Engineering Applications of Artificial Intelligence 77 (2019) 46–58

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

journal homepage: www.elsevier.com/locate/engappai

Orienteering-based informative path planning for environmental monitoring

Lorenzo Bottarelli *, Manuele Bicego, Jason Blum, Alessandro Farinelli

Department of Computer Science, University of Verona, Italy

ARTICLE INFO

Keywords:
Informative path planning
Mobile sensors
Active learning
Gaussian process
Orienteering

ABSTRACT

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 problem is known as *level set estimation*. 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. In this paper, 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. In this paper we propose two active learning algorithms aimed at facing this issue: specifically, (i) SBOLSE casts informative path planning into an orienteering problem and (ii) 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 show that our approaches can compute informative paths that achieve a high quality classification, while significantly reducing the travel distance and the computation time.

1. 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 the specific phenomena of interest (La and Sheng, 2013; La et al., 2015; Garces and Sbarbaro, 2011). Computational methods are often used to facilitate environmental monitoring, for example Cheng et al. (2003) propose and expert system for the analysis of the water quality in a city. An other 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 (Muttil and Chau, 2007) or the dissolved oxygen (DO) vary across the environment. Most environmental monitoring applications require the collection of large datasets, frequently in harsh conditions. In recent years the use of unmanned vehicles for monitoring spatial phenomena has gained increasing attention (Cao et al., 2013). The monitoring operation of a lake for example, could be performed through the use of autonomous surface vessels (ASVs), or by a heterogeneous system composed of marine, terrestrial and airborne platforms (Dunbabin and Marques, 2012).

When deploying unmanned vehicles for environmental monitoring, the data collection process must consider limited resources such as time, energy and computation power that constrain the operation range of the platforms. The goal is to use a mobile platform with low on-board computation power, such as the one showed in Fig. 1, to generate an accurate model of the environmental phenomena of interest. In this context (Hollinger and Sukhatme, 2014), 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. Therefore, if 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 literature offers different path selection strategies (Singh et al., 2009). Traditional nonadaptive (offline) methods generate the path before any observations are made. In contrast, adaptive (online) methods plan the path based on the previously collected data (Batalin et al., 2004; Rahimi et al., 2004; Singh et al., 2006). These adaptive techniques incrementally generate the model of the environmental phenomena of interest during the data collection phase and focus the information

* Corresponding author.
E-mail addresses: Lorenzo.Bottarelli@univr.it (L. Bottarelli), Manuele.Bicego@univr.it (M. Bicego), Jason.Blum@univr.it (J. Blum), Alessandro.Farinelli@univr.it (A. Farinelli).

https://doi.org/10.1016/j.engappai.2018.09.015
Received 13 October 2017; Received in revised form 14 September 2018; Accepted 19 September 2018
Available online 9 October 2018

Fig. 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.

collection process on specific regions of the environment where the phenomena exhibits critical values. For example, in a lake such a region 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 (Pang and Farrell, 2006). 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 (Hitz et al., 2014).

Previous work on the level set estimation problem such as the one proposed by Dantu and Sukhatme (2007) focused on a network composed by a combination of static and mobile sensors. In the manuscript of Gotovos et al. (2013) 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. (2014) 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 from Gotovos et al. (2013), 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., 2017) 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.

This paper is inserted in the aforementioned scenario, and aims at facing the problem of level set estimation by using Active Learning (AL) techniques with sequential measurements. In a general discussion on active learning Liu et al. (2009) present the use of active learning techniques on spatial data where the cost is proportional to the distance traveled, ignoring the intermediate points along the path. In contrast, we have an additional objective, where we aim also at determining efficient paths for mobile sensors (instead of determining single sampling locations) so to optimize the data collection 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 Fig. 1. This platform is small (about 1 meter long and 50 cm wide) and it is equipped with various probes that can measure parameters such as pH, dissolved oxygen, temperature, and electrical conductivity with sampling rate between 1 and 10 Hz. 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. 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.

We introduce a novel algorithm (SBOLSE) that makes use of an orienteering problem formulation for the level set estimation. 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 LSE 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. Finally, we also introduce a novel greedy path selection procedure (PULSE) which represents a baseline greedy strategy for comparisons.

Specifically, the main contributions[1] of this paper 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 propose a novel greedy algorithm called PULSE for selecting measurement paths that exploits a less accurate but computationally faster path selection procedure. PULSE only accounts for the presence of information, not the magnitude of information gain. It is used as a baseline strategy for comparisons in the continuous-sampling setting.
- We test our algorithms on a real world dataset of water pH level and on synthetic datasets extracted from CO_2 maps. 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 sensors 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 (Scherer et al., 2015), precision agriculture (Tokekar et al., 2016; Popovic et al., 2016), sea-floor target localization (McMahon et al., 2017) and radio signal source localization (Shahidian and Soltanizadeh, 2016).

2. Problem statement and background

2.1. Problem statement

Following Gotovos et al. (2013), Bottarelli et al. (2016) and Bottarelli et al. (2017), we formalize the level set estimation as an active learning

[1] Aspects of this work have already been presented in the conference papers (Bottarelli et al., 2016) and (Bottarelli et al., 2017).

problem, where we want to select a path for a mobile sensor so as to optimize the information gathering process.

An unknown scalar field represents the environmental phenomena of interest, and every location in space has an associated scalar value. More formally, given a set of locations $D \subseteq \mathbb{R}^d$ 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 locations $x \in D$ into either the superlevel set $H = \{x \mid f(x) > h\}$ or the sublevel set $L = \{x \mid f(x) \leq h\}$. The problem is defined as the selection of the set of locations x_i where to perform (noisy) measurements $y_i = f(x_i) + e_i$. Such locations should be selected to maximize the classification accuracy of all points, while minimizing the total traveled distance required for the sensor to analyze these locations.

2.2. The LSE algorithm (Gotovos et al., 2013)

In this section we summarize the method of Gotovos et al. (2013), which represents the starting point for our techniques. Gotovos et al. (2013) called this technique LSE. This approach is based on Gaussian Processes (GP), a technique which offers a way to model unknown functions without using parameters. Such tools are widely used in machine learning (Rasmussen and Williams, 2006; Kim and Kim, 2013; Ycel et al., 2013). In the paper proposed by Gotovos et al. (2013), the unknown function to be modeled using a GP is the unknown scalar field f of the environmental phenomena of interest. Briefly, a GP is defined by a mean function $\mu(x)$ (that can be assumed to be zero without loss of generality) and by a kernel function (covariance function) $k(x, x')$ which represents the smoothness properties of the modeled function. A GP can then be denoted as $\mathcal{GP}(\mu(x), k(x, x'))$.

Authors in Gotovos et al. (2013) consider a set of noisy measurements $Y_t = \{y_1, y_2, \cdots, y_t\}$ taken at locations $X_t = \{x_1, x_2, \ldots, x_t\}$ and assume that $y_i = f(x_i) + e_i$ where $e_i \sim \mathcal{N}(0, \sigma_n^2)$ (i.e., measurement noise with zero mean). Given the GP prior $\mathcal{GP}(0, k(x, x'))$, the posterior over f is still a GP and its mean and variance can be computed as follows (Rasmussen and Williams, 2006):

$$\mu_t(x) = \mathbf{k}_t(x)^T (\mathbf{K}_t + \sigma_n^2 \mathbf{I})^{-1} Y_t \tag{1}$$

$$\sigma_t^2(x) = k(x, x) - \mathbf{k}_t(x)^T (\mathbf{K}_t + \sigma^2 \mathbf{I})^{-1} \mathbf{k}_t(x) \tag{2}$$

where $\mathbf{k}_t(x) = [k(x_1, x), \ldots, k(x_t, x)]^T$ and $\mathbf{K}_t = [k(x, x')]_{x,x' \in X_t}$

Using these equations, the GP is built with the new measurements acquired by the sensor. However, in practical applications, the update of the posterior is computationally expensive as it requires inverting an $n \times n$ matrix. n is the number of the samples acquired, which can be thousands of elements in real-world applications. Hence, it is crucial to reduce the frequency of this computation.

Given the region of interest, Gotovos et al. (2013) discretize it into a grid where each element represents a small portion of the surface. These elements compose the set of sample locations (points) D, and the goal is to classify each location $x_i \in D$ into two sets H or L with respect to a threshold level h. The LSE algorithm uses the inferred mean (1) and variance (2) from the GP to construct an interval:

$$Q_t(x) = \left[\mu_{t-1}(x) \pm \beta_t^{1/2} \sigma_{t-1}(x)\right] \tag{3}$$

for any $x \in D$. 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 Gotovos et al. (2013).

Then, in order to classify every point x into H or L, authors in Gotovos et al. (2013) define the following confidence interval using the intersection of all previous $Q_t(x)$ intervals for every point x:

$$C_t(x) = \bigcap_{i=1}^{t} Q_i(x) \tag{4}$$

The classification of a point x depends on the position of its confidence interval with respect to the threshold level h. Specifically, for each location $x \in D$ if its confidence interval $C_t(x)$ lies entirely above h, then $f(x) > h$ with high probability, and we can classify x into the superlevel set H. Similarly, when the entire $C_t(x)$ lies below h then we can classify x into the sublevel set L. These conditions are relaxed with an accuracy parameter ϵ (introduced in Gotovos et al. (2013)) as shown in the following equations:

$$H_t = \{x \mid min(C_t(x)) + \epsilon > h\} \tag{5}$$

$$L_t = \{x \mid max(C_t(x)) - \epsilon \leq h\} \tag{6}$$

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:

$$U_t = D \setminus (L_t \cup H_t) \tag{7}$$

In order to classify the points in U_t according to the Eqs. (5) and (6), it is necessary to acquire more data by selecting new sampling locations $x_i \in U_t$. To this end, the algorithm at each iteration uses the confidence interval for each unclassified point to derive the following ambiguity value:

$$a_t(x) = min\{max(C_t(x)) - h, h - min(C_t(x))\} \tag{8}$$

The point x_t with the highest ambiguity value represents the location with the highest information content. As such, it becomes the next point to measure.

In addition to the LSE algorithm, Gotovos et al. (2013) discuss the batch version where multiple locations are selected by taking mutual information into account. Although the main goal of their approach is to select multiple locations and to compute an efficient path between them, in both cases their assumption is that the process of acquiring a new point of data is costly. Therefore their main goal is to minimize the number of sampling locations. Moreover, during the movement of the mobile agent from one location to next, the agent does not acquire any further data. We will see in this paper how GP-based active learning techniques can be derived to explicitly consider the scenario of continuous data sampling.

2.3. Orienteering

The Orienteering Problem (OP) originates from the sport game of orienteering. In the orienteering game, the start and end points are specified along with a set of other locations (i.e., 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 model several different contexts. For example, consider the problem in which a traveling salesperson has a set of cities which he/she could visit. 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).

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 index = 1, an ending location index = 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.

The OP can easily be defined using a weighted undirected graph $G = (V, E)$ where $V = \{v_1, \ldots, v_N\}$ is the set of locations (nodes) and E is the set of edges. 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 $e_{ij} \in E$. 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.

Therefore, the OP is a combination of node selection and shortest path computation between the graphs' nodes, hence it can be cast as a

combination of the Traveling Salesman Problem (TSP) problems (Cormen et al., 2009) 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) (Golden et al., 1987). The OP is known to be an NP-hard problem, as it contains the well known traveling salesman problem as a special case.

This NP-hard problem arises in scheduling and routing applications, and it is also known as the selective traveling salesperson problem (Laporte and Martello, 1990; Thomadsen and Stidsen, 2003) or the maximum collection problem (Kataoka and Morito, 1988). 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 OP 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 we suggest the survey proposed by Vansteenwegen et al. (2011).

2.4. Topological skeletonization

In shape analysis and digital image processing, *skeletonization* is a process for reducing regions of an image to a thin (skeletal) representation while erasing most of the original pixels (see example in Fig. 2). 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 Blum (1967). 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* (Gonzalez and Woods, 2006).

Skeletonization is used in several applications such as digital image processing, computer vision or path planning for a mobile robot among obstacles (de Leon and A., 1998). 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. In this paper we are interested in the skeletonization process to reduce the number of points that we must consider when planning the path for the robotic platforms.

2.5. Exemplar based clustering with affinity propagation

Clustering aims at partitioning a set of objects into groups (or clusters) based on the concept of similarity. Objects in the same group should be similar, whereas objects belonging to different groups should be dissimilar. Clustering is the subject of active research in several fields such as statistics, pattern recognition, and machine learning (Xu and Tian, 2015).

In this work, we are interested in clustering to reduce the computation required by our algorithm, and we make use of a powerful clustering technique, called Affinity Propagation (Frey and Dueck, 2007). This technique faces the problem of clustering by assuming that each cluster is represented by one object, called an exemplar. All points of the problem have to choose one exemplar (i.e. a representative). Points which have chosen the same exemplar are in the same cluster. The choice is estimated by recursively exchanging messages between points, until a good set of exemplars emerges. The efficiency and accuracy of this algorithm have been shown in different applications (Frey and Dueck, 2007). Moreover, a useful property of Affinity Propagation is that it does not need a specified number of clusters beforehand; clusters emerges spontaneously from data.

Specifically, 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 a negative squared error. In our context, the error is the Euclidean distance: for points x_i and x_k, $s(i,k) = -\|x_i - x_k\|^2$. Alternatively, when appropriate for some applications, similarities may be set by hand. As previously mentioned, the algorithm does not require an explicit number of clusters. This 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.

3. SBOLSE algorithm

Even if the LSE solutions proposed by Gotovos et al. (2013) 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. For example, the mobile platforms we use here are equipped with probes that measure various parameters (e.g., the DO or the PH level) with a given frequency while the platform is moving. In this scenario, our goal is then minimizing the total path length while collecting as much information as possible to correctly classify all locations $x_i \in D$.

In what follows we present our Skeleton-Based Orienteering for Level Set Estimation (SBOLSE) algorithm. 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 it is necessary to optimize the total path of the mobile platform to minimize battery consumption.

The proposed algorithm is based on a Gaussian Processes modeling of the scalar field and considers the knowledge about unclassified locations $x_i \in U_t$ 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 OP, 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.

Algorithm 1 SBOLSE algorithm

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

1: $t \leftarrow 0$
2: $x_0 \leftarrow x_{start}$
3: $H_0 \leftarrow \emptyset, L_0 \leftarrow \emptyset, U_0 \leftarrow D$
4: **while** $H_t \cup L_t \neq D$ **do**
5: $t \leftarrow t + 1$
6: Compute GP posterior $\mu(x)$ and $\sigma^2(x)$ for all $x \in U_t$
7: Classify and update H_t, L_t, U_t according to LSE (Gotovos et al., 2013)
8: $x_c \leftarrow$ current position
9: $G \leftarrow buildGraph(x_c, U_t)$
10: $path \leftarrow orienteeringStep(G, budget)$
11: Execute $path$
12: **end while**
13: $H \leftarrow H_t, L \leftarrow L_t$

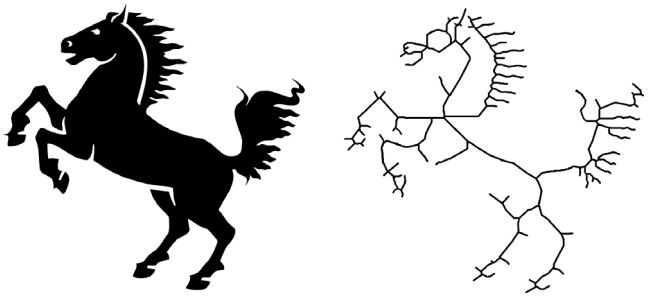

Fig. 2. Example of a topological skeletonization applied to an image.

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 $C_t(x)$ for each point $x \in U_t(x)$, 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 Eq. (8) 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 $H_t \cup L_t = D$, 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 (Gotovos et al., 2013), these are evaluated and possibly re-classified considering newly acquired data.

3.1. Building the graph

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.

Algorithm 2 buildGraph procedure

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

1: $V \leftarrow v_1 \equiv x_c$
2: $w(v_1) \leftarrow 0$
3: $n \leftarrow 1$
4: **for all** $x_i \in U_t$ **do**
5: $n \leftarrow n + 1$
6: $V \leftarrow V \cup v_n \equiv x_i$
7: $w(v_n) \leftarrow a(x_i)$
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: $G \leftarrow (V, E)$

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 (Eq. (8)) 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.

3.2. Orienteering step

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 OP). 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 OP 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 OP 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 GP 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.

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**

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 3.4. 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 OP 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.

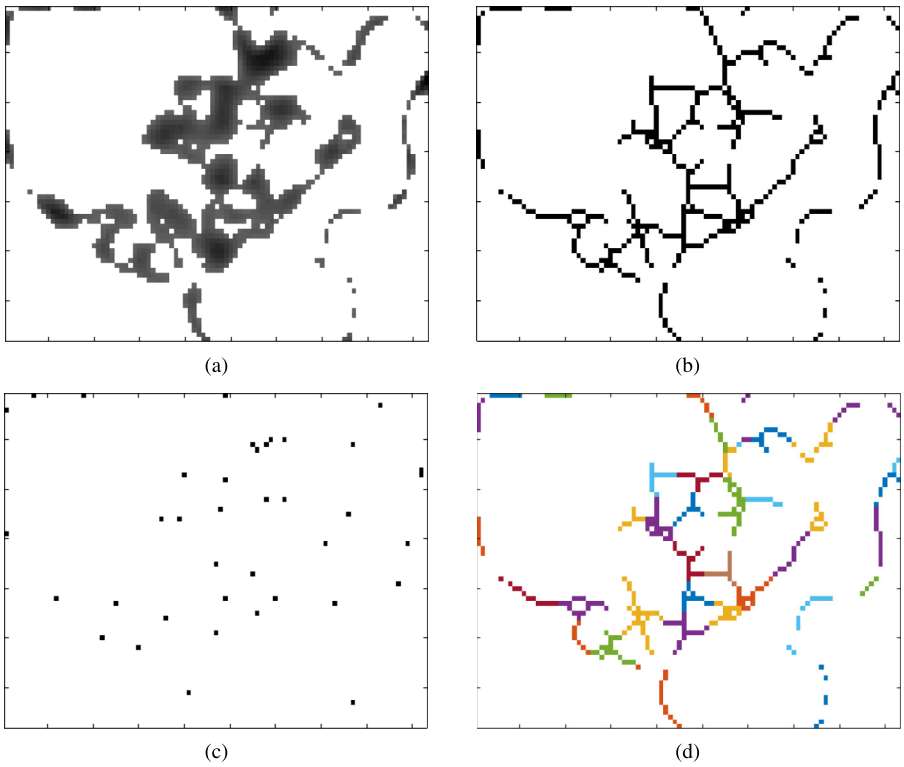

Fig. 3. Example of the topological skeletonization and Exemplar Based Clustering heuristic applied to the data matrix containing the ambiguity measure for the unclassified points U_t. 3a data matrix before the skeletonization, a darker color corresponds to an higher value of ambiguity. 3b matrix after the skeletonization operation. 3c Exemplars selected with the EBC heuristic and 3d the corresponding clusters identified with different colors.

3.3. 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 (Eq. (8)) 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 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 3.1 (see an example in Figs. 3a and 3b). Note that in the *buildGraph* procedure each point that is maintained after the skeletonization represents a node of a complete graph.

3.4. 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 ← ∅
2:  bestPathValue ← 0
3:  V' ← heuristic(V)
4:  for i in range(2, |V'|) do
5:    if ||v_1 − v_i|| ≤ budget then
6:      path ← orienteeringHeuristic(G, v_1, v_i, B)
7:      if value(path) > bestPathValue then
8:        bestPath ← path
9:        bestPathValue ← value(path)
10:     end if
11:   end if
12: end for
```

In what follows we propose the main heuristic that we implemented and used in order to determine the new locations set V'. Three additional baseline heuristics can be found in the Appendix A.

3.4.1. EBC heuristic

The main heuristic we propose is based on the Exemplar Based Clustering (EBC) performed with the Affinity Propagation technique (Frey and Dueck, 2007) introduced in Section 2.5. 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 2.5 the Affinity Propagation procedure takes as input a set of real-value

similarities between points and a set of real numbers 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. More details on how we set these measures in our experiments can be found in Section 5. An example of the effects of the Exemplar Base Clustering phase on the real dataset can be observe in Figs. 3c and 3d.

3.5. Theoretical analysis

For what concerns the theoretical analysis of our approach, notice that (Gotovos et al., 2013) 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 us 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 $|D|$ times. The complexity of the body of the loop is the sum of four main components: (i) the computation of the Gaussian Process that requires $\mathcal{O}(|D|^3)$ due to the need to invert a $|D| \times |D|$ matrix. (ii) The execution of the *buildGraph* sub-procedure which has a complexity $\mathcal{O}(|U_t|^2)$. The cardinality of set U_t is $|D|$ at the first iteration and decreases over time according to how many points have been classified. (iii) The classification according to LSE (Gotovos et al., 2013) requires $\mathcal{O}(|D|)$. (iv) The execution of the *orienteeringStep* sub-procedure. The complexity of this step 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 (Chekuri et al., 2012). As previously mentioned in Section 3.4 the major bottleneck is the multiple executions of the orienteering step that, in the worst case scenario, are $|D| - t$. Even with a less efficient implementation of the orienteering heuristic (e.g. a $\mathcal{O}(|D|^3)$ the execution of the orienteeringStep sub-procedure is on the order of $\mathcal{O}(|D|^4)$.

Hence, the combined complexity of the algorithm is on the order of $\mathcal{O}(|D|^5)$. 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 experiments in Section 5, we detail the actual computation time required to perform the technique using two different datasets.

4. PULSE algorithm

In what follows we present another algorithm, inspired by LSE, which we call Path-Update LSE (PULSE) algorithm. As with the previous SBOLSE algorithm, this is specifically designed for continuous sampling sensors for which (i) the cost required to perform an individual measurement is negligible, (ii) it is necessary to optimize the total path of the agent in order to reduce the battery consumption and (iii) we need an 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 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 tradeoff 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. We use this technique as a comparison for SBOLSE, showing that the orienteering formulation is an important factor for informative path planning.

Algorithm 5 PULSE algorithm

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

1: $t \leftarrow 0$
2: $x_0 \leftarrow x_{start}$
3: $H_0 \leftarrow \emptyset, L_0 \leftarrow \emptyset, U_0 \leftarrow D$
4: **while** $H_t \cup L_t \neq D$ **do**
5: $t \leftarrow t + 1$
6: Compute GP posterior $\mu(x)$ and $\sigma^2(x)$ for all $x \in U_t$
7: Classify and update H_t, L_t, U_t according to LSE (Gotovos et al., 2013)
8: $x_{t-1} \leftarrow x_t$
9: $x_t \leftarrow$next location according to LSE (Gotovos et al., 2013)
10: $path \leftarrow$ pathSelection(x_{t-1}, x_t, U)
11: Execute $path$
12: **end while**
13: $H \leftarrow H_t, L \leftarrow L_t$

Algorithm 6 pathSelection procedure

Input: last position x_{t-1}, next location x_t, unclassified elements U_t
Output: $path$

1: $i \leftarrow 0$
2: $x_{next_0} \leftarrow x_{t-1}$
3: $path \leftarrow x_{t-1}$
4: **while** $x_{next_i} \neq x_t$ **do**
5: $i \leftarrow i + 1$
6: $d \leftarrow ||x_{next_{i-1}} - x_t||$
7: $A \leftarrow \emptyset$
8: **for all** $x \in U_t$ **do**
9: **if** $||x - x_t|| < d$ **then**
10: $A \leftarrow A \cup x$
11: **end if**
12: **end for**
13: $x_{next_i} \leftarrow min_{x \in A}\left(||x_i - x_{next_{i-1}}||\right)$
14: $path \leftarrow path \cup x_{next_i}$
15: **end while**

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 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 $C_t(x)$ for each point $x \in U_t(x)$, classify them in one of the three sets and then compute the next sample to be evaluated using the ambiguity defined by Eq. (8) (line 7). We then compute a path between the current location x_{t-1} and the selected point x_t using the path selection procedure (Algorithm 6). The algorithm terminates when $H_t \cup L_t = D$, 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.

4.1. Path selection

At each time step t the algorithm keeps track of the starting position 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 x_t. In order to select an informative path towards the destination, the path selection procedure analyzes each point $x \in U_t$, i.e. locations that still have to be classified and therefore potentially carrying some useful information, selecting a path $\{x_{t-1} = x_{next_0}, x_{next_1}, \dots, x_{next_n} = x_t\}$ 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 x_{next_i} point determined by the procedure meets the condition to always approach the destination point, i.e.

$$\|x_{next_i} - x_t\| < \|x_{next_{i-1}} - x_t\| \tag{9}$$

where $\|x' - x''\|$ is the Euclidean distance between locations x' and x''. In more detail, given the two points $x_{next_{i-1}}$ and x_t, the region of the space which contains points meeting this condition defines a convex area (see example in Fig. 4) and we call this area A_{t_i} (Algorithm 6, lines 8–10). The procedure analyzes all points $x_i \in U_t \cap A_{t_i}$ and selects as 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 Gotovos et al. (2013) only in the path selection procedure we employ. Specifically, our path selection procedure selects only points that meet the condition in Eq. (9) (Algorithm 6, lines 6 and 9). As we build the path from x_{t-1} to x_t the area A_{t_i} shrinks and converges towards the destination point x_t. This allows the path to include informative points that lie inside this area (example in Fig. 4), while ensuring that the path is going towards the most interesting point defined by the ambiguity measure $a_t(x)$ introduced by Gotovos et al. (2013). For what concerns the convergence analysis of this approach the same argument made in Section 3.5 is valid.

4.2. Batch variant

Here we describe a variant of the PULSE algorithm which is aimed 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 computation time required and the path's efficiency. Following Gotovos et al. (2013), we exploit the fact that the updated predictive variance in Eq. (2) depends only on the location of a measurement, not on the measurement value. 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 other point $x_i \in U_t$. This process is repeated adding the location with the new highest ambiguity to a set.

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) (Applegate et al., 2007). 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. This algorithm allows us to trade off adaptivity in favor of a reduction of the total traveled distance required to classify all points $x \in D$.

5. Experiments

In this section we present the empirical evaluation of our proposed techniques, comparing them with literature alternatives on two datasets, analyzing different aspects of the approaches. More in detail, in Section 5.1 we describe the datasets we used in our experiments; in Section 5.2 we present the comparison between all of our algorithms (as well as state of the art competitors) on a real world dataset; and finally, in Section 5.3, we present the results on a synthetic dataset. We employed this second dataset to have experiments on larger instances.

The algorithms we compare are the following:

- **PULSE:** Our algorithm as explained in Section 4.
- **PULSE**$_{bXX}$: This algorithm is the batch variant of PULSE as described in Section 4.2.
- **CS:** This is a variant of LSE as described by Gotovos et al. (2013) 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 Gotovos et al. (2013) for the continuous measuring setting.
- **SBOLSE:** Our algorithm described in Section 3.
- **SB-EBC:** SBOLSE algorithm with Exemplar Based Clustering heuristic as described in Section 3.4.1.
- **ARS-CIPP**$_n$: This is the adaptive re-planning scheme algorithm proposed by Hitz et al. (2017). The number n represents the number of control points of the B-spline that is optimized by the algorithm.

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

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. (1987). 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).

For the SBOLSE algorithm with the EBC heuristic we set the measures required by the affinity propagation algorithm 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 k and point i as $s(k,i) = (maxDist - \|k-i\|)$. 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 k with the ambiguity measure of that location, that is $s(k,k) = a_t(k)$. The set of preferences is then normalized so as to have all values between 0 and 1.

We performed the skeletonization with a basic technique, based on morphological operators, as implemented in the MATLAB function `bwmorph`.

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. As previously done by Gotovos et al. (2013), Bottarelli et al. (2016) and Bottarelli et al. (2017), 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.

5.1. Datasets

The real dataset consists of measurements of the pH level extracted from waters of the Persian Gulf near Doha, Qatar using the boat in Fig. 1. The data forms a 68×93 grid where each element represents a sampling location 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

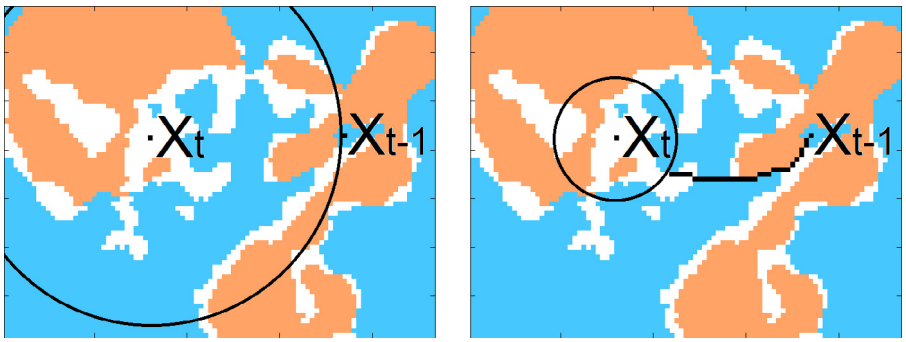

Fig. 4. Example of runtime execution of the path selection procedure. The white areas represent location that are still unclassified. The circle represents the area A. On top the beginning of the procedure and on bottom we can observe the path that has been built after some iterations.

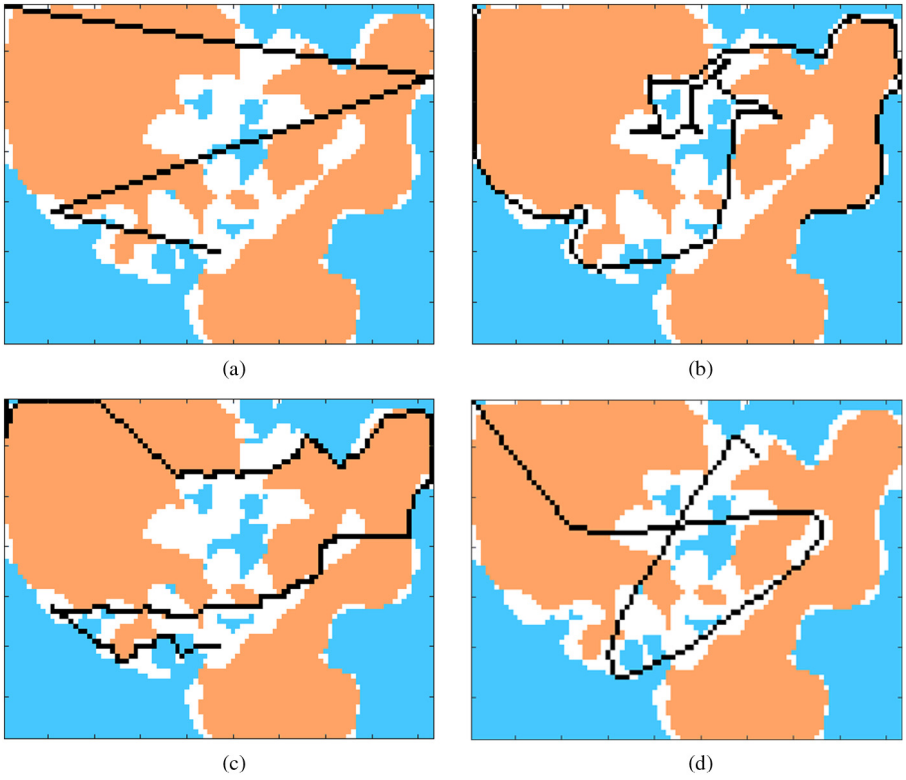

Fig. 5. Real dataset experiments. The white areas represent location that are still unclassified and black lines display a portion of the path selected by the algorithms: (a) CS, (b) SBOLSE, (c) PULSE and (d) ARS-CIPP.

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 (Rasmussen and Williams, 2006) kernel function.

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. The dataset has been extracted from portions of CO_2 maps[2] 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 (Rasmussen and Williams, 2006) kernel function. With five priors per grid and ten grids, we performed a total of 50 tests with each algorithm.

5.2. Real data experiments

For what concerns the real dataset, as done in previous approaches (Gotovos et al., 2013; Bottarelli et al., 2016, 2017) we performed tests to determine the β and ϵ parameter values that allow a high accuracy for all the algorithms. 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 Appendix C Table C.5.

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

Table 1
F_1-score, total traveled distance (meters) and computation time (seconds) using the real world pH dataset. $\bar{x}$ is the average of all 30 experiments and $SE_{\bar{x}}$ is the standard error of the mean.

	F_1-score		Traveled dist. (m)		Comp. time (s)	
	$\bar{x}$	$SE_{\bar{x}}$	$\bar{x}$	$SE_{\bar{x}}$	$\bar{x}$	$SE_{\bar{x}}$
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

Table 2
F_1-score, total traveled distance (meters) and computation time (seconds) using the synthetic CO_2 dataset, $\bar{x}$ is the average of all 50 experiments and $SE_{\bar{x}}$ is the standard error of the mean.

	F_1-score		Traveled dist. (m)		Comp. time (s)	
	$\bar{x}$	$SE_{\bar{x}}$	$\bar{x}$	$SE_{\bar{x}}$	$\bar{x}$	$SE_{\bar{x}}$
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-$CIPP_7$	98.25	0.089	2616.0	63.44	192.9	4.75

As we can observe in Table 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 other techniques but with the higher computation time required. 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 5 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 Fig. 5.

Notice that, results reported in Table 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. 6a 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 m 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.

5.3. 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 experiments on the synthetic dataset are shown in Table 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 m 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 5 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 (−95.5%), or using the PULSE batch algorithm with a comparable path length and computation time.

Similarly to what explained in Section 5.2, 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. 6b 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 m 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-$CIPP_7$ that obtains an F_1-score of 97.55.

6. Conclusions

In this paper 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. The different variants are able to obtain a high quality classification with a shorter path and a lower computation time compared to the current stat of the art algorithms for the level set estimation problem. As future work we will address the multi agent case for the level set estimation problem. Team orienteering problem techniques should offer a viable solution in this scenario.

Acknowledgments

This work was supported by the European Union's Horizon 2020 research and innovation programme, Italy under grant agreement No. 689341. This work reflects only the authors' view and the EASME is not responsible for any use that may be made of the information it contains.

Appendix A. 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 point 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.

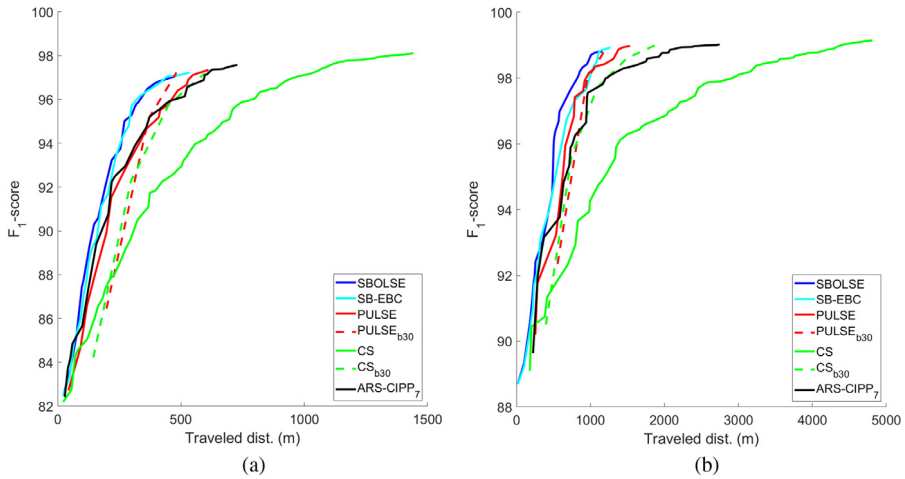

Fig. 6. Evolution of the F_1-score as a function of the distance traveled: (a) on a real-world instance; (b) on a synthetic instance.

Table B.3
Average time (seconds) on the real-world and synthetic datasets varying the Random, Sparse and Sample heuristics' parameter.

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

Table B.4
Average traveled distance (meters) on the real-world and synthetic datasets varying the Random, Sparse and Sample heuristics' parameter.

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

Appendix B. 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 on both the real world dataset and the synthetic datasets described in Section 5.1. We identify the different techniques as follows::

- **SB-Random**(p): SBOLSE algorithm with random sparsification heuristic as described in Appendix A.
- **SB-Sparse**(p): SBOLSE algorithm with lowest data point sparsification heuristic as described in Appendix A.
- **SB-Sample**(p): SBOLSE algorithm with discrete random sampling heuristic as described in Appendix A.

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.

B.1. End-point heuristics results

We obtained a significant reduction in computation time on the real world dataset. As we can observe on Table B.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 B.4 that we have a

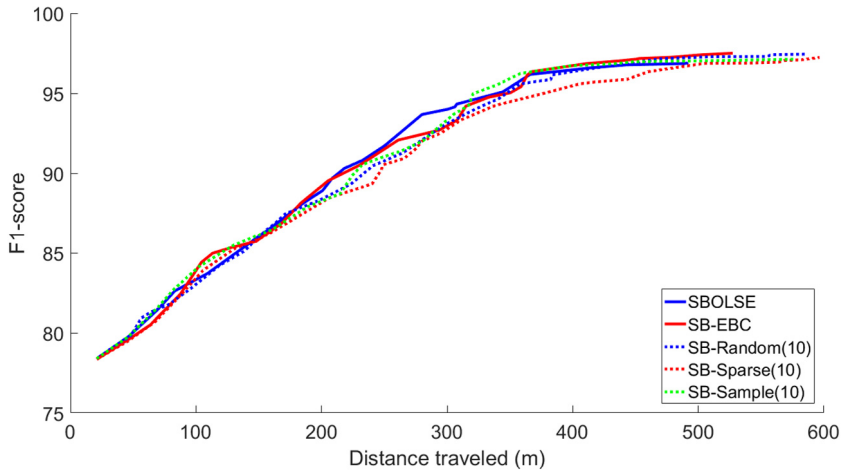

Fig. B.7. Runtime F_1-score comparison on the typical example instance of the real dataset, varying the path length between SBOLSE, SB-EBC, SB-Random, SB-Sparse, and SB-Sample algorithms.

Table C.5
F_1-score, total traveled distance (meters) and computation time (seconds) using the real world pH dataset. $\bar{x}$ is the average of all 30 experiments and $SE_{\bar{x}}$ is the standard error of the mean.

	F_1-score		Traveled dist. (m)		Comp. time (s)	
	$\bar{x}$	$SE_{\bar{x}}$	$\bar{x}$	$SE_{\bar{x}}$	$\bar{x}$	$SE_{\bar{x}}$
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
$ARS\text{-}CIPP_3$	97.60	0.058	899.2	13.44	147.7	2.57
$ARS\text{-}CIPP_7$	97.57	0.045	736.1	10.30	114.5	1.70
$ARS\text{-}CIPP_{11}$	97.64	0.054	812.9	11.95	150.4	2.69
$ARS\text{-}CIPP_{15}$	97.68	0.050	895.8	15.36	187.4	4.18
$ARS\text{-}CIPP_{19}$	97.75	0.056	962.1	11.05	227.7	3.70

Table C.6
F_1-score, total traveled distance (meters) and computation time (seconds) using the synthetic CO_2 dataset, $\bar{x}$ is the average of all 50 experiments and $SE_{\bar{x}}$ is the standard error of the mean.

	F_1-score		Traveled dist. (m)		Comp. time (s)	
	$\bar{x}$	$SE_{\bar{x}}$	$\bar{x}$	$SE_{\bar{x}}$	$\bar{x}$	$SE_{\bar{x}}$
PULSE	98.22	0.090	1709.4	35.37	**23.9**	.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
$ARS\text{-}CIPP_3$	98.23	0.082	3059.0	75.11	323.7	9.96
$ARS\text{-}CIPP_7$	98.25	0.089	2616.0	63.44	192.9	4.75
$ARS\text{-}CIPP_{11}$	98.28	0.089	2777.4	62.30	218.2	5.02
$ARS\text{-}CIPP_{15}$	98.32	0.082	3052.7	61.33	265.0	5.89
$ARS\text{-}CIPP_{19}$	98.39	0.075	3433.2	74.52	319.6	7.43

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. B.7. 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.

For the synthetic dataset we obtained an even greater reduction in computation time. As we can observe on Table B.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 B.4). Specifically, with only 10% of the points selected we obtained an increase of roughly 21%–23% with the three heuristics.

Appendix C. Complete results

For completeness, in Tables C.5 and C.6 we report the same results of Tables 1 and 2 with the additional heuristics presented in Appendix A. Regarding the SB-Random, SB-Sparse and SB-Sample we report results with parameter $p = 10$ because it represents a good tradeoff between time and path length. Moreover, in the following tables we also present the complete experiments performed with the technique proposed by Hitz et al. (2017).

References

Applegate, D.L., Bixby, R.E., Chvatal, V., Cook, W.J., 2007. The traveling salesman problem: a computational study. In: Princeton Series in Applied Mathematics, Princeton University Press, Princeton, NJ, USA.

Batalin, M.A., Rahimi, M., Yu, Y., Liu, D., Kansal, A., Sukhatme, G.S., Kaiser, W.J., Hansen, M., Pottie, G.J., Srivastava, M., Estrin, D., 2004. Proceedings of the 2Nd International Conference on Embedded Networked Sensor Systems. In: SenSys '04, ACM, New York, NY, USA, pp. 25–38.

Blum, H., 1967. In: Wathen-Dunn, W. (Ed.), Models Percept. Speech Visual Form 362–380.

Bottarelli, L., Bicego, M., Blum, J., Farinelli, A., 2016. 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). pp. 1256–1264.

Bottarelli, L., Blum, J., Bicego, M., Farinelli, A., 2017. Proceedings of the Symposium on Applied Computing. In: SAC '17, ACM, New York, NY, USA, pp. 262–267.

Cao, N., Low, K.H., Dolan, J.M., 2013. Proceedings of the 2013 International Conference on Autonomous Agents and Multi-agent Systems. In: AAMAS '13, International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, pp. 7–14.

Chekuri, C., Korula, N., Pál, M., 2012. ACM Trans. Algorithms 8 (3), 23:1–23:27.

Cheng, H., Yang, Z., Chan, C.W., 2003. Eng. Appl. Artif. Intell. 16 (2), 159–166, Applications of Artificial Intelligence for Management and Control of Pollution Minimization and Mitigation Processes.

Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C., 2009. Introduction to Algorithms, Third MIT Press.

Dantu, K., Sukhatme, G., 2007. Robotics and Automation, 2007 IEEE International Conference on. pp. 3665–3672.

Dunbabin, M., Marques, L., 2012. IEEE Robot. Autom. Mag. 19 (1), 24–39.

Frey, B.J., Dueck, D., 2007. Science 315 (5814), 972–976.

Garces, H., Sbarbaro, D., 2011. Eng. Appl. Artif. Intell. 24 (2), 341–349.

Golden, B.L., Levy, L., Vohra, R., 1987. Nav. Res. Logist. (NRL) 34 (3), 307–318.

Gonzalez, R.C., Woods, R.E., 2006. Digital Image Processing (3rd Edition). Prentice-Hall, Inc., Upper Saddle River, NJ, USA.
Gotovos, A., Casati, N., Hitz, G., Krause, A., 2013. Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence. In: IJCAI '13, AAAI Press, pp. 1344–1350.
Hitz, G., Galceran, E., Garneau, M., Pomerleau, F., Siegwart, R., 2017. J. Field Robot. 34 (8), 1427–1449.
Hitz, G., Gotovos, A., Pomerleau, F., Garneau, M.E., Pradalier, C., Krause, A., Siegwart, R., 2014. Robotics and Automation (ICRA), 2014 IEEE International Conference on. pp. 2658–2664.
Hollinger, G.A., Sukhatme, G.S., 2014. Int. J. Robot. Res. 33 (9), 1271–1287.
Kataoka, S., Morito, S., 1988. J. Oper. Res. Soc. Japan 31 (4), 515–531.
Kim, S., Kim, J., 2013. IEEE Trans. Cybern. 43 (5), 1335–1346.
La, H.M., Sheng, W., 2013. IEEE Trans. Cybern 43 (2), 766–778.
La, H.M., Sheng, W., Chen, J., 2015. IEEE Trans. Syst. Man Cybern.: Syst. 45 (1), 1–12.
Laporte, G., Martello, S., 1990. Discrete Appl. Math. 26 (2), 193–207.
de Leon, J.L.D., A., J.H.S., 1998. IEEE Trans. Syst. Man Cybern. B 28 (3), 467–472.
Liu, A., Jun, G., Ghosh, J., 2009. Proceedings of the 2009 SIAM International Conference on Data Mining. pp. 814–825.
McMahon, J., Yetkin, H., Wolek, A., Waters, Z.J., Stilwell, D.J., 2017. 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). pp. 87–94.
Muttil, N., Chau, K.W., 2007. Eng. Appl. Artif. Intell. 20 (6), 735–744.
Pang, S., Farrell, J.A., 2006. IEEE Trans. Syst. Man Cybern. B 36 (5), 1068–1080.
Popovic, M., Hitz, G., Nieto, J.I., Sa, I., Siegwart, R., Galceran, E., 2016. Online informative path planning for active classification using uavs., CoRR abs/1609.08446.
Rahimi, M., Pon, R., Kaiser, W.J., Sukhatme, G.S., Estrin, D., Srivastava, M., 2004. Robotics and Automation, 2004. Proceedings. ICRA '04. 2004 IEEE International Conference on, Vol. 4. pp. 3537–3544.
Rasmussen, C.E., Williams, C.K.I., 2006. Gaussian Processes for Machine Learning. MIT Press, Cambridge, MA, USA.
Scherer, J., Yahyanejad, S., Hayat, S., Yanmaz, E., Andre, T., Khan, A., Vukadinovic, V., Bettstetter, C., Hellwagner, H., Rinner, B., 2015. Proceedings of the First Workshop on Micro Aerial Vehicle Networks, Systems, and Applications for Civilian Use. In: DroNet '15, ACM, New York, NY, USA, pp. 33–38.
Shahidian, S.A.A., Soltanizadeh, H., 2016. Aerosp. Sci. Technol. 58, 189–196.
Singh, A., Krause, A., Guestrin, C., Kaiser, W.J., 2009. J. Artif. Int. Res. 34 (1), 707–755.
Singh, A., Nowak, R., Ramanathan, P., 2006. Proceedings of the 5th International Conference on Information Processing in Sensor Networks. In: IPSN '06, ACM, New York, NY, USA, pp. 60–68.
Thomadsen, T., Stidsen, T., 2003. The quadratic selective travelling salesman problem., IMM-Technical Report-2003-17, Informatics and Mathematical Modelling, Technical University of Denmark, DTU, Richard Petersens Plads, Building 305, DK-2800 Kgs. Lyngby.
Tokekar, P., Hook, J.V., Mulla, D., Isler, V., 2016. IEEE Trans. Robot. 32 (6), 1498–1511.
Vansteenwegen, P., Souffriau, W., Van Oudheusden, D., 2011. European J. Oper. Res. 209 (1), 1–10.
Xu, D., Tian, Y., 2015. Ann. Data Sci. 2 (2), 165–193.
Ycel, Z., Salah, A., Merili, A., Merili, T., Valenti, R., Gevers, T., 2013. IEEE Trans. Cybern. 43 (3), 829–842.