big(& V x_{i,j} - B x_{i,j} s_j + G x_{i,j} r_i \\ & + (V - B) r_i s_j + (B - V) s_j - a_{i,j} x_{i,j} \\ & + \sum_{t,k} O_{i,j,t,k} x_{i,j} x_{t,k} \Big)\end{aligned} \tag{10.6}$$

Notice that in principle to ensure that a set of nodes are either all O or 1 an additional variable (column switch) is not necessary. However, in this case the column switches combined with the row switches with the additional introduce term $(V - B) r_i s_j$ are convenient in order to ensure that our QUBO formulation is a proper model for the one-bicluster problem.

Specifically, now we can show that for all valid solutions, the extra constraints (i.e., row and column constraints) contribute with a zero value, while for all non-valid solutions they contribute with a strictly positive value. In particular, we prove the following theorem:

Theorem 3 (Model validity)**.** *Given a model of a data-matrix with N rows and M columns and values $B > V > 0$ and $G > B - V$, for all assignments* *that do not violate a row or a column constraint such extra constraints provide a null contribution to the objective function. For all other configurations, the contribution is* > 0.

Proof. Given the objective function in equation (10.6), we can observe that in each addend of the summation the terms that depend from the combined constraint structure are:

$$Vx_{i,j} - Bx_{i,j}s_j + Gx_{i,j}r_i + (V-B)r_is_j + (B-V)s_j. \tag{10.7}$$

Hence, each of these addends depend exclusively on three binary variables, namely a node from the inner graph $x_{i,j}$ and the two switches r_i and s_j. Now we compute the value of the term for the combined constraint structure equation (10.7) exhaustively for all eight cases of the three variables:

1. $[x_{i,j} = 0,\ r_i = 0,\ s_j = 0]$: 0
2. $[x_{i,j} = 0,\ r_i = 0,\ s_j = 1]$: $B - V$
3. $[x_{i,j} = 0,\ r_i = 1,\ s_j = 0]$: 0
4. $[x_{i,j} = 0,\ r_i = 1,\ s_j = 1]$: $V - B + B - V = 0$
5. $[x_{i,j} = 1,\ r_i = 0,\ s_j = 0]$: V
6. $[x_{i,j} = 1,\ r_i = 0,\ s_j = 1]$: $V - B + B - V = 0$
7. $[x_{i,j} = 1,\ r_i = 1,\ s_j = 0]$: $V + G$
8. $[x_{i,j} = 1,\ r_i = 1,\ s_j = 1]$: $V - B + G + V - B + B - V = V - B + G$

For cases 1,3,4,6 which represent a valid assignment where all the inner graph nodes are in compliance with the switches (i.e., do not violate row or a column constraints), the contribution is 0. For all the other configurations which represent a non-valid assignment, the contribution is greater that 0 (this is because $B > V > 0$ and $G > B - V$). □

In order to complete the model, we have to identify the appropriate values for V, B and G. To do so, we observe that a configuration that does not comply with all the switches constraints should increase more than the decrease in value that can derive from taking such a configuration in the inner graph, namely the values assigned to the structure should be high enough to ensure that the objective function does not minimize for the non-valid configurations. Although intuitively we can simply choose high values, to maintain the range of possible values as small as possible, we investigate what the lowest admissible ones are. Let us indicate with R a configuration for the row switches, S a configuration for the column switches, X a configuration for the inner graph nodes in compliance with the switches and $\overline{X}$ a configuration where any subset of X does not comply with the corresponding switches.

We can then show the following theorem:

Theorem 4 (Determining V, B, G)**.** *Given the specific switches configurations R and S and the valid solution (X, R, S), we have that:*

$$O(\overline{X}, R, S) - O(X, R, S) > 0 \\ \iff \\ \big(V > V_m \wedge B > B_m \wedge G > G_m\big) \tag{10.8}$$

for all invalid solutions $(\overline{X}, R, S)$, where

$$\begin{aligned} V_m &= \max_{i,j}\{a_{i,j}\} \\ B_m &= V + \max_{i,j}\Big\{ - a_{i,j} + \sum_{t,k} O_{i,j,t,k}\Big\} \\ G_m &= B - V + \max_{i,j}\{a_{i,j}\} \end{aligned} \tag{10.9}$$

Proof. Similarly to what we did for Theorem 1, we now compute the value of equation (10.6) for all configurations of the three binary variables $x_{i,j}$, r_i and s_j:

1. $[x_{i,j} = 0,\ r_i = 0,\ s_j = 0]$: 0
2. $[x_{i,j} = 0,\ r_i = 0,\ s_j = 1]$: $B - V$
3. $[x_{i,j} = 0,\ r_i = 1,\ s_j = 0]$: 0
4. $[x_{i,j} = 0,\ r_i = 1,\ s_j = 1]$: $V - B + B - V = 0$
5. $[x_{i,j} = 1,\ r_i = 0,\ s_j = 0]$: $V - a_{i,j} + \sum_{t,k} O_{i,j,t,k} x_{t,k}$
6. $[x_{i,j} = 1,\ r_i = 0,\ s_j = 1]$: $V - B + B - V - a_{i,j} + \sum_{t,k} O_{i,j,t,k} x_{t,k} = - a_{i,j} + \sum_{t,k} O_{i,j,t,k} x_{t,k}$
7. $[x_{i,j} = 1,\ r_i = 1,\ s_j = 0]$: $V + G - a_{i,j} + \sum_{t,k} O_{i,j,t,k} x_{t,k}$
8. $[x_{i,j} = 1,\ r_i = 1,\ s_j = 1]$: $V - B + G + V - B + B - V - a_{i,j} + \sum_{t,k} O_{i,j,t,k} x_{t,k} = V - B + G - a_{i,j} + \sum_{t,k} O_{i,j,t,k} x_{t,k}$

In order to ensure the desired behavior, the difference between a non eligible configuration $(\overline{X}, R, S)$ and an eligible configuration (X, R, S) must be higher than 0. Let us impose this condition to the difference between the previous eight cases:

- **[5]**−**[1]** $> 0 \Rightarrow V - a_{i,j} + \sum_{t,k} O_{i,j,t,k} x_{t,k} > 0$
- **[7]**−**[3]** $> 0 \Rightarrow V + G - a_{i,j} + \sum_{t,k} O_{i,j,t,k} x_{t,k} > 0$
- **[8]**−**[4]** $> 0 \Rightarrow V - B + G - a_{i,j} + \sum_{t,k} O_{i,j,t,k} x_{t,k} > 0$
- **[2]**−**[6]** $> 0 \Rightarrow B - V + a_{i,j} - \sum_{t,k} O_{i,j,t,k} x_{t,k} > 0$

Because the coherence measure $O_{i,j,t,k}$ is always greater or equal to 0, we are now ready to determine the minimum value to assign to V, B and G. From the first difference [5]−[1], we have:

$$V > \max_{i,j}\{a_{i,j}\} = V_m$$

From the last difference [2]−[6] we have that:

$$B > V + \max_{i,j}\left\{ - a_{i,j} + \sum_{t,k} O_{i,j,t,k} \right\} = B_m$$

And from the third one [8]−[4] we have:

$$G > B - V + \max_{i,j}\{a_{i,j}\} = G_m$$

The second one [7]−[3] holds because of V and G already defined. □

10.2.3 Properties of the model

Theorems 3 and 4 ensure that, by building the model as described above, for any valid configuration (i.e., a configuration that describes a bicluster), the contribution of the column and row constraints to the objective function is null. For all valid assignments the objective function reported in equation (10.6) reduces to equation (10.5), hence the configuration that minimizes (10.6) is the same that maximizes equation (10.1) (i.e., the most coherent bicluster). Moreover, for any non-valid assignment (i.e., an assignment that does not encode a bicluster) the contribution of the row and column constraints will be strictly positive hence such configuration will always be discarded in favor of a valid assignment.

The proposed model can be further simplified. In particular, we can reduce the number of edges (quadratic terms) by observing that if a couple of nodes (in the inner graph) on different rows and columns are active (i.e., two nodes on the opposite corners of a rectangle) also the other two nodes on the other diagonal of the rectangle must be active to comply with the switches. The terms $O_{i,j,t,k} x_{i,j} x_{t,k}$ and $O_{t,j,i,k} x_{t,j} x_{i,k}$ either contribute both or none to the objective function. Hence, we can add both values $O_{i,j,t,k} + O_{t,j,i,k}$ to a single edge and remove the other one. Hence, regardless of the coherence measure used, we can remove half of the diagonal edges. An example of the complete simplified model is shown in Figure 10.5.

As for space complexity, given an input matrix $N \times M$, the model has $NM + N + M$ binary variables. The number of edges depends on the coherence metric used. In particular, for the constant coherence equation (10.2), we have in the worst case (i.e., when all the coherence measures are different from 0) $NM(NM - 1)/2 - NM(N - 1)(M - 1)/4 + 3NM$ edges. For the additive coherence equation (10.3), we must insert into the model only the diagonal edges (see Figure 10.5); hence, the total number of edges, in the worst case, is $NM(N - 1)(M - 1)/4 + 3NM$.

Our main motivation for this contribution is to investigate the possibility to exploit the quantum annealing process to solve the biclustering problem. Based on the above analysis, the main computational bottleneck for our model is space requirements. While the worst-case analysis reveals a polynomial complexity for what concerns space, typical application domains for biclustering can involve data matrix with a large number of rows and columns (i.e., thousands of genes and hundreds of experiments). For such numbers, the space requirement for our model becomes an issue that needs an adequate treatment. To this purpose, in the next section, we also propose a sparsification method in order to simplify the model by eliminating a given percentage of edges using a heuristic. Moreover, following previous approaches [62], we use a decomposition technique in order to aggregate biclusters extracted from sub-matrices.

10.3 Empirical evaluation

Having described and analyzed QUBO model for biclustering, here we present its empirical evaluation. In what follows we first describe the methodology we used to perform the experiments and then we present the results obtained by following established evaluation protocols for biclustering techniques [173].

10.3.1 Evaluation Methodology

The main goals of our empirical evaluation are:

1. Validate the accuracy of the QUBO model for biclustering comparing it with state-of-the-art approaches (BICRELS [172] and FLOC3 [184]).

2. Evaluate how the removal of edges from the model affects the quality of the solutions.

3. Evaluate the quality of our model through a widely exploited biclustering dataset [146].

4. Assess the applicability of the model on current state-of-the-art quantum processing units (i.e., the D-Wave architecture).

To this aim, we created a synthetic dataset so as to accurately measure the performance of our approach. In particular, the dataset is composed of 10×10 matrices with a constant random-positioned bicluster that occupies the 25 percent of the elements. Then, we added a Gaussian noise to each matrix, where the standard deviation of such Gaussian noise is a percentage of the difference between the mean of the entries belonging to the biclusters and the mean of all the others. In particular, we considered 5 different percentage values from 0 (no noise) to 0.2. We generated a set of 15 matrices per noise level for a total of 75 matrices. This dataset allows us to measure the accuracy of the algorithms by comparing the bicluster extracted from the models with the ground-truth (i.e., the bicluster that is present in the data-matrix). Given C the set of elements of the bicluster found and L the set of elements of the real bicluster, to measure such accuracy we use two established metrics [173]:

- $Purity = |C \cap L|/|C|$ which represents how many elements of the solution belong to the real bicluster.

- $InversePurity = |C \cap L|/|L|$ which represents how many elements of the real bicluster have been found.

10.3.2 Validating the accuracy of the QUBO model

For each of the 75 matrices of the dataset, the QUBO form has been solved by using the CPLEX optimization library and by applying 24 different weights w (constant for each $O_{i,j,t,k}$) to the similarity measure (equation (10.3)), for a total of 1800 tests.

Here we present the results of Purity and Inverse Purity as a function of the noise level. For each noise level we analyzed and set the parameters of the procedures with the values that gives the best average result on the 15 matrices with that noise level. Please note that the optimal value of w, which influences the size of the ideal biclusters, depends on the data context and has to be determined empirically. Solving each instance takes milliseconds; hence, the overhead to determine the optimal value of w is not an issue. Note that this is the same protocol used in [61]. Results in Figure 10.6 show that our QUBO model significantly outperforms BICRELS and FLOC3 in terms of quality of the bicluster extracted.

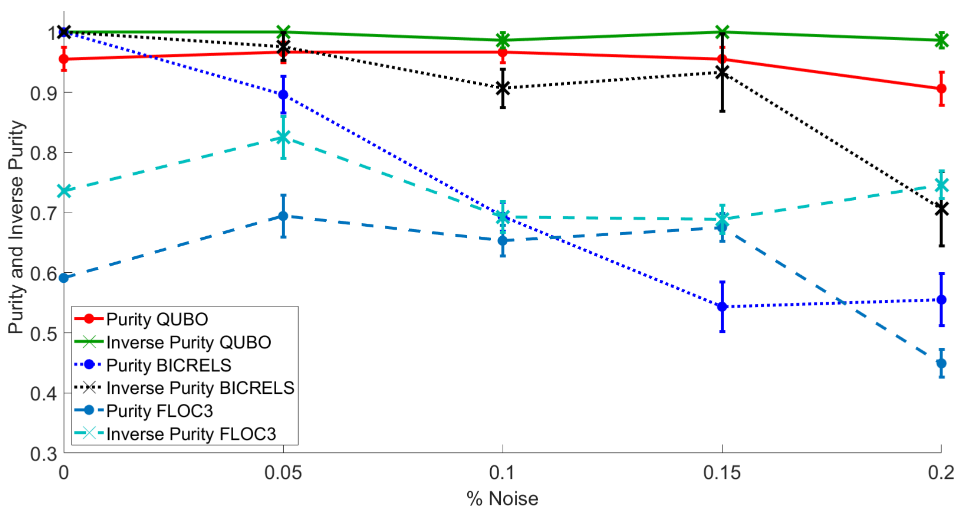


Figure 10.6: Mean performance over the 15 matrices for each noise level of our QUBO model, BICRELS and FLOC3. The error bars represent the standard error of the mean.

10.3.3 Sparsification of the model

We can observe that our model exhibits some degree of redundancy. In particular not all the edges in the 'inner graph' (that represent the similarity measurements between points of the input matrix), affect with the same weight the selection of the optimal solution. For example, assume we know the sub-matrix that forms the most coherent bicluster, intuitively many of the edges internal to such sub-matrix will have a low value (because the elements of the bicluster are coherent). As a consequence, most of such edges could be removed from the model. For this reason, we tested how the model behaves by removing different percentages of edges from the inner graph of the 1800 instances previously described. Specifically, we removed from 0 to 90% (with steps of 10%) of the edges for a total of 18,000 tests.

Note that sparsification is only intended to be a practical heuristic to address larger matrix, but we cannot provide any guarantees on how this affect optimality. In contrast, our aim is to investigate whether simple sparsification heuristics could maintain a good level of accuracy while providing significant reductions in space for the model. While assessing which edges are redundant (without knowing the bicluster) is not straightforward, the empirical evaluation shows that some simple heuristics do provide a significant gain.

In more detail, our procedure for sparsification computes a value for each internal edge and then sort edges according to such value. We then remove a specific percentage X (with $X \in \{0, 10, 20, \ldots, 90\}$) of these edges. We tried different values for the edges that are all based on a combination of the function $O_{i,j,t,k}$ and the values of the matrix entries that relates to this function (i.e., $a_{i,j}$, $a_{t,k}$, $a_{i,k}$, $a_{t,j}$).

The different heuristics we tested are:

- **Edge value sparsification:** In this heuristic we eliminate edges from the inner graph by choosing the desired percentage with the lowest value, that is the edges between nodes with the lowest additive coherence measure $O_{i,j,t,k}$ calculated according to equation (10.3).
- **Inverse edge value sparsification:** This heuristic is the inverse of the previous one. Here we eliminate edges from the inner graph by choosing the desired percentage with the highest value, that is the edges between nodes with the highest additive coherence measure $O_{i,j,t,k}$ calculated according to equation (10.3).
- **Ratio sparsification:** In this heuristic we eliminate edges from the inner graph by choosing the desired percentage with the lowest ratio $\frac{O_{i,j,t,k}}{a_{i,j} + a_{t,k} + a_{i,k} + a_{t,j}}$, where $O_{i,j,t,k}$ is the additive coherence measure calculated according to equation (10.3).
- **Inverse ratio sparsification:** This heuristic is the inverse of the previous one. Here we eliminate edges from the inner graph by choosing the desired percentage with the highest ratio $\frac{O_{i,j,t,k}}{a_{i,j} + a_{t,k} + a_{i,k} + a_{t,j}}$, where $O_{i,j,t,k}$ is the additive coherence measure calculated according to equation (10.3).
- **Random sparsification:** We used as an additional comparison baseline a random sparsification where we remove a specific percentage X (with $X \in \{0, 10, 20, \ldots, 90\}$) of the edges.

Our results confirm that with these simple heuristics it is possible to achieve similar level of accuracy with approximately half the edges of the QUBO model. Overall the 'Inverse ratio sparsification' shows the better performance. Figure 10.7 reports a comparison of the 'Inverse ratio sparsification' heuristic against the random approach.

10.3.4 Evaluation on benchmarking data-set

We evaluated our model on the benchmarking synthetic dataset introduced in [146].[1] The matrices proposed in that dataset contains 100×50 entries. Such matrices cannot be directly analyzed by our approach due to the space complexity associated with our model (see Section 10.2.3). However, following previous approaches [62], we can extract biclusters from sub-matrices

[1]Available at http://www.tik.ee.ethz.ch/sop/bimax (Scenario I – Noise)

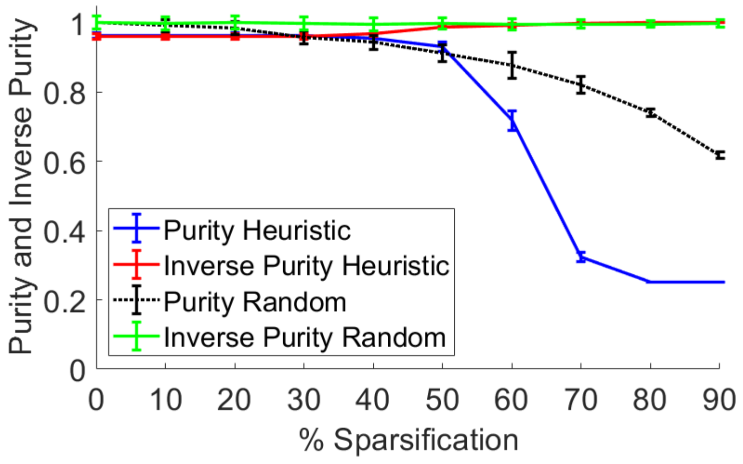


Figure 10.7: Average of Purity and Inverse Purity varying noise level for both random and heuristic sparsifications described in Section 10.3.3. Error bars represents the standar error of the mean.

and then aggregate the results. In particular, in our experiments, we consider a 10×10 window that selects a portion of the data matrix and we shift this windows over the data with a full coverage and an overlap degree of 5 rows/columns. We call each sub-matrix a *kernel*. The proposed protocol consists of the following three steps:

1. *Generate the bicluster set.* We extract one bicluster from each kernel using the additive coherence.

2. *Aggregate the results.* We group the biclusters provided by step 1 by using a similarity based clustering algorithm (Affinity Propagation [73]). We defined as similarity between two biclusters the number of rows/columns they share.

3. *Retrieve the final bicluster.* Please notice that the coherency in biclusters obtained at the previous step is not guaranteed. For this reason, we assign to each bicluster a score, exploiting the objective function (equation 10.1), i.e., evaluating the objective function for such bicluster. This step is repeated for all groups obtained in step 2 and by keeping the best solution (according to the objective function) we keep the most coherent solution.

The accuracy of the resulting biclusters has been assessed with the same metrics used in [146], i.e., the *Gene Match Score* which is a measure of similarity between the retrieved biclusters and the real ones. Results in Figure 10.8 shows that our method is competitive with other state-of-the-art approaches (see Figure 2a in [146]), confirming the potentials of the proposed approach.

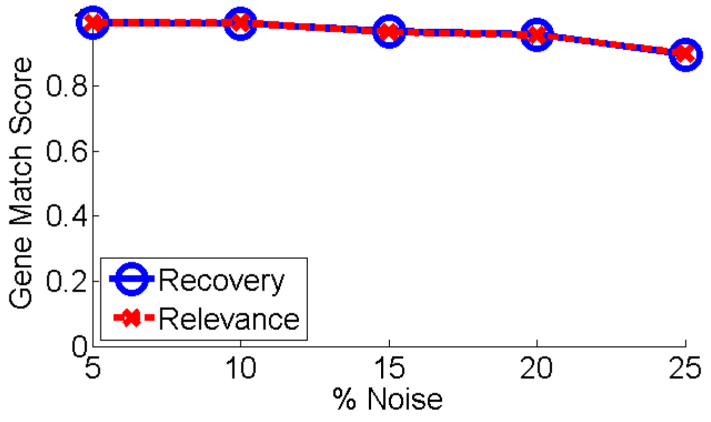


Figure 10.8: Performance evaluation on benchmarking dataset as explained in Section 10.3.4. The relevance reflects to what extent the generated biclusters represent true biclusters in the gene dimension. In contrast, recovery quantifies how well each of the true biclusters is recovered by the biclustering algorithm.

10.4 D-Wave experiments

The D-Wave machine is able to minimize an objective function expressed in accordance to the Ising Model of statistical mechanics. This model can be arranged in a graph whose nodes represent the qubits and the edges represent interactions (couplers) between them.

The energy of the Ising model is expressed by the Hamiltonian:

$$H(\sigma_1 \dots \sigma_n) = \sum_{\langle i\ j \rangle} J_{ij}\sigma_i\sigma_j + \sum_j h_j\sigma_j \tag{10.10}$$

where $\sigma \in \{+1, -1\}$ and h_j is the external magnetic field in site j. The interaction between the qubits in site i and the one in site j is given by J_{ij} and it can be either ferromagnetic ($J_{ij} < 0$ that tends to align the qubits) or anti-ferromagnetic ($J_{ij} > 0$ that tends to misalign the qubits). The Ising energy minimization problem is equivalent through an arithmetic transformation to a quadratic unconstrained binary formulation. This means that minimizing the latter corresponds to finding the ground state energy of the associated Ising model [23] or vice versa.

Due to the engineering difficulties for the physical realization of the processor, the D-Wave machine has a well defined, fixed architecture in terms of qubits and couplers. In particular it is composed by *unit cells*, i.e groups of eight qubits disposed as a complete bipartite graph (see Figure 10.9a or equivalently Figure 10.9b).

These unit cells are disposed in a two dimensional matrix, where the left hand nodes within each unit cell connect to the relative nodes in the up/down unit cells and the right hand nodes connect to the relative nodes in the left/right unit cells (see Figure 10.10a or equivalently Figure 10.10b). The topology of this structure is the so called *chimera graph*.

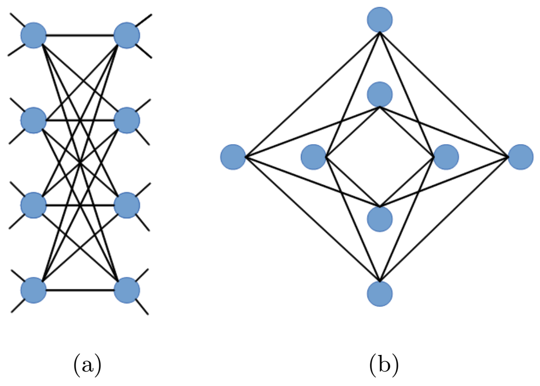


Figure 10.9: D-Wave unit cell as shown in [83]. Figures 10.9a and 10.9b are equivalent representations.

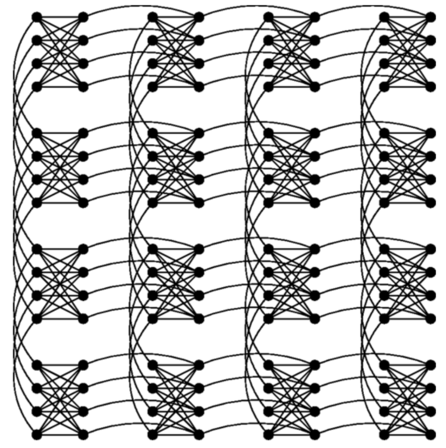
10.4.1 Minor embedding process

Different problems require different QUBO formulations and as a consequence graphs with different connectivity. In order to embed a problem into the fixed D-Wave architecture previously presented, we can choose between two possible options:

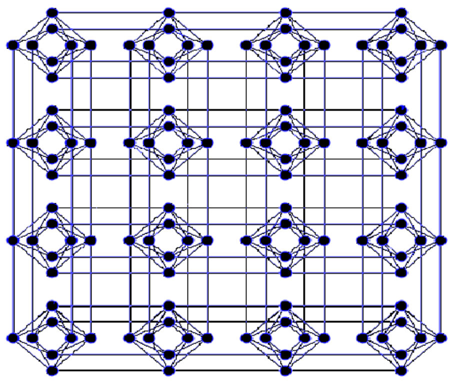
1. Formulate the QUBO model by taking into account the fixed structure of the hardware graph. That is, create a QUBO model that is directly mappable onto the chimera graph.
2. Create a logical formulation (a logical graph as we did in our QUBO models described both in Chapters 9 and 10) and subsequently embed it into the physical chimera structure through an operation called the minor embedding, where a minor of the hardware graph is calculated.

In graph theory, according to Wagner's Theorem [179], a graph H is a minor of a graph G if a graph isomorphic to H can be obtained from a subgraph of G by successive application of these three operations:

- Deletion of an edge.



(a)



(b)

Figure 10.10: Graphical representation of the *chimera graph* of a D-Wave chip with 128 qubits, Figures 10.10a and 10.10b are equivalent representations.

- Deletion of a node that is isolated after an edge deletion.
- Contraction of an edge.

An edge contraction is the act of removing the edge and simultaneously merging the two nodes with all their incident edges. A small example is shown in Figure 10.11.

Without assumptions on the input and target graphs, finding the optimal minor (i.e. the graph that uses the lowest possible number of physical qubits) is NP-hard. Although with a fixed input graph there is in principle a polynomial time algorithm due to the Robertson-Seymour theorem [151],

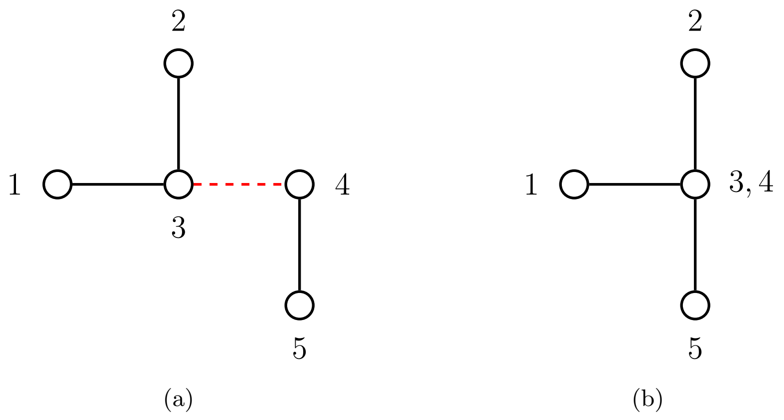


Figure 10.11: Example of the contraction operation on the red (dotted) edge, before (Figure 10.11a) and after (Figure 10.11b).

the proof is non-constructive and in practice we are limited to heuristics or complete algorithms that are intractable.

However, a non-optimal minor embedding can work properly in order to solve an instance with the D-Wave at a price of devoting more physical qubits than the optimal embedding. Heuristics in order to solve this problem has been developed and more detail on the minor embedding techniques for the D-wave device can be found in literature [40, 51, 52, 97].

10.4.2 Embedding the QUBO model in the D-Wave architecture

In this section, we report the results of the embedding process performed on a D-Wave 2X$^{\text{TM}}$ machine. The D-Wave 2X$^{\text{TM}}$ machine that we used is hosted at NASA Ames Research Laboratory and has 12×12 unit cells for a total of 1152 qubits,[2], see [60] for more details on its hardware and performance.

As explained, the minor embedding process determines a mapping of the physical qubits into the problem's variables, i.e., which physical qubits should represent which variable of the logical QUBO formulation. Note that, even if the number of nodes of the model is smaller than the number of qubits of the processor, it is not always possible to find a valid embedding. In particular, the embedding into the hardware architecture usually requires more variables, since some nodes are represented by several physical qubits (a "chain" of qubits) due to the sparse connectivity of the hardware graph. All the experiments described here have been performed by applying the embedding process to our model using the official D-Wave libraries that

[2]Note that only 1097 of 1152 qubits are operational.

implements the heuristic proposed by Cai et al. [40].

The parameters we used in the embedding process are those standard provided by the D-Wave. Moreover, we perform only a single embedding attempt with standard parameters. This approach, also followed in literature, (e.g., [136]), is based on the Cai et al. heuristics [40] mentioned before which may be very suboptimal. As done in [177], [147] and [142], we could study an optimal choice of the parameters that is more appropriate for the biclustering problem. This may lead to a better performance of the D-Wave on our problem.

For our experiments on the D-Wave, we randomly generated the following instances (matrices) for the biclustering problem:

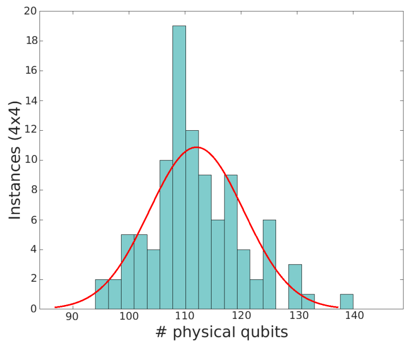
- 100 instances of a size of 4×4 and with bicluster of 2×2
- 100 instances of a size of 5×5, 50 of which with a bicluster 2×3 and 50 with a bicluster 3×2
- 100 instances of a size of 6×6 and with bicluster of 3×3

All these instances are without noise and from these we generated the QUBO models using the additive coherence measure (equation 10.3) with a weight parameter $w = 1$. Results of the number of physical qubits required after this embedding phase can be observed in the histograms with a Gaussian distribution fit in Figure 10.12a for the 4×4 instances, Figure 10.12b for the 5×5 instances and Figure 10.12c for the 6×6 instances. In Table 10.1 we report the aggregated results with mean and standard deviation of the number of physical qubits required per instance.

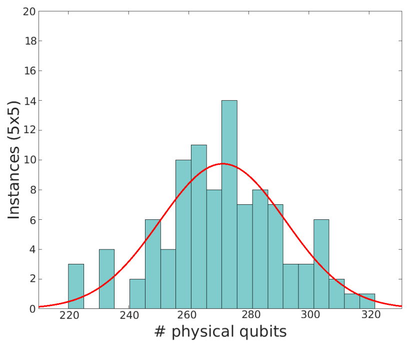
Table 10.1: Results of the embedding phase: number of physical qubits required to embed an instance.

Size	**Min**	**Max**	μ	σ
4×4	94	139	112.03	8.45
5×5	220	321	271.33	20.90
6×6	511	757	634.37	49.41

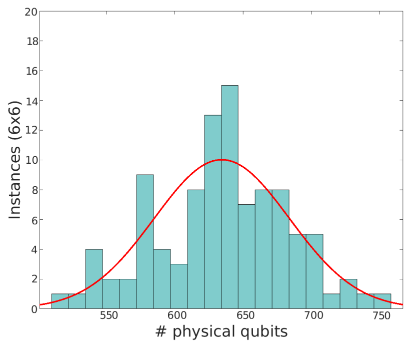
We can observe that the number of physical qubits required grows significantly as the instance size increases. With just a starting matrix of 6×6, we already require almost half of the available physical qubits. As previously mentioned, this is due to the fact that the few available connections between physical qubits on the D-Wave architecture necessarily lead to the use of a high number of physical qubits to represent a single logical qubit. In fact, our biclustering model consists of fully connected sub-components which leads to a quadratic overhead even for the most efficient embedding [30]. More details on the number of physical qubits required to represent a single logical one after the embedding phase can be observed in Figure 10.13a for



(a)



(b)



(c)

Figure 10.12: Results of the embedding phase: number of physical qubits required to embed (a) the 4×4 instances, (b) the 5×5 instances and (c) the 6×6 instances.

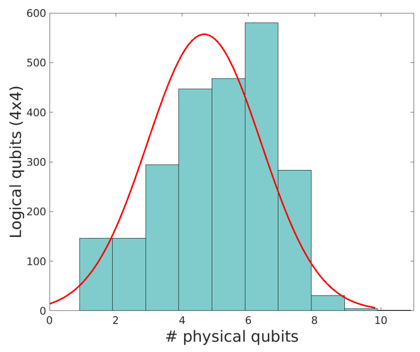
the 4×4 instances, Figure 10.13b for the 5×5 instances and Figure 10.13c for the 6×6 instances.

In Table 10.2 we report the aggregated results with mean and standard deviation of the number of physical qubits required per single logical qubit. As reported in all the cases, for some logic qubits, the embed requires a minimum of 1 physical qubits. However, the maximum number required grows as the instance dimension increases. Specifically, the maximum number required for some qubits in the 6×6 instances is 30, which it is three times the maximum required number for the 4×4 instance.

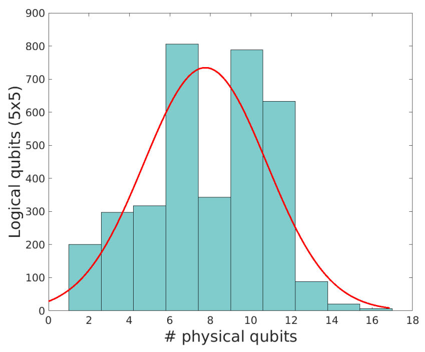
10.4.3 Suitability of D-Wave for biclustering problem

Before tackling an optimization problem with a quantum annealing device, as suggested by King et al. [95] it is crucial to ensure that the problem shows:

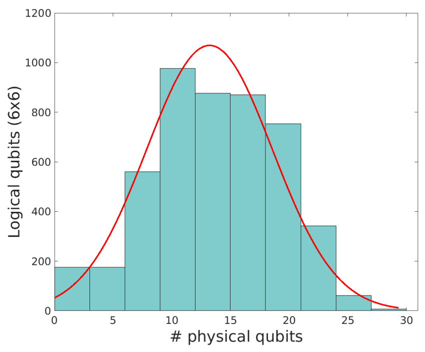
- *global frustration*, i.e., it requires a non trivial combinatorial optimiza-



(a)



(b)



(c)

Figure 10.13: Results of the embedding phase: number of physical qubits per logical qubit in (a) the 4×4 instances, (b) the 5×5 instances and (c) the 6×6 instances.

Table 10.2: Results of the embedding phase: number of physical qubits per logical qubit varying instance size.

Size	**Min**	**Max**	μ	σ
4×4	1	10	4.67	1.72
5×5	1	17	7.75	3.04
6×6	1	30	13.22	5.37

tion. An Ising model is frustrated when the competition between ferromagnetic and anti-ferromagnetic couplings leads to a ground state where the interaction energies between qubits cannot be simultaneously minimized.

- *local ruggedness*, i.e., the problem presents a landscape with tall and thin barriers.

As biclustering is known to be NP-hard, we expect that its logical

Ising/QUBO formulation straightforwardly displays global frustration. In fact, it is possible to show such a behavior even in the limit of a 1-dimensional matrix biclustering, which can be seen as the building block of any biclustering instance (see Figure 10.14).

In this trivial case, the geometry of the problem is reduced to a complete graph with three vertices. A frustrated behavior, with two ferromagnetic couplings and an anti-ferromagnetic one, prevails when the magnitudes of the weights associated with the three edges become of the same order,[3] i.e., when the B parameter is significantly larger than V.

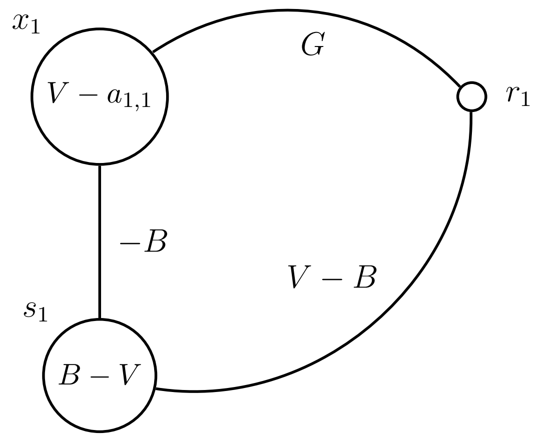


Figure 10.14: Graph of the complete model for $N = 1$ and $M = 1$.

If we consider an arbitrary $N \times M$ biclustering instance, frustration increases because of the presence of NM triangular loops as shown in Figure 10.14, which share vertices among themselves. Moreover, as we can see from equation 10.9, since B has a linear dependence on N and M, while V does not, this automatically pushes the model in a $V \ll B$ highly frustrated regime when increasing N or M.

Usually, the complex landscape typical of frustrated systems only guarantees the presence of many local minima and maxima and it does not imply that barriers separating them are tall and narrow enough for quantum annealing to work properly. In our biclustering model, such condition of local ruggedness is given by the QUBO formulation, since the geometry of the problem results in clusters of nodes which are internally ferro-magnetic coupled [95]. This feature, from the point of view of the energy landscape, translates into the presence of high and narrow barriers separating minima. To summarize, the complexity of our biclustering model results in a macroscopically interesting landscape with multiple local minima (global frustration) and the particular geometry of the problem guarantees the high

[3]Note that parameter G can always be chosen close to B.

and narrow barriers in the landscape (local ruggedness).

10.4.4 D-Wave experiments results

The objective of this experimental phase is to determine whether the D-Wave 2X$^{\mathrm{TM}}$ machine is able to retrieve the optimal solution of the QUBO objective functions of the instances previously described. The D-Wave takes as input the number of reads a *num_reads* parameter which identifies the number of states (output solutions) to read from the solver in each programming cycle (which we set as described later) along with other hardware specific parameters for which we used the default values of the machine (e.g., the default annealing time for every read of 20 microseconds). In this experimental phase, we solved every instance previously described in Section 10.4.2 with the following protocol:

- We solved the QUBO instance using the CPLEX library in order to find the configuration that gives the optimum of the objective function. For more details on the optimization using CPLEX please refer to Appendix B.

- We run the instance on the D-Wave 2X$^{\mathrm{TM}}$ machine. Specifically, we run a programming cycle asking for 10,000 reads. Hence the D-Wave samples the objective function 10,000 times and returns the 10,000 solutions.

- We process the sampled solutions comparing them to the one obtained with CPLEX, in order to check if the optimum has been found.

- If the optimum has not been found, we repeat the process with a new programming cycle (we set the maximum iteration to 1000 cycle. However, it was not necessary to perform so many cycles as can be seen in the following results).

Regarding the 4×4 instances, as we can observe in Figure 10.15, we obtained most of optimum solution in just one programming cycle and no more than 4 cycles was required to solve all the 100 instances. As expected, the number of cycles required grows as the instance size increases. In more details, regarding the 5×5 matrices, we also obtained most of the times the optimum in one cycle and solved all 100 instances in no more than 55 cycles and regarding the 6×6 we always obtained the optimum solution in less then 550 cycles. Specifically, just one 6×6 instance required 550 runs, we solved all the other 99 instances in up to 228 cycles. We also report in Figure 10.16 the average number of cycles required per instance size. These results lead to the conclusion that it was always possible to get the optimal solution for all generated QUBO instances.

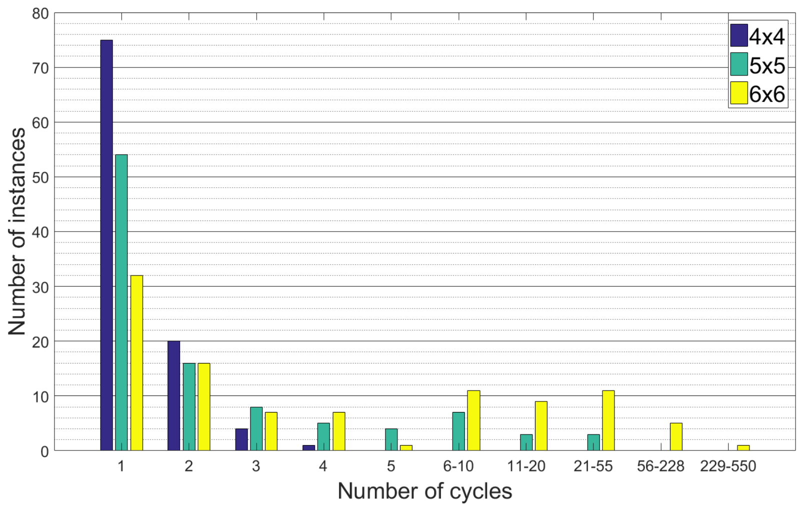


Figure 10.15: Histogram of the number of instances where the optimum of the objective function has been found after a specific number of programming cycles, varying the instance size.

As previously done in literature [150], we compute the probability of success Ps for a $20\mu s$ annealing time (which is the annealing time we used for a single read). For each set of instances of the same dimension, we then compute the expected number of runs $k = \dfrac{\ln(0.01)}{\ln(1-Ps)}$ required to obtain a 99% success probability and multiply it for $20\mu s$ to compute the total annealing time required to obtain a 99% success. Results are shown in Figure 10.17. Although solving such small instances with CPLEX takes milliseconds, with these experiments we have demonstrated the feasibility of using the D-Wave to solve the biclustering problem. We expect with further developments of the D-Wave machine to be able to solve bigger instances and reduce the annealing time required.

10.5 Conclusions

In this chapter we investigated the possible use of quantum annealing in order to solve biclustering problems. In particular, we introduced a quadratic unconstrained binary optimization model for the one-bicluster problem and analyzed its properties and correctness.

To cluster the elements of an input data-matrix we used as similarity concept two different coherence measures, namely the constant and the additive coherence. Such measures allows to find biclusters with different properties.

We intensively tested our model with different parameters using the ad-

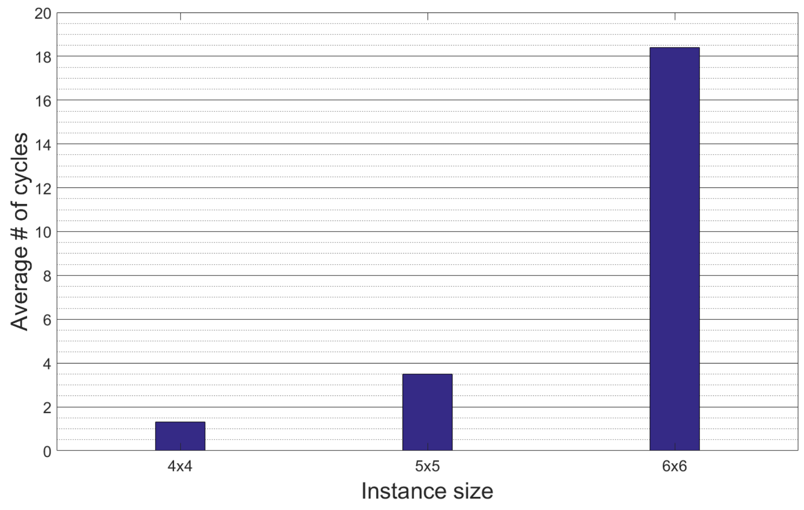


Figure 10.16: Average number of programming cycles required to find the optimum solution of the QUBO objective function varying the instance size.

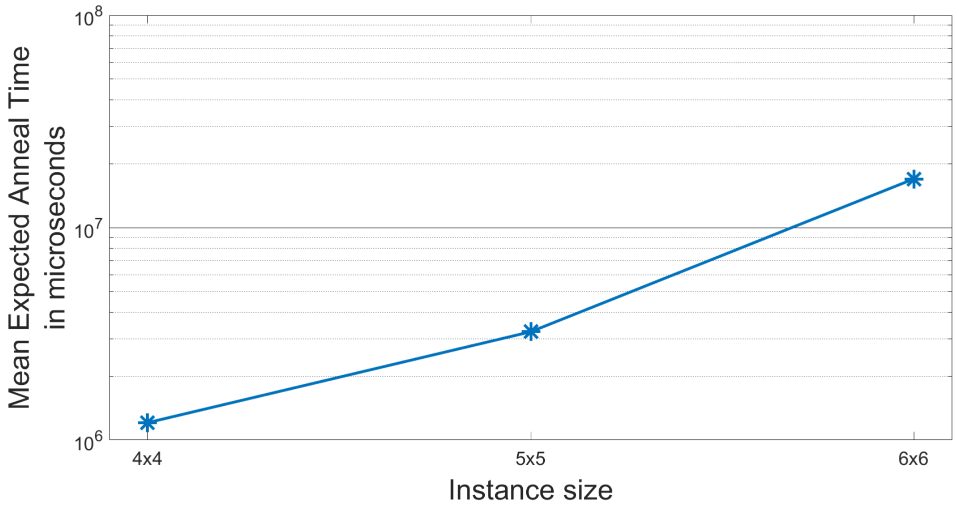


Figure 10.17: Expected total annealing time in microseconds to 99% success for the three instance sizes.

ditive coherence measure on a synthetic dataset. A comparison of the results against state-of-the-art techniques shows the validity of our approach. In particular our model is able to retrieve biclusters with a higher accuracy, significantly outperforming the previous approaches.

Moreover, for a deeper analysis, we investigated different sparsification heuristics in order to mitigate the space complexity of the model, showing that it is possible to obtain a significant gain in space while maintaining a good quality of the solutions.

As for the practical applicability of quantum annealing to solve the biclustering problem, we have tested our model by means of real experiments on a D-Wave 2X$^{\mathrm{TM}}$ machine. Results suggest that the use of a quantum annealing approach is feasible only for small matrices. This is due to the current architecture of the D-Wave processor.

Although the use of sophisticated representations techniques to ameliorate the limitations imposed by current experimental hardware [21] or the use of frameworks to test the statistical significance of the discovered biclusters by filtering the solutions with state-of-the-art statistical tests [86] is outside of the scope of this contribution, we believe that further developments of the D-Wave machine including the use of a larger number of qubits with higher connectivity could allow us to practically use quantum annealing for hard real-world problems involving biclustering. Thus, the model that we proposed takes an important step toward the effective use of quantum annealing for solving the biclustering problem.

Chapter 11

Conclusions and future work

In many environmental monitoring applications the data collection process must consider limited resources. As a consequence, it is fundamental to carefully select the sampling locations in order to maximize the information obtained from the environment. In thesis we proposed novel solutions for several interesting problems in this field. Our approaches to address information gathering problems are characterized by different properties such as the efficiency of the computation or the strategy of acquisition of data from the environment.

As a matter of fact, problems involving environmental monitoring applications can be distinguished into two macro areas depending on the strategy of acquisition of data from the environment. When the information gathering process is performed using mobile sensors (such as unmanned ground vehicles, unmanned aerial vehicles or autonomous surface vessels) the key problem is the generation of sensing trajectories constrained by the limited resources available, such as, battery lifetime or computational capabilities of the mobile platform. Instead, when the information gathering process is performed using a set of fixed sensors, the key problem is the selection of the locations where to carry out the deployment, constrained by the limited number of sensors available.

Our thesis is inserted in the above mentioned scenarios and our contributions range between these two macro areas. Specifically:

- For the case of mobile sensors, in Part II of the thesis, we focus on the problem known as *level set estimation*. In the level set estimation problem we have an unknown scalar field that represents an environmental phenomena of interest. The objective is to partition each location of the scalar field into two groups that represent location either above or below a given threshold level. The objective is to gather as much information as possible from the environment in order to model the unknown scalar field and be able to classify each location.

  Unlike other approaches in the literature, we consider the case where mobile platforms with low computational power can continuously acquire measurements with a negligible cost. This scenario requires to optimize sensing trajectories by considering that we can obtain data throughout the entire path executed by the sensor.

In this context we propose two novel algorithms specifically designed to solve this problem minimizing the computational effort. Specifically, SBOLSE (Chapter 4) that casts informative path planning into an orienteering problem and PULSE (Chapter 5) that exploits a less accurate but computationally faster path selection procedure.

We evaluated the performance of our algorithms both on a real world and a synthetic dataset. Results show that our techniques are able to obtain a high quality classification with a shorter path and a lower computation time compared to state-of-the-art algorithms in literature for the level set estimation problem.

- In the context of fixed sensors, in Part III of the thesis, we focused on the variance minimization of a Gaussian process. When modeling an unknown phenomena using a Gaussian process a key problem is to decide on the locations where measurements are going to be taken while considering constraints on the number of available sensors.

  Unlike other approaches in literature that consider only a set of available sampling location to choose from, we consider the case where sampling location can be chose from a domain of interest that is continuous, e.g. a portion of the a body of water.

  In this context we proposed two novel techniques specifically designed to select a set of sampling location from a continuous domain in order to minimize the posterior variance of a Gaussian process. Specifically, a gradient descent procedure (Chapter 7) to iteratively improve an initial set of observations and a novel heuristic technique (Chapter 8) that behaves as gradient descent on an approximated objective function but with a much faster computation time.

  We evaluated our techniques with different settings and results show that in many cases it is possible to obtain a significant improvement with respect the well-known submodular greedy procedure used in literature.

The above mentioned contributions are composed by a set of classical heuristic algorithms. In addition to that, in Part IV of the thesis, we also investigate the possibility of using a quantum computation approach in the context of information gathering optimization. The emergence of quantum information processing could provide a viable alternative to combat the intrinsic complexity of several hard optimization problems.

A notable progress in this direction is due to the recent development of the D-Wave quantum annealer, whose processor is designed to solve Quadratic Unconstrained Binary Optimization (QUBO) problem. As a consequence many works in literature investigate the possibility of using quantum annealing to address hard artificial intelligence problems by proposing their QUBO formulation.

In this context we proposed two QUBO models designed to solve two specific problems in the context of information gathering for environmental monitoring applications. In more details:

- We propose (in Chapter 9) a QUBO model that falls within the same context of Gaussian process variance reduction. In contrast to the others two contributions (gradient descent and heuristic) presented in this thesis, our QUBO model formulates the problem in a discrete manner, i.e. sampling points can be chosen from a given set of feasible locations.

  Results of our empirical evaluation shows that the optimum of the QUBO objective function represents a good solution for the problem obtaining comparable and in some cases better results than the widely used submodular technique. Hence, the model we proposed takes an important step towards sampling optimization of Gaussian process in the context of quantum computation.

- We propose (in Chapter 10) a QUBO model for the biclustering problem. Biclustering, also known in other scenarios as subspace clustering, is a term used to encompass a large set of data mining techniques. Although biclustering is widely used in different scenarios (e.g. gene expression microarray data analysis), recent work has adopted biclustering for the analysis of data in environmental monitoring applications.

  We investigated the use of quantum annealing for biclustering problems, in particular we evaluated by testing our model by means of experiments on a D-Wave 2X$^{\text{TM}}$ machine. Results shows a practical applicability of quantum annealing for small instances of the problem.

Our contributions proposed in this thesis open several new research possibilities which extend to different aspects of environmental monitoring applications:

- The techniques proposed in the context of mobile sensors can be extended in various ways. For example, a natural extension is represented by the application of multiple platforms. In this case the algorithms proposed could be extended through the use of techniques derived from the multi-agent system literature.

Another possible interesting research direction is given by the introduction of additional constraints on the planned trajectories. Since many different mobile robots are employed during monitoring operations, these constraints could represent the physical maneuver capabilities (e.g steering capacity) of the specific drones used in a given application.

- In the context of optimizing sensing locations to minimize the posterior variance of the Gaussian process, the techniques we proposed are designed and tested with the most common used squared exponential kernel. A natural evolution of these contributions is the application of different kernels, allowing to study and define the properties and the performances in different scenarios.

  Moreover, a possible extension is the introduction of additional constraints that define specific properties that the sampling locations must satisfy. For example the enforcement of a maximum distance such that sensors can reliably communicate one with the other.

- Regarding our QUBO model for the Gaussian process posterior variance reduction, we proved that the optimum of the objective function represents a good solution for the problem. However, as future work an empirical evaluation using a D-Wave quantum annealer is necessary to determine the degree of precision and the utility of the model.

The increasing use of sensor networks and mobile platforms in real-world applications requires effective computational models and efficient solution techniques. Overall, this thesis provides novel contributions in the context of information gathering optimization for environmental monitoring applications. We believe that our work provides a novel set of tools that allows to address these challenging real-world applications under new perspectives.

This thesis shows that the use of QUBO approaches are effective and flexible for environmental monitoring applications. The study of new quantum computation techniques for environmental monitoring and mobile sensors control and exploration constitutes an interesting evolution of the topics covered in this thesis. However, we believe that the use of such models, coupled with the development and advances in the quantum computing field can provide significant advances beyond environmental monitoring applications.

Appendices

Appendix A

Calculation of the gradient

This appendix refers to the calculation of the gradient used in the gradient descent procedure presented in Chapter 7.

Given the objective function:

$$J(\{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_K\}) = \sum_{\mathbf{x}_i \in \mathcal{X}} \sigma^2(\mathbf{x}_i) \tag{A.1}$$

where $\sigma^2(x)$ is the Gaussian process variance computed as defined by equation 3.7 that we report here below for convenience:

$$\mathbb{V}[f_*] \triangleq \sigma^2(\mathbf{x}_*) = k(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{k}_*^T(\mathbf{K} + \sigma_n^2\mathbf{I})^{-1}\mathbf{k}_* \tag{A.2}$$

we want to compute the gradient ∇J. As mentioned before in Section 7.1.1 given the set of K measurements and a domain $\mathcal{X}$ in d dimensions, the objective function J is a Kd multi-dimensional function. The gradient of a multi-variable scalar function denotes the vector of partial derivatives with respect to the components. In what follows we will present the derivative with respect to a single component, that is the derivative of the function with respect to the position of a candidate sampling point.

Moreover, for the sum rule of derivatives $\frac{\partial}{\partial x}(u + v) = \frac{\partial u}{\partial x} + \frac{\partial v}{\partial x}$ we can simply show the derivative of the Gaussian process variance in a single point x_i. The complete derivative will be the sum of the single variance's derivatives.

For the sake of readability of the calculations we slightly change the syntax of the variance in a point x as follows:

$$\sigma^2(x) = k(x, x) - k(x, \bar{t})^T \left(k(\bar{t}, \bar{t}) + \sigma_n^2 I \right)^{-1} k(x, \bar{t}) \tag{A.3}$$

where $\bar{t}$ identifies the vector of sampling points and the partial derivative $\frac{\partial}{\partial t_{i,d}}\sigma^2(x)$ indicates the derivative of the variance in x by moving a candidate sampling location t_i along d-th dimension:

$$\frac{\partial}{\partial t_{i,d}}\sigma^2(x) = -\left[0\cdots 0\frac{\partial}{\partial t_{i,d}}k(x,t_{i,d})0\cdots 0\right]\left(k(\bar{t},\bar{t})+\sigma_n^2 I\right)^{-1}k(x,\bar{t})$$
$$+\,k(x,\bar{t})^T\left(k(\bar{t},\bar{t})+\sigma_n^2 I\right)^{-1}\left[\frac{\partial}{\partial t_{i,d}}\left(k(\bar{t},\bar{t})+\sigma_n^2 I\right)\right]\left(k(\bar{t},\bar{t})+\sigma_n^2 I\right)^{-1}k(x,\bar{t})$$
$$-\,k(x,\bar{t})^T\left(k(\bar{t},\bar{t})+\sigma_n^2 I\right)^{-1}\left[0\cdots 0\frac{\partial}{\partial t_{i,d}}k(x,t_{i,d})0\cdots 0\right]^T \quad \text{(A.4)}$$

Where $\frac{\partial}{\partial t_{i,d}}\left(k(\bar{t},\bar{t})+\sigma_n^2 I\right) =$

$$\begin{bmatrix}
0 & \dots & 0 & \frac{\partial}{\partial t_{i,d}}k(t_1,t_{i,d}) & 0 & \dots & 0\\
\vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots\\
0 & \dots & 0 & \vdots & 0 & \dots & 0\\
\frac{\partial}{\partial t_{i,d}}k(t_{i,d},t_1) & \dots & \dots & \frac{\partial}{\partial t_{i,d}}k(t_{i,d},t_{i,d}) & \dots & \dots & \frac{\partial}{\partial t_{i,d}}k(t_{i,d},t_n)\\
0 & \dots & 0 & \vdots & 0 & \dots & 0\\
\vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots\\
0 & \dots & 0 & \frac{\partial}{\partial t_{i,d}}k(t_n,t_{i,d}) & 0 & \dots & 0
\end{bmatrix} \quad \text{(A.5)}$$

where $\frac{\partial}{\partial t_{i,d}}k(x,x')$ denote the partial derivative of the kernel function used in the Gaussian process regression.

This approach is general to any derivable kernel function $k(x,x')$. However, as discussed in the empirical evaluation (Section 7.3), in our experiments for the gradient descent contribute of the thesis we are using the squared exponential kernel:

$$k(\mathbf{x},\mathbf{x}') = \sigma_f^2 \exp\left(-\frac{(\mathbf{x}-\mathbf{x}')^T(\mathbf{x}-\mathbf{x}')}{2l^2}\right) \quad \text{(A.6)}$$

that in the case of a d-dimensional domain can be rewritten as:

$$k(\mathbf{x},\mathbf{x}') = \sigma_f^2 \exp\left(-\frac{[x_1-x_1',\cdots,x_d-x_d'][x_1-x_1',\cdots,x_d-x_d']^T}{2l^2}\right) \quad \text{(A.7)}$$

Hence, the derivative of this specific kernel function obtained by moving a candidate training point t_i along the d-th dimension is:

$$\frac{\partial}{\partial t_{i,d}}k(x,t_{i,d}) = \frac{\sigma_f^2(x_d-t_{i,d})}{l^2}\exp\left(-\frac{[x_1-t_1,\cdots,x_d-t_d][x_1-t_1,\cdots,x_d-t_d]^T}{2l^2}\right) \quad \text{(A.8)}$$

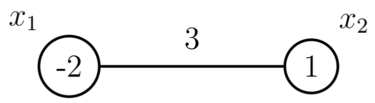
Appendix B

CPLEX model

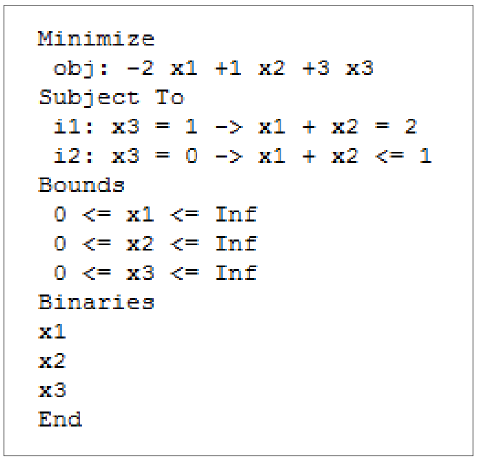
In order to optimize the QUBO models presented in Chapters 9 and 10 using ILOG CPLEX Optimization Studio, each QUBO instance has been linearized using the following procedure:

- For each node i in the QUBO graph, we create a variable x_i with a coefficient equal to the value inside the node itself.
- For each edge in the QUBO graph between two nodes i and j:
  - We create a new variable x_k with a coefficient equal to the value of the edge itself.
  - We create two constraints (using the syntax required by CPLEX):
    1. $x_k = 1$ -> $x_i + x_j = 2$
    2. $x_k = 0$ -> $x_i + x_j <= 1$
- We bound the variables to be binary $\{0, 1\}$.

The total number of variables in the CPLEX linear form is equal to the number of nodes and edges combined. For example, given the following QUBO instance graph:



its linear formulation using CPLEX syntax is shown in Figure B.1



```
Minimize
 obj: -2 x1 +1 x2 +3 x3
Subject To
 i1: x3 = 1 -> x1 + x2 = 2
 i2: x3 = 0 -> x1 + x2 <= 1
Bounds
 0 <= x1 <= Inf
 0 <= x2 <= Inf
 0 <= x3 <= Inf
Binaries
x1
x2
x3
End
```

Figure B.1: Example with CPLEX syntax of a linearized QUBO problem. Variable x_3 represents the edge between nodes x_1 and x_2.

Bibliography

[1] Ameer Ahmed Abbasi and Mohamed Younis. A survey on clustering algorithms for wireless sensor networks. *Computer Communications*, 30(14):2826 – 2841, 2007. Network Coverage and Routing Schemes for Wireless Sensor Networks.

[2] Steven H Adachi and Maxwell P Henderson. Application of quantum annealing to training of deep neural networks. *arXiv preprint arXiv:1510.06356*, 2015.

[3] Wesam H Al-Sabban, Luis F Gonzalez, and Ryan N Smith. Wind-energy based path planning for unmanned aerial vehicles using markov decision processes. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 784–789. IEEE, 2013.

[4] Francesco Amigoni. Experimental evaluation of some exploration strategies for mobile robots. In *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pages 2818–2823. IEEE, 2008.

[5] Francesco Amigoni and Vincenzo Caglioti. An information-based exploration strategy for environment mapping with mobile robots. *Robotics and Autonomous Systems*, 58(5):684–699, 2010.

[6] Mohammad H Amin. Searching for quantum speedup in quasistatic quantum annealers. *Physical Review A*, 92(5):052323, 2015.

[7] David L Applegate, Robert E Bixby, Vasek Chvatal, and William J Cook. The traveling salesman problem: A computational study. *Princeton Series in Applied Mathematics, Princeton University Press*, 2007.

[8] Nikolay A Atanasov, Jerome Le Ny, and George J Pappas. Distributed algorithms for stochastic source seeking with mobile robot networks. *Journal of Dynamic Systems, Measurement, and Control*, 137(3):031004, 2015.

[9] Wassim Ayadi, Mourad Elloumi, and Jin Kao Hao. Bimine+: an efficient algorithm for discovering relevant biclusters of dna microarray data. *Knowledge-Based Systems*, 35:224–234, 2012.

[10] Liviu Badea. Generalized clustergrams for overlapping biclusters. In *Proceedings of the 21st International Jont Conference on Artifical Intelligence*, IJCAI'09, pages 1383–1388, San Francisco, CA, USA, 2009. Morgan Kaufmann Publishers Inc.

[11] Francisco Barahona. On the computational complexity of ising spin glass models. *Journal of Physics A: Mathematical and General*, 15(10):3241, 1982.

[12] Maxim A Batalin, Mohammad Rahimi, Yan Yu, Duo Liu, Aman Kansal, Gaurav S Sukhatme, William J Kaiser, Mark Hansen, Gregory J Pottie, Mani Srivastava, and Deborah Estrin. Call and response: Experiments in sampling the environment. In *Proceedings of the 2Nd International Conference on Embedded Networked Sensor Systems*, SenSys '04, pages 25–38, New York, NY, USA, 2004. ACM.

[13] Christian Bauckhage. Lecture notes on data science: Soft k-means clustering.

[14] Behzad Bayat, Naveena Crasta, Alessandro Crespi, António M. Pascoal, and Auke Ijspeert. Environmental monitoring using autonomous vehicles: a survey of recent searching techniques. *Current Opinion in Biotechnology*, 45:76 – 84, 2017. Energy biotechnology • Environmental biotechnology.

[15] Behzad Bayat, Naveena Crasta, Howard Li, and Auke Ijspeert. Optimal search strategies for pollutant source localization. In *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*, pages 1801–1807. Ieee, 2016.

[16] Amir Ben-Dor, Benny Chor, Richard Karp, and Zohar Yakhini. Discovering local structure in gene expression data: the order-preserving submatrix problem. *Journal of computational biology*, 10(3-4):373–384, 2003.

[17] Marcello Benedetti, John Realpe-Gómez, Rupak Biswas, and Alejandro Perdomo-Ortiz. Estimation of effective temperatures in quantum annealers for sampling applications: A case study with possible applications in deep learning. *Physical Review A*, 94(2):022308, 2016.

[18] Marcello Benedetti, John Realpe-Gómez, Rupak Biswas, and Alejandro Perdomo-Ortiz. Quantum-assisted learning of graphical models with arbitrary pairwise connectivity. *arXiv preprint arXiv:1609.02542*, 2016.

[19] Pavel Berkhin. *A Survey of Clustering Data Mining Techniques*, pages 25–71. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006.

[20] James C Bezdek. Objective function clustering. In *Pattern recognition with fuzzy objective function algorithms*, pages 43–93. Springer, 1981.

[21] Zhengbing Bian, Fabian Chudak, Robert Israel, Brad Lackey, William G. Macready, and Aidan Roy. Discrete optimization using quantum annealing on sparse ising models. *Frontiers in Physics*, 2:56, 2014.

[22] Zhengbing Bian, Fabian Chudak, William Macready, Aidan Roy, Roberto Sebastiani, and Stefano Varotti. Solving sat and maxsat with a quantum annealer: Foundations, encodings, and preliminary results. *arXiv preprint arXiv:1811.02524*, 2018.

[23] Zhengbing Bian, Fabian Chudak, William G Macready, and Geordie Rose. The ising model: teaching an old problem new tricks. *D-wave systems*, 2, 2010.

[24] Manuele Bicego, Pietro Lovato, Alberto Ferrarini, and Massimo Delledonne. Biclustering of expression microarray data with topic models. In *Pattern Recognition (ICPR), 2010 20th International Conference on*, pages 2728–2731. IEEE, 2010.

[25] Jonathan Binney and Gaurav S Sukhatme. Branch and bound for informative path planning. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 2147–2154. IEEE, 2012.

[26] Rupak Biswas, Zhang Jiang, Kostya Kechezhi, Sergey Knysh, Salvatore Mandrà, Bryan O'Gorman, Alejandro Perdomo-Ortiz, Andre Petukhov, John Realpe-Gómez, Eleanor Rieffel, Davide Venturelli, Fedir Vasko, and Zhihui Wang. A nasa perspective on quantum computing: Opportunities and challenges. *Parallel Computing*, 64:81 – 98, 2017. High-End Computing for Next-Generation Scientific Discovery.

[27] Harry Blum. A Transformation for Extracting New Descriptors of Shape. *Models for the Perception of Speech and Visual Form*, pages 362–380, 1967.

[28] Sergio Boixo, Tameem Albash, Federico M Spedalieri, Nicholas Chancellor, and Daniel A Lidar. Experimental signature of programmable quantum annealing. *Nature Communications*, 4(2067), 2013.

[29] Sergio Boixo, Troels F Rønnow, Sergei V Isakov, Zhihui Wang, David Wecker, Daniel A Lidar, John M Martinis, and Matthias Troyer. Evidence for quantum annealing with more than one hundred qubits. *Nature Physics*, 10:218–224, 2014.

[30] Tomas Boothby, Andrew D. King, and Aidan Roy. Fast clique minor generation in chimera qubit connectivity graphs. *Quantum Information Processing*, 15(1):495–508, 2016.

[31] Max Born and Vladimir Fock. Beweis des adiabatensatzes. *Zeitschrift für Physik*, 51(3-4):165–180, 1928.

[32] Lorenzo Bottarelli, Manuele Bicego, Jason Blum, Nicola Bombieri, Alessandro Farinelli, and Luca Veggian. Orienteering-based path selection for mobile sensors. In *Proceedings of the 3rd Italian Workshop on Artificial Intelligence and Robotics - A workshop of the XV International Conference of the Italian Association for Artificial Intelligence, AIRO@AI*IA 2016, Genova, Italy, November 28, 2016.*, pages 36–40, 2016.

[33] Lorenzo Bottarelli, Manuele Bicego, Jason Blum, and Alessandro Farinelli. Skeleton-based orienteering for level set estimation. In *ECAI 2016 - 22nd European Conference on Artificial Intelligence, 29 August-2 September 2016, The Hague, The Netherlands - Including Prestigious Applications of Artificial Intelligence (PAIS 2016)*, pages 1256–1264, 2016.

[34] Lorenzo Bottarelli, Manuele Bicego, Jason Blum, and Alessandro Farinelli. Orienteering-based informative path planning for environmental monitoring. *Engineering Applications of Artificial Intelligence*, 77:46 – 58, 2019.

[35] Lorenzo Bottarelli, Manuele Bicego, Matteo Denitto, Alessandra Di Pierro, and Alessandro Farinelli. A quantum annealing approach to biclustering. In *Theory and Practice of Natural Computing - 5th International Conference, TPNC 2016, Sendai, Japan, December 12-13, 2016, Proceedings*, pages 175–187, 2016.

[36] Lorenzo Bottarelli, Manuele Bicego, Matteo Denitto, Alessandra Di Pierro, Alessandro Farinelli, and Riccardo Mengoni. Biclustering with a quantum annealer. *Soft Comput.*, 22(18):6247–6260, 2018.

[37] Lorenzo Bottarelli, Jason Blum, Manuele Bicego, and Alessandro Farinelli. Path efficient level set estimation for mobile sensors. In *Proceedings of the Symposium on Applied Computing*, SAC '17, pages 262–267, New York, NY, USA, 2017. ACM.

[38] Lorenzo Bottarelli and Marco Loog. Gradient descent for Gaussian processes variance reduction. In *Structural, Syntactic, and Statistical Pattern Recognition - Joint IAPR International Workshop, S+SSPR 2018, Beijing, China, August 17-19, 2018, Proceedings*, pages 160–169, 2018.

[39] J. Brooke, D. Bitko, T. F., Rosenbaum, and G. Aeppli. Quantum annealing of a disordered magnet. *Science*, 284(5415):779–781, 1999.

[40] Jun Cai, William G Macready, and Aidan Roy. A practical heuristic for finding graph minors. *ArXiv e-prints*, June 2014.

[41] Nannan Cao, Kian Hsiang Low, and John M. Dolan. Multi-robot informative path planning for active sensing of environmental phenomena: A tale of two algorithms. In *Proceedings of the 2013 International Conference on Autonomous Agents and Multi-agent Systems*, AAMAS '13, pages 7–14, Richland, SC, 2013. International Foundation for Autonomous Agents and Multiagent Systems.

[42] Luca Carlone, Jingjing Du, Miguel Kaouk Ng, Basilio Bona, and Marina Indri. Active slam and exploration with particle filters using kullback-leibler divergence. *Journal of Intelligent & Robotic Systems*, 75(2):291–311, 2014.

[43] Alberto Castellini, Giovanni Beltrame, Manuele Bicego, Jason Blum, Matteo Denitto, and Alessandro Farinelli. Unsupervised activity recognition for autonomous water drones. In *Proceedings of the 33rd Annual ACM Symposium on Applied Computing*, SAC '18, pages 840–842, New York, NY, USA, 2018. ACM.

[44] Alberto Castellini, Francesco Masillo, Manuele Bicego, Domenico Bloisi, Jason Blum, Alessandro Farinelli, and Sergio Peigner. Subspace clustering for situation assessment in aquatic drones. In *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*, SAC '19, New York, NY, USA, 2019. ACM.

[45] Chandra Chekuri, Nitish Korula, and Martin Pál. Improved algorithms for orienteering and related problems. *ACM Trans. Algorithms*, 8(3):23:1–23:27, July 2012.

[46] Hongguang Cheng, Zhifeng Yang, and Christine W. Chan. An expert system for decision support of municipal water pollution control. *Engineering Applications of Artificial Intelligence*, 16(2):159 – 166, 2003. Applications of Artificial Intelligence for Management and Control of Pollution Minimization and Mitigation Processes.

[47] Yizong Cheng and George M Church. Biclustering of expression data. In *Ismb*, volume 8, pages 93–103, 2000.

[48] Doo-Hyun Cho, Jung-Su Ha, Sujin Lee, Sunghyun Moon, and Han-Lim Choi. Informative path planning and mapping with multiple uavs in wind fields. In *Distributed Autonomous Robotic Systems*, pages 269–283. Springer, 2018.

[49] Han-Lim Choi. *Adaptive sampling and forecasting with mobile sensor networks.* PhD thesis, Massachusetts Institute of Technology, 2009.

[50] Han-Lim Choi and Jonathan P How. Continuous trajectory planning of mobile sensors for informative forecasting. *Automatica*, 46(8):1266–1275, 2010.

[51] Vicky Choi. Minor-embedding in adiabatic quantum computation: I. the parameter setting problem. *Quantum Information Processing*, 7(5):193–209, 2008.

[52] Vicky Choi. Minor-embedding in adiabatic quantum computation: Ii. minor-universal graph design. *Quantum Information Processing*, 10(3):343–353, 2011.

[53] Stephen A Cook. The complexity of theorem-proving procedures. In *Proceedings of the third annual ACM symposium on Theory of computing*, pages 151–158. ACM, 1971.

[54] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms.* MIT Press, third edition, 2009.

[55] Noel Cressie. Statistics for spatial data. *Terra Nova*, 4(5):613–617, 1992.

[56] David Culler, Deborah Estrin, and Mani Srivastava. Guest editors' introduction: Overview of sensor networks. *Computer*, 37(8):41–49, 2004.

[57] Karthik Dantu and Gaurav S Sukhatme. Detecting and tracking level sets of scalar fields using a robotic sensor network. In *Robotics and Automation, 2007 IEEE International Conference on*, pages 3665–3672, April 2007.

[58] Abhimanyu Das and David Kempe. Algorithms for subset selection in linear regression. In *Proceedings of the fortieth annual ACM symposium on Theory of computing*, pages 45–54. ACM, 2008.

[59] Arnab Das and Bikas K. Chakrabarti. *Colloquium* : Quantum annealing and analog quantum computation. *Rev. Mod. Phys.*, 80:1061–1081, Sep 2008.

[60] Vasil S. Denchev, Sergio Boixo, Sergei V. Isakov, Nan Ding, Ryan Babbush, Vadim Smelyanskiy, John Martinis, and Hartmut Neven. What is the computational value of finite-range tunneling? *Phys. Rev. X*, 6:031015, Aug 2016.

[61] Matteo Denitto, Alessandro Farinelli, Mário AT Figueiredo, and Manuele Bicego. A biclustering approach based on factor graphs and the max-sum algorithm. *Pattern Recognition*, 62:114–124, 2017.

[62] Matteo Denitto, Alessandro Farinelli, Giuditta Franco, and Manuele Bicego. A binary factor graph model for biclustering. In *Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR)*, pages 394–403. Springer, 2014.

[63] Amol Deshpande, Carlos Guestrin, Samuel R Madden, Joseph M Hellerstein, and Wei Hong. Model-driven data acquisition in sensor networks. In *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30*, pages 588–599. VLDB Endowment, 2004.

[64] Inderjit S Dhillon. Co-clustering documents and words using bipartite spectral graph partitioning. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 269–274. ACM, 2001.

[65] Jose Luis Diaz de Leon and Humberto Sossa. Automatic path planning for a mobile robot among obstacles of arbitrary shape. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 28(3):467–472, Jun 1998.

[66] Sara Dolnicar, Sebastian Kaiser, Katie Lazarevski, and Friedrich Leisch. Biclustering: Overcoming data dimensionality problems in market segmentation. *Journal of Travel Research*, 51(1):41–49, 2012.

[67] Matthew Dunbabin and Lino Marques. Robots for environmental monitoring: Significant advancements and applications. *Robotics Automation Magazine, IEEE*, 19(1):24–39, March 2012.

[68] Joseph C Dunn. A fuzzy relative of the isodata process and its use in detecting compact well-separated clusters. 1973.

[69] Edward Farhi, Jeffrey Goldstone, Sam Gutmann, and Michael Sipser. Quantum computation by adiabatic evolution. Technical report, 2000.

[70] Richard P. Feynman. Simulating physics with computers. *International Journal of Theoretical Physics*, 21(6):467–488, Jun 1982.

[71] A.B. Finnila, M.A. Gomez, C. Sebenik, C. Stenson, and J.D. Doll. Quantum annealing: A new method for minimizing multidimensional functions. *Chemical Physics Letters*, 219(5–6):343 – 348, 1994.

[72] Jose L Flores, Iñaki Inza, Pedro Larrañaga, and Borja Calvo. A new measure for gene expression biclustering based on non-parametric correlation. *Computer methods and programs in biomedicine*, 112(3):367–397, 2013.

[73] Brendan J. Frey and Delbert Dueck. Clustering by passing messages between data points. *Science*, 315(5814):972–976, 2007.

[74] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics New York, NY, USA:, 2001.

[75] Hugo Garces and Daniel Sbarbaro. Outliers detection in environmental monitoring databases. *Engineering Applications of Artificial Intelligence*, 24(2):341 – 349, 2011.

[76] Asim Ghosh and Sudip Mukherjee. Quantum annealing and computation : A brief documentary note. 2013.

[77] Bruce L. Golden, Larry Levy, and Rakesh Vohra. The orienteering problem. *Naval Research Logistics (NRL)*, 34(3):307–318, 1987.

[78] Rafael C. Gonzalez and Richard E. Woods. *Digital Image Processing (3rd Edition)*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 2006.

[79] Alkis Gotovos, Nathalie Casati, Gregory Hitz, and Andreas Krause. Active learning for level set estimation. In *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence*, IJCAI '13, pages 1344–1350. AAAI Press, 2013.

[80] Carlos Guestrin, Andreas Krause, and Ajit Paul Singh. Near-optimal sensor placements in Gaussian processes. In *Proceedings of the 22nd international conference on Machine learning*, pages 265–272. ACM, 2005.

[81] Aldy Gunawan, Hoong Chuin Lau, and Pieter Vansteenwegen. Orienteering problem: A survey of recent variants, solution approaches and applications. *European Journal of Operational Research*, 255(2):315 – 332, 2016.

[82] John A Hartigan and Manchek A Wong. Algorithm as 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1):100–108, 1979.

[83] Corporate Headquarters. Programming with d-wave: Map coloring problem, 2013.

[84] Rui Henriques, Cláudia Antunes, and Sara C Madeira. A structured view on pattern mining-based biclustering. *Pattern Recognition*, 48(12):3941–3958, 2015.

[85] Rui Henriques and Sara C Madeira. Bicpam: Pattern-based biclustering for biomedical data analysis. *Algorithms for Molecular Biology*, 9(1):27, 2014.

[86] Rui Henriques and Sara C. Madeira. Bsig: evaluating the statistical significance of biclustering solutions. *Data Mining and Knowledge Discovery*, 32(1):124–161, Jan 2018.

[87] Gregory Hitz, Enric Galceran, Marie-Ève Garneau, François Pomerleau, and Roland Siegwart. Adaptive continuous-space informative path planning for online environmental monitoring. *Journal of Field Robotics*, 34(8):1427–1449, 2017.

[88] Gregory Hitz, Alkis Gotovos, Francois Pomerleau, Marie-Éve Garneau, Cédric Pradalier, Andreas Krause, and Roland Y Siegwart. Fully autonomous focused exploration for robotic environmental monitoring. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 2658–2664, May 2014.

[89] Geoffrey A. Hollinger and Gaurav S. Sukhatme. Sampling-based robotic information gathering algorithms. *Int. J. Rob. Res.*, 33(9):1271–1287, August 2014.

[90] Mark Johnson, Mohammad Amin, S Gildert, Trevor Lanting, F Hamze, N Dickson, R Harris, Andrew Berkley, J Johansson, Paul Bunyk, E M Chapple, C Enderud, Jeremy Hilton, Kamran Karimi, E Ladizinsky, Nicolas Ladizinsky, T Oh, I Perminov, C Rich, and Geordie Rose. Quantum annealing with manufactured spins. *Nature*, 473(7346):194–198, May 2011.

[91] Tadashi Kadowaki and Hidetoshi Nishimori. Quantum annealing in the transverse ising model. *Phys. Rev. E*, 58:5355–5363, Nov 1998.

[92] Seiji Kataoka and Susumu Morito. An algorithm for single constraint maximum collection problem. *Journal of the Operations Research Society of Japan*, 31(4):515–31, 1988.

[93] Kavi K Khedo, Rajiv Perseedoss, Avinash Mungur, et al. A wireless sensor network air pollution monitoring system. *arXiv preprint arXiv:1005.1737*, 2010.

[94] James King, Sheir Yarkoni, Jack Raymond, Isil Ozfidan, Andrew D King, Mayssam Mohammadi Nevisi, Jeremy P Hilton, and Cather-

ine C McGeoch. Quantum annealing amid local ruggedness and global frustration. *arXiv preprint arXiv:1701.04579*, 2017.

[95] James King, Sheir Yarkoni, Jack Raymond, Isil Ozfidan, Andrew D. King, Mayssam Mohammadi Nevisi, Jeremy P. Hilton, and Catherine C. McGeoch. Quantum annealing amid local ruggedness and global frustration. *ArXiv e-prints*, Mar 2017.

[96] Scott Kirkpatrick, C Daniel Gelatt, and Mario P Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.

[97] Christine Klymko, Blair D Sullivan, and Travis S Humble. Adiabatic quantum programming: minor embedding with hard faults. *Quantum Information Processing*, 13(3):709–729, 2014.

[98] Gary Kochenberger, Jin-Kao Hao, Fred Glover, Mark Lewis, Zhipeng Lü, Haibo Wang, and Yang Wang. The unconstrained binary quadratic programming problem: a survey. *Journal of Combinatorial Optimization*, 28(1):58–81, 2014.

[99] Jacob Kogan, Charles Nicholas, Marc Teboulle, et al. *Grouping multidimensional data*. Springer, 2006.

[100] Michail Kontitsis, Evangelos A Theodorou, and Emanuel Todorov. Multi-robot active slam with relative entropy optimization. In *American Control Conference (ACC), 2013*, pages 2757–2764. IEEE, 2013.

[101] Andreas Krause. *Optimizing sensing*. PhD thesis, PhD thesis, Carnegie Mellon University.(Section 5.7), 2008.

[102] Andreas Krause and Daniel Golovin. Submodular function maximization., 2014.

[103] Andreas Krause and Carlos Guestrin. Near-optimal observation selection using submodular functions. In *National Conference on Artificial Intelligence (AAAI), Nectar track*, July 2007.

[104] Andreas Krause, Carlos Guestrin, Anupam Gupta, and Jon Kleinberg. Near-optimal sensor placements: Maximizing information while minimizing communication cost. In *Proceedings of the 5th international conference on Information processing in sensor networks*, pages 2–10. ACM, 2006.

[105] Andreas Krause, Carlos Guestrin, Anupam Gupta, and Jon Kleinberg. Robust sensor placements at informative and communication-efficient locations. *ACM Trans. Sen. Netw.*, 7(4):31:1–31:33, February 2011.

[106] Andreas Krause, Jure Leskovec, Carlos Guestrin, Jeanne VanBriesen, and Christos Faloutsos. Efficient sensor placement optimization for securing large water distribution networks. *Journal of Water Resources Planning and Management*, 134(6):516–526, 2008.

[107] Andreas Krause, H Brendan McMahan, Carlos Guestrin, and Anupam Gupta. Robust submodular observation selection. *Journal of Machine Learning Research*, 9(Dec):2761–2801, 2008.

[108] Andreas Krause, Ajit Singh, and Carlos Guestrin. Near-optimal sensor placements in Gaussian processes: Theory, efficient algorithms and empirical studies. *Journal of Machine Learning Research*, 9(Feb):235–284, 2008.

[109] Vijay Kumar, Daniela Rus, and Sanjiv Singh. Robot and sensor networks for first responders. *IEEE Pervasive computing*, 3(4):24–33, 2004.

[110] Hung M La and Weihua Sheng. Distributed sensor fusion for scalar field mapping using mobile sensor networks. *IEEE Transactions on Cybernetics*, 43(2):766–778, April 2013.

[111] Hung M La, Weihua Sheng, and Jiming Chen. Cooperative and active sensing in mobile sensor networks for scalar field mapping. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 45(1):1–12, Jan 2015.

[112] Jack W Langelaan, Nicholas Alley, and James Neidhoefer. Wind field estimation for small unmanned aerial vehicles. *Journal of Guidance, Control, and Dynamics*, 34(4):1016–1030, 2011.

[113] Gilbert Laporte and Silvano Martello. The selective travelling salesman problem. *Discrete Applied Mathematics*, 26(2):193 – 207, 1990.

[114] Daniel S Levine et al. *Information-rich path planning under general constraints using rapidly-exploring random trees.* PhD thesis, Massachusetts Institute of Technology, 2010.

[115] Zhan Wei Lim, David Hsu, and Wee Sun Lee. Adaptive informative path planning in metric spaces. *The International Journal of Robotics Research*, 35(5):585–598, 2016.

[116] Stuart Lloyd. Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2):129–137, 1982.

[117] Wenjie Lu. *Autonomous sensor path planning and control for active information gathering.* PhD thesis, 2014.

[118] David J.C. MacKay. *Information theory, inference and learning algorithms.* Cambridge university press, 2003.

[119] James MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA, 1967.

[120] Sara C Madeira and Arlindo L Oliveira. Biclustering algorithms for biological data analysis: a survey. *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)*, 1(1):24–45, 2004.

[121] Kirk Martinez, Jane K Hart, and Royan Ong. Environmental sensor networks. *Computer*, 37(8):50–56, 2004.

[122] Michael Marzec. Portfolio optimization: applications in quantum computing. *Handbook of High-Frequency Trading and Modeling in Finance (John Wiley & Sons, Inc., 2016) pp*, pages 73–106, 2016.

[123] James McMahon, Harun Yetkin, Artur Wolek, Zachary J Waters, and Daniel J Stilwell. Towards real-time search planning in subsea environments. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 87–94, Sept 2017.

[124] Anirban Mukhopadhyay, Ujjwal Maulik, Sanghamitra Bandyopadhyay, and Carlos A Coello Coello. Survey of multiobjective evolutionary algorithms for data mining: Part ii. *IEEE Transactions on Evolutionary Computation*, 18(1):20–35, 2014.

[125] Kevin P. Murphy. *Machine Learning: A Probabilistic Perspective.* The MIT Press, 2012.

[126] Nitin Muttil and Kwok-Wing Chau. Machine-learning paradigms for selecting ecologically significant input variables. *Engineering Applications of Artificial Intelligence*, 20(6):735 – 744, 2007.

[127] Masahiko Nagai, Tianen Chen, Ryosuke Shibasaki, Hideo Kumagai, and Afzal Ahmed. Uav-borne 3-d mapping system by multisensor integration. *IEEE Transactions on Geoscience and Remote Sensing*, 47(3):701–708, March 2009.

[128] Nasser M Nasrabadi. Pattern recognition and machine learning. *Journal of electronic imaging*, 16(4):049901, 2007.

[129] George L Nemhauser, Laurence A Wolsey, and Marshall L Fisher. An analysis of approximations for maximizing submodular set functions—i. *Mathematical Programming*, 14(1):265–294, Dec 1978.

[130] Patrick P Neumann, Victor Hernandez Bennetts, Achim J Lilienthal, Matthias Bartholmai, and Jochen H Schiller. Gas source localization with a micro-drone using bio-inspired and particle filter-based algorithms. *Advanced Robotics*, 27(9):725–738, 2013.

[131] Hartmut Neven, Vasil S. Denchev, Geordie Rose, and William G. Macready. Training a binary classifier with the quantum adiabatic algorithm. `http://arxiv.org/abs/0811.0416`, 2008.

[132] Hartmut Neven, Geordie Rose, and William G. Macready. Image recognition with an adiabatic quantum computer I. Mapping to quadratic unconstrained binary optimization. `http://arxiv.org/abs/0804.4457`, 2008.

[133] Jerzy Neyman. *Proceedings of the Fourth Berkeley Symposium on Mathematical Statistics and Probability.* Univ of California Press, 1961.

[134] Ali Oghabian, Sami Kilpinen, Sampsa Hautaniemi, and Elena Czeizler. Biclustering methods: biological relevance and application in gene expression analysis. *PloS one*, 9(3):e90801, 2014.

[135] Bryan O'Gorman, Alejandro Perdomo-Ortiz, Ryan Babbush, Alán Aspuru-Guzik, and V Smelyanskiy. Bayesian network structure learning using quantum annealing. *The European Physical Journal Special Topics*, 224(1):163–188, 2015.

[136] Bryan O'Gorman, Eleanor Rieffel, Minh Do, Davide Venturelli, and Jeremy Frank. Compiling planning into quantum optimization problems: A comparative study. In *Proceedings of the Workshop on Constraint Satisfaction Techniques for Planning and Scheduling Problems (COPLAS-15)*, pages 11–20, 2015.

[137] Anibal Ollero, J Alcázar, F Cuesta, F López-Pichaco, and C Nogales. Helicopter teleoperation for aerial monitoring in the comets multi-uav system. In *3rd IARP Workshop on Service, Assistive and Personal Robots*, 2003.

[138] Avi Ostfeld, James G Uber, Elad Salomons, Jonathan W Berry, William E Hart, Cindy A Phillips, Jean-Paul Watson, Gianluca Dorini, Philip Jonkergouw, Zoran Kapelan, et al. The battle of the water sensor networks (bwsn): A design challenge for engineers and algorithms. *Journal of Water Resources Planning and Management*, 134(6):556–568, 2008.

[139] Shuo Pang and Jay A Farrell. Chemical plume source localization. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 36(5):1068–1080, Oct 2006.

[140] Tatdow Pansombut, William Hendrix, Zekai Jacob Gao, Brent E Harrison, and Nagiza F Samatova. Biclustering-driven ensemble of bayesian belief network classifiers for underdetermined problems. In *IJCAI Proceedings-International Joint Conference on Artificial Intelligence*, volume 22, page 1439, 2011.

[141] Alejandro Perdomo-Ortiz, Neil Dickson, Marshall Drew-Brook, Geordie Rose, and Alán Aspuru-Guzik. Finding low-energy conformations of lattice protein models by quantum annealing. *Scientific Report*, 2(517), 2012.

[142] Alejandro Perdomo-Ortiz, Joseph Fluegemann, Rupak Biswas, and Vadim N. Smelyanskiy. A performance estimator for quantum annealers: Gauge selection and parameter setting. *ArXiv e-prints*, Mar 2015.

[143] Alejandro Perdomo-Ortiz, Joseph Fluegemann, Sriram Narasimhan, Rupak Biswas, and Vadim N Smelyanskiy. A quantum annealing approach for fault detection and diagnosis of graph-based systems. *The European Physical Journal Special Topics*, 224(1):131–148, 2015.

[144] Marija Popovic, Gregory Hitz, Juan I. Nieto, Inkyu Sa, Roland Siegwart, and Enric Galceran. Online informative path planning for active classification using uavs. *CoRR*, abs/1609.08446, 2016.

[145] Thomas Powers, David W Krout, Jeff Bilmes, and Les Atlas. Constrained robust submodular sensor selection with application to multistatic sonar arrays. *IET Radar, Sonar & Navigation*, 11(12):1776–1781, 2017.

[146] Amela Prelić, Stefan Bleuler, Philip Zimmermann, Anja Wille, Peter Bühlmann, Wilhelm Gruissem, Lars Hennig, Lothar Thiele, and Eckart Zitzler. A systematic comparison and evaluation of biclustering methods for gene expression data. *Bioinformatics*, 22(9):1122–1129, 2006.

[147] Kristen L Pudenz. Parameter setting for quantum annealers. In *2016 IEEE High Performance Extreme Computing Conference (HPEC)*, pages 1–6, Sept 2016.

[148] Mohammad Rahimi, Richard Pon, William J Kaiser, Gaurav S Sukhatme, Deborah Estrin, and Mani Srivastava. Adaptive sampling for environmental robotics. In *Robotics and Automation, 2004. Proceedings. ICRA '04. 2004 IEEE International Conference on*, volume 4, pages 3537–3544 Vol.4, April 2004.

[149] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning.* MIT Press, Cambridge, MA, USA, 2006.

[150] Eleanor G Rieffel, Davide Venturelli, Bryan O'Gorman, Minh B Do, Elicia M Prystay, and Vadim N Smelyanskiy. A case study in programming a quantum annealer for hard operational planning problems. *Quantum Information Processing*, 14:1–36, January 2015.

[151] Neil Robertson and P.D. Seymour. Graph minors. xx. wagner's conjecture. *Journal of Combinatorial Theory, Series B*, 92(2):325 – 357, 2004. Special Issue Dedicated to Professor W.T. Tutte.

[152] Gili Rosenberg, Poya Haghnegahdar, Phil Goddard, Peter Carr, Kesheng Wu, and Marcos López De Prado. Solving the optimal trading trajectory problem using a quantum annealer. *IEEE Journal of Selected Topics in Signal Processing*, 10(6):1053–1060, 2016.

[153] Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

[154] Andrew Russell. Comparing search algorithms for robotic underground chemical source location. *Autonomous Robots*, 38(1):49–63, Jan 2015.

[155] Giuseppe E Santoro and Erio Tosatti. Optimization using quantum mechanics: quantum annealing through adiabatic evolution. *Journal of Physics A: Mathematical and General*, 39(36):R393, 2006.

[156] Jürgen Scherer, Saeed Yahyanejad, Samira Hayat, Evsen Yanmaz, Torsten Andre, Asif Khan, Vladimir Vukadinovic, Christian Bettstetter, Hermann Hellwagner, and Bernhard Rinner. An autonomous multi-uav system for search and rescue. In *Proceedings of the First Workshop on Micro Aerial Vehicle Networks, Systems, and Applications for Civilian Use*, DroNet '15, pages 33–38, New York, NY, USA, 2015. ACM.

[157] Maria Schuld, Ilya Sinayskiy, and Francesco Petruccione. An introduction to quantum machine learning. *Contemporary Physics*, 56(2):172–185, 2015.

[158] Seyyed Ali Asghar Shahidian and Hadi Soltanizadeh. Path planning for two unmanned aerial vehicles in passive localization of radio sources. *Aerospace Science and Technology*, 58:189 – 196, 2016.

[159] Robert Sim and Nicholas Roy. Global a-optimal robot exploration in slam. In *Robotics and Automation, 2005. ICRA 2005. Proceedings*

of the 2005 IEEE International Conference on, pages 661–666. IEEE, 2005.

[160] Aarti Singh, Robert Nowak, and Parmesh Ramanathan. Active learning for adaptive mobile sensing networks. In *Proceedings of the 5th International Conference on Information Processing in Sensor Networks*, IPSN '06, pages 60–68, New York, NY, USA, 2006. ACM.

[161] Amarjeet Singh, Andreas Krause, Carlos Guestrin, and William J. Kaiser. Efficient informative sensing using multiple robots. *J. Artif. Int. Res.*, 34(1):707–755, April 2009.

[162] Amarjeet Singh, Andreas Krause, Carlos Guestrin, William J Kaiser, and Maxim A Batalin. Efficient planning of informative paths for multiple robots. In *IJCAI*, volume 7, pages 2204–2211, 2007.

[163] Amarjeet Singh, Andreas Krause, and William J Kaiser. Nonmyopic adaptive informative path planning for multiple robots. In *IJCAI*, volume 3, page 2, 2009.

[164] Vadim N Smelyanskiy, Eleanor G Rieffel, Sergey I Knysh, Colin P Williams, Mark W Johnson, Murray C Thom, William G Macready, and Kristen L Pudenz. A near-term quantum computing approach for hard computational problems in space exploration. `http://arxiv.org/abs/1204.2821`, 2012.

[165] Brendan Smith, Michael Beman, David Gravano, and YangQuan Chen. Development and validation of a microbe detecting uav payload. In *2015 Workshop on Research, Education and Development of Unmanned Aerial Systems (RED-UAS)*, pages 258–264, Nov 2015.

[166] Chih-Li Sunga, Robert B Gramacyb, and Benjamin Haalanda. Exploiting variance reduction potential in local Gaussian process search. *arXiv preprint arXiv:1604.04980*, 2016.

[167] Tommy Thomadsen and Thomas K Stidsen. The quadratic selective travelling salesman problem. Technical report, Informatics and Mathematical Modelling, Technical University of Denmark, DTU, Richard Petersens Plads, Building 305, DK-2800 Kgs. Lyngby, 2003.

[168] Pratap Tokekar, Elliot Branson, Joshua Vander Hook, and Volkan Isler. Tracking aquatic invaders: Autonomous robots for monitoring invasive fish. *IEEE Robotics & Automation Magazine*, 20(3):33–41, 2013.

[169] Pratap Tokekar, Joshua Vander Hook, David Mulla, and Volkan Isler. Sensor planning for a symbiotic UAV and UGV system for precision

agriculture. *IEEE Transactions on Robotics*, 32(6):1498–1511, Dec 2016.

[170] Tony T Tran, Minh Do, Eleanor G Rieffel, Jeremy Frank, Zhihui Wang, Bryan O'Gorman, Davide Venturelli, and J Christopher Beck. A hybrid quantum-classical approach to solving scheduling problems. In *Ninth Annual Symposium on Combinatorial Search*, 2016.

[171] Tony T Tran, Zhihui Wang, Minh Do, Eleanor G Rieffel, Jeremy Frank, Bryan O'Gorman, Davide Venturelli, and J Christopher Beck. Explorations of quantum-classical approaches to scheduling a mars lander activity problem. In *AAAI Workshop: Planning for Hybrid Systems*, 2016.

[172] Duy Tin Truong, Roberto Battiti, and Mauro Brunato. A repeated local search algorithm for biclustering of gene expression data. In Edwin Hancock and Marcello Pelillo, editors, *Similarity-Based Pattern Recognition*, pages 281–296, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[173] Kewei Tu, Xixiu Ouyang, Dingyi Han, Yong Yu, and Vasant Honavar. Exemplar-based robust coherent biclustering. In *Proceedings of the 2011 SIAM International Conference on Data Mining*, pages 884–895. SIAM, 2011.

[174] Vasileios Tzoumas. *Resilient Submodular Maximization for Control and Sensing*. PhD thesis, University of Pennsylvania, 2018.

[175] Pieter Vansteenwegen, Wouter Souffriau, and Dirk Van Oudheusden. The orienteering problem: a survey. *European Journal of Operational Research*, 209(1):1–10, 2011.

[176] Davide Venturelli and Alexei Kondratyev. Reverse quantum annealing approach to portfolio optimization problems. *arXiv preprint arXiv:1810.08584*, 2018.

[177] Davide Venturelli, Salvatore Mandrà, Sergey Knysh, Bryan O'Gorman, Rupak Biswas, and Vadim Smelyanskiy. Quantum optimization of fully connected spin glasses. *Phys. Rev. X*, 5:031040, Sep 2015.

[178] Davide Venturelli, Dominic JJ Marchand, and Galo Rojo. Quantum annealing implementation of job-shop scheduling. *arXiv preprint arXiv:1506.08479*, 2015.

[179] Klaus Wagner. Über eine eigenschaft der ebenen komplexe. *Mathematische Annalen*, 114(1):570–579, 1937.

[180] John Ware and Nicholas Roy. An analysis of wind field estimation and exploitation for quadrotor flight in the urban canopy layer. In *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, pages 1507–1514. IEEE, 2016.

[181] Haitao Xiang and Lei Tian. Development of a low-cost agricultural remote sensing system based on an autonomous unmanned aerial vehicle (uav). *Biosystems Engineering*, 108(2):174 – 190, 2011.

[182] Dongkuan Xu and Yingjie Tian. A comprehensive survey of clustering algorithms. *Annals of Data Science*, 2(2):165–193, 2015.

[183] Rui Xu and D. Wunsch. Survey of clustering algorithms. *IEEE Transactions on Neural Networks*, 16(3):645–678, May 2005.

[184] Jiong Yang, Haixun Wang, Wei Wang, and Philip S Yu. An improved biclustering method for analyzing gene expression profiles. *International Journal on Artificial Intelligence Tools*, 14(05):771–789, 2005.

[185] Yong Zhou and Arthur W Toga. Efficient skeletonization of volumetric objects. *IEEE Transactions on visualization and computer graphics*, 5(3):196, 1999.

Sommario

L'obiettivo del monitoraggio ambientale è quello di raccogliere informazioni dall'ambiente e generare un accurato modello per uno specifico fenomeno di interesse.

Possiamo distinguere le varie applicazioni di monitoraggio ambientale in due macro aree a seconda della strategia di acquisizione dei dati dall'ambiente. Nel primo caso, l'utilizzo di sensori fissi dispiegati nell'ambiente consente di monitorare con un flusso costante di informazioni che provengono da un insieme predeterminato di luoghi nello spazio. Diversamente, l'uso di piattaforme mobili consente di scegliere in modo adattivo e rapido le posizioni di rilevamento in base alle esigenze. Per alcune applicazioni (ad esempio il monitoraggio dell'acqua) ciò può ridurre significativamente i costi associati al monitoraggio rispetto alle classiche analisi effettuate da operatori umani.

Tuttavia, entrambi i casi condividono un problema comune da risolvere. Il processo di raccolta dei dati deve tenere in considerazione risorse limitate e il problema chiave è scegliere dove eseguire osservazioni (misurazioni) al fine di acquisire le informazioni dall'ambiente nel modo più efficace possibile e ridurre l'incertezza sui fenomeni analizzati. Possiamo generalizzare questo concetto sotto il nome di *information gathering.* In generale, massimizzare le informazioni che possiamo ottenere dall'ambiente è un problema NP-hard. Di conseguenza, ottimizzare la selezione dei punti di campionamento diventa cruciale in questo contesto.

Nel caso di sensori mobili il problema di ridurre l'incertezza su un processo fisico richiede di calcolare traiettorie di campionamento vincolate dalle limitate risorse disponibili, come ad esempio la durata della batteria della piattaforma o la potenza di calcolo disponibile a bordo. Questo problema viene solitamente definito come *Informative path planning (IPP).* Nell'altro caso, l'osservazione con una rete di sensori fissi richiede di selezionare in anticipo le posizioni specifiche in cui i sensori devono essere dispiegati. Solitamente, il processo di selezione di un insieme limitato di postazioni informative viene eseguito risolvendo un problema di ottimizzazione combinatoria che modella il processo di raccolta delle informazioni.

Questa tesi si concentra sui problemi sopra citati. Nello specifico, indaghiamo diversi problemi e proponiamo innovativi algoritmi ed euristiche legati all'ottimizzazione delle tecniche di raccolta di informazioni per applicazioni di monitoraggio ambientale, sia nel caso in cui vengano utilizzati sensori mobili che fissi. Inoltre, studiamo anche la possibilità di utilizzare un approccio di computazione quantistica nel contesto dell'ottimizzazione della raccolta di informazioni.