



Savva, F., Anagnostopoulos, C. and Triantafillou, P. (2020) SuRF: Identification of Interesting Data Regions with Surrogate Models. In: 36th IEEE International Conference on Data Engineering (IEEE ICDE), Dallas, TX, USA, 20-24 April 2020, pp. 1321-1332. ISBN 9781728129037 (doi:[10.1109/ICDE48307.2020.00118](https://doi.org/10.1109/ICDE48307.2020.00118))

There may be differences between this version and the published version. You are advised to consult the publisher's version if you wish to cite from it.

<http://eprints.gla.ac.uk/209812/>

Deposited on 11 February 2020

Enlighten – Research publications by members of the University of Glasgow  
<http://eprints.gla.ac.uk>

# SuRF: Identification of Interesting Data Regions with Surrogate Models

Fotis Savva  
University of Glasgow, UK  
f.savva.1@research.gla.ac.uk

Christos Anagnostopoulos  
University of Glasgow, UK  
christos.anagnostopoulos@glasgow.ac.uk

Peter Triantafillou  
University of Warwick, UK  
p.triantafillou@warwick.ac.uk

**Abstract**—Several data mining tasks focus on repeatedly inspecting multidimensional data regions summarized by a statistic. The value of this statistic (e.g., region-population sizes, order moments) is used to classify the region’s interesting-ness. These regions can be naively extracted from the entire dataspace – however, this is extremely time-consuming and compute-resource demanding. This paper studies the reverse problem: analysts provide a cut-off value for a statistic of interest and in turn our proposed framework efficiently identifies multidimensional regions whose statistic exceeds (or is below) the given cut-off value (according to user’s needs). However, as data dimensions and size increase, such task inevitably becomes laborious and costly. To alleviate this cost, our solution, coined SuRF (SURrogate Region Finder), leverages historical region evaluations to train surrogate models that learn to approximate the distribution of the statistic of interest. It then makes use of evolutionary multi-modal optimization to effectively and efficiently identify regions of interest regardless of data size and dimensionality. The accuracy, efficiency, and scalability of our approach are demonstrated with experiments using synthetic and real-world datasets and compared with other methods.

**Index Terms**—Surrogate model estimation, statistical learning, swarm intelligence, evolutionary multimodal optimization.

## I. INTRODUCTION

Consider Exploratory Data Analysis (EDA) whereby analysts engage in repeatedly selecting regions in their data and subsequently summarizing them by extracting statistics [15]. For instance, analyzing spatial data one might filter out all data points except the ones of a specific district and then measure the number of data points within that region to infer the interesting-ness of it. Multiple methods/algorithms/visualizations implicitly adopt this process and are part of an analyst’s toolbox. A problem with this approach is that the task of mining regions of interest is a tedious and laborious process and in the worst case has exponential complexity. The interesting-ness of a region can also be measured by comparing its extracted statistic with a given threshold by the analysts. Regions whose statistics are greater/less than a given threshold are deemed more interesting.

This new analytics task (query) studied here represents a natural and intuitive way for identifying interesting data regions primarily because the associated threshold is often known implicitly/explicitly. For instance, when the task is to identify geographic regions which have an index less/more than a national average or for tasks in which readings beyond a threshold are deemed harmful (eg pollution levels or in

nuclear reactors). The same task is useful, for instance, in cluster analysis [26] when deciding which clusters to prune, in detecting regions of interest in fMRI scans [25] (where only the regions that are ‘activated’ are shown), when trying to identify landmarks [29] based on tracking data, etc. For these reasons, we believe mining regions that exceed a statistic threshold is in many cases more appropriate than having the analyst ask for the *top – k* regions of interest.

### A. Use Case Examples

Let 2-dimensional spatial coordinates describe the locations of Crime Incidents (or any data points with spatial dimensions like traffic congestion and pollution levels in urban areas, etc.). Proactively identifying regions which contain a pre-defined number of data points within them can advise analysts as to which areas are worth looking into. For instance, a region having more crime incidents than a global threshold or having higher average deprivation/crime-index indicator could suggest lack of infrastructure, policing or social/economic disparities compared to other regions. Identifying such regions is non-trivial – we will show that a naive approach has exponential complexity.

Use cases are not restricted to density of data points within regions. Consider, for example, data from activity trackers. Analysts may wish to find time-frames (regions/ranges in time) with high ratio of a specific activity (e.g., sitting, standing, cardio, etc). These constitute crucial information about the activity patterns of a user. If other attributes are also incorporated, (e.g., GPS coordinates, geo-spatial readings from accelerometers, etc.) the analysts can learn when & where an activity occurs most often, along with what type of readings indicate the activity is taking place. Note that these regions of interest demarcate boundaries in multidimensional space – this is of high value to analysts as it makes them easy to interpret.

As a use case in high dimensional data spaces, consider Machine Learning (ML) classification, where analysts are interested in finding regions with a high ratio of certain classes, which implicitly suggest classification boundaries. This task cannot be performed visually unless dimensionality reduction is employed, which does not guarantee fine-grained and accurate results and may suggest regions which are no longer interpretable.

## B. Contributions

We study a new analytics task (query) whereby data regions of interest are identified, given a threshold of a statistic of regions. We contribute a learning & optimization methodology to efficiently and accurately process such *per* analyst queries. We formulate this, as an optimization problem which can be of multimodal nature (as multiple regions matching the analyst request can exist). We identify the back-end data/analytics system as being a bottleneck in examining the validity of the proposed regions. To alleviate this key problem, we propose the use of ML models to learn from past evaluations and approximate the behavior of the back-end system, i.e., to find surrogate models that replace the back-end data system for this task. We then use these models in an evolutionary multimodal optimization for identifying the regions of interest per analyst request. Concretely, the paper provides the following technical contributions:

- We formalize the task of mining interesting regions based on statistics given a cut-off value and provide objective functions for optimization.
- We propose the use of multimodal multiple-swarm optimization algorithm to locate multiple regions of interest
- We adopt statistical learning for approximating the back-end system via past function evaluations. Both ML-driven approximation and evolutionary optimization alleviate the inherent complexity of the considered task.
- Finally, we provide extensive experimental results evaluating and comparing SuRF and the various algorithmic strategies with other methods.

The rest of the paper is organized as follows: Section II formalizes the problem of finding interesting regions and describes a baseline algorithm. Section III defines the optimization problem and introduces evolutionary multimodal optimization for solving it. Section IV describes the type of surrogate model needed to approximate the behavior of the back-end analytics system. Finally Section V contains a comprehensive list of experiments and results that assess the accuracy and efficiency of SuRF.

## II. PROBLEM DEFINITION & RATIONALE

*Definition 1: (Data Vector)* Let  $\mathbf{a} = (a_1, \dots, a_d)^\top \in \mathbb{R}^d$  denote a multivariate random data vector. A dataset  $\mathcal{B}$  is a collection of  $N$  data vectors  $\{\mathbf{a}_k\}_{k=1}^N$ .

*Definition 2: (Statistic Region)* We define a *statistic region* in a  $d$ -dimensional vector space via the  $(2d+1)$ -dimensional information vector  $\mathbf{q} = [\mathbf{x}, \mathbf{l}, y]^\top$ , where  $\mathbf{x} = [x_1, \dots, x_d]^\top \in \mathbb{R}^d$  is the region center point of the hyper-rectangle with side lengths  $\mathbf{l} = [l_1, \dots, l_d]^\top \in \mathbb{R}_+^d$  across the  $d$  dimensions. A statistic region  $\mathbf{q}$  over dataset  $\mathcal{B}$  is associated with the subset  $\mathcal{D} \subseteq \mathcal{B}$  encompassing vectors  $\mathbf{a}$  such that  $\{\mathbf{a} \in \mathcal{D} \mid \bigwedge_{i=1}^d (x_i - l_i \leq a_i \leq x_i + l_i)\}$ . The component  $y = f(\mathbf{x}, \mathbf{l})$  denotes a statistical mapping  $f : \mathbb{R}^d \times \mathbb{R}_+^d \mapsto \mathbb{R}$  over  $\mathcal{D}$  from the hyper-rectangle  $[\mathbf{x}, \mathbf{l}]$  to a statistic of interest  $y \in \mathbb{R}$ , i.e., scalar  $y$  is the *statistic* extracted from the data vectors in  $\mathcal{D}$ . This can be (not limited to) e.g., number of vectors in

$\mathcal{D}$ , i.e.,  $y = f(\mathbf{x}, \mathbf{l}) = |\mathcal{D}|$ , or the average  $\bar{a}_i$  of dimension  $a_i$ , i.e.,  $y = f(\mathbf{x}, \mathbf{l}; i) = \frac{1}{|\mathcal{D}|} \sum_{k=1}^{|\mathcal{D}|} a_{i,k}$ ,  $\mathbf{a}_k \in \mathcal{D}$ . Note that in the case of the average  $\bar{a}_i$  the  $i$ -th dimension is not part of the defined hyper-rectangle and the definition becomes  $\{\mathbf{a} \in \mathcal{D} \mid \bigwedge_{j \neq i} (x_j - l_j \leq a_j \leq x_j + l_j)\}$ .

*Definition 3: (Surrogate Model)* Given a region  $\mathbf{q}$ , the corresponding mapping  $f$  returns a *local* statistic applied to data vectors in  $\mathcal{D}$ .  $f$  depends on the conditional data distribution  $p(\mathbf{a}|\mathbf{x}, \mathbf{l})$  defined by the hyper-rectangle  $[\mathbf{x}, \mathbf{l}]$ . In addition,  $f$  can be considered as an aggregate function that summarizes the data vectors contained in  $\mathcal{D}$ . There is no restriction to the nature of  $f$  as it can be decomposable (COUNT, SUM) or non-decomposable (MEDIAN). The actual evaluation of  $f$  is computationally expensive as the complete data subset  $\mathcal{D}$  has to be identified, given region  $\mathbf{q}$  out of all data points. Therefore, we rest on a surrogate model  $\hat{f}$  to approximate  $f$ , i.e.,  $f \approx \hat{f}$  given any random  $\mathbf{q}$ . The surrogate model  $\hat{f}$  must be inexpensive to evaluate and has to approximate the true function  $f$  with high accuracy. More details of how  $\hat{f}$  is obtained are given at Section IV.

Imagine a data set  $\mathcal{B}$  that holds a number of recorded incidents, as in our example of Crime Incidents in Section I-A. A random data vector  $\mathbf{a}$  represents a recorded incident in 2D spatial coordinates. A statistic region defined by  $\mathbf{q}$  holds a subset  $\mathcal{D}$  of the recorded incidents and is summarized by statistic  $y$ , which for our purpose, represents the number of recorded incidents within the given region. Hence, a surrogate model is a function  $\hat{f}$  that approximates the true underlying function and produces this statistic for different regions. Through our formulation we will refer to this example for clarity.

*Problem 1:* Given a user requested threshold  $y_R \in \mathbb{R}$ , seek the  $k$  unknown regions  $\{\mathbf{q}_k\}$  over the vectorial space of  $\mathcal{B}$  such that their statistics  $\{y_k\}$  are less (or greater) than  $y_R$ . That is, find the  $k$  unknown regions  $\{\mathbf{q}_k\}$  defined by  $[\mathbf{x}_k, \mathbf{l}_k]$ :

$$\{\mathbf{q}_k \in \mathbb{R}^{2d+1} : y_k = f(\mathbf{x}_k, \mathbf{l}_k) < y_R, \forall k\}. \quad (1)$$

For instance, given a threshold  $y_R = 600$  we seek to find an arbitrary number of regions enclosing more/less than 600 incidents. Note: we adopt  $(y_k > y_R)$  in the case where the sought statistics are all greater than  $y_R$ . The regions are defined as hyper-rectangles as it is an established method [5], [12], [16], [28] of portraying which regions are interesting in an interpretable manner.

To avoid the inherent computationally heavy task of evaluating all possible (not trivially countable) sub-regions that satisfy (1), we approximate a solution to Problem 1 using surrogate models  $\{\hat{f}\}$  over a data set  $\mathcal{B}$ . Evidently, this approach introduces approximation of the evaluation of (1) by *replacing*  $f(\mathbf{x}_k, \mathbf{l}_k)$  with  $\hat{f}(\mathbf{x}_k, \mathbf{l}_k)$ . We also, define an objective function that helps us find multiple regions by finding *local*-optima. In the optimization function, we incorporate region *size* defined by  $\mathbf{l}$ , to penalize large regions, as an arbitrarily large region might not be informative enough. For instance, if we seek regions with a number of incidents larger than  $y_R$  with  $y_R < |\mathcal{B}|$ . Then a region covering all data vectors

(all recorded incidents) is the optimal result. By factoring in the region size we allow the analyst to choose to focus their attention on larger/smaller areas. The objective function is defined as:

$$J(\mathbf{x}, \mathbf{l}) = \frac{y_R - f(\mathbf{x}, \mathbf{l})}{\left(\prod_{i=1}^d l_i\right)^c}. \quad (2)$$

Eq. (2) indicates that the result of the objective is inversely proportional to a region's size. A single *global* optimal solution maximizing the objective at Eq. (2) would be an infinitesimal box surrounding a single point with the greatest difference given by  $y_R - f(\mathbf{x}, \mathbf{l})$ . Indeed this would be a valid solution and might be of interest to the analyst. However, as we will later show there could be multiple *local* optimal solutions meeting the constraints (introduced at (3)) and maximizing (2). Hence, we are not interested in finding one global solution to the given objective, but many. This is also motivated by the analysts need as they would be interested in identifying all regions that are above/less than a threshold to compare underlying causes or perform further analysis. To allow the user some flexibility in terms of region sizes, we introduce a tuning scalar parameter  $c$ , which allows the user to restrict solutions to smaller/larger areas. Hence, we seek the region(s):

$$[\mathbf{x}^*, \mathbf{l}^*] = \arg \max_{[\mathbf{x}, \mathbf{l}] \in \mathbb{R}^{2d}} J(\mathbf{x}, \mathbf{l}) \quad \text{s.t. } f(\mathbf{x}, \mathbf{l}) < y_R. \quad (3)$$

In the case  $f(\mathbf{x}, \mathbf{l}) > y_R$ , we maximize  $-J(\mathbf{x}, \mathbf{l})$ . In the remainder we use (3) without loss of generality. To avoid computational burden we take the logarithm of (2) obtaining:

$$\mathcal{J}(\mathbf{x}, \mathbf{l}) = \log(J(\mathbf{x}, \mathbf{l})) = \log(y_R - f(\mathbf{x}, \mathbf{l})) - c\|\boldsymbol{\xi}\|_1, \quad (4)$$

where  $\boldsymbol{\xi} = [\log(l_1), \dots, \log(l_d)]^\top$  and  $\|\boldsymbol{\xi}\|_1 = \sum_{i=1}^d \log(l_i)$  is the  $L_1$  norm of the log-vector of  $\mathbf{l} = [l_1, \dots, l_d]^\top$ . An interesting property arises from (4) as the logarithm is undefined for negative values. Thus, the objective implicitly rejects regions in which  $y_R - f(\mathbf{x}, \mathbf{l}) < 0$  conforming to the constraint of finding regions less than  $y_R$  (and vice versa for  $f(\mathbf{x}, \mathbf{l}) > y_R$ ), as will be shown in our experiments. In (4),  $c > 0$  is the  $L_1$  regularization parameter limiting the size of  $\boldsymbol{\xi}$  (and of  $\mathbf{l}$ ) coefficients and results in finding fine-grained regions (in size), as discussed later.

#### A. Baseline Complexity

Before elaborating on our computationally efficient approximate solution of Problem 1, we first report on a baseline solution. The computational complexity of mining the  $k$  regions in (1) grows exponentially with data dimensionality  $d$  and size  $N$ . It is not trivial to find exact solutions given continuous data domains of (if not all) different dimensions in  $\mathcal{B}$ . Given continuous (real-valued) attributes  $x_i$ , one way of solving Problem 1 is to perform an exhaustive search. Initially, we could discretize the data using a finite number of multidimensional center points to obtain several  $n$  regions  $\{\mathbf{x}_1 \preceq \mathbf{x}_2 \dots \preceq \mathbf{x}_n\}$ ;  $\mathbf{x}_i \in \mathbb{R}^d$  ( $\preceq$  denotes the point-wise inequality between values of the same dimension). This discretization yields an approximate solution, as the optimal

center for a region could lie in-between the proposed centers. In addition, the arbitrary size of the regions adds another level of complexity to the exhaustive search as we have to consider  $n$  regions with varying sizes across dimensions, such that  $\{\mathbf{l}_1 \preceq \mathbf{l}_2, \dots \preceq \mathbf{l}_m\}$ , which again is an approximate size of the optimal region. Thus, to obtain potential regions via exhaustive search yields asymptotic complexity of  $\mathcal{O}((n \times m)^d)$ . We then have to evaluate the result for each of the obtained regions using (4). Since the objective in (4) entails the evaluation of  $f$  over  $\mathcal{D} \subseteq \mathcal{B}$ , the baseline complexity becomes  $\mathcal{O}((n \times m)^d \times N)$ , assuming that  $f$  can be computed in a single pass over  $\mathcal{B}$  in linear time. As dimensions  $d$  and data vectors  $N$  grow, the task becomes prohibitively costly. Hence, we now show how to leverage evolutionary multi-modal optimization algorithms and surrogate models to reduce the complexity for the mining task at hand.

### III. OPTIMIZATION & VIABLE SOLUTIONS

Given that the baseline complexity of solving this task becomes exponential we seek alternatives. Our task is to maximize the objective in (4) in an efficient manner. We first discuss the form of the objective which will help us identify candidate optimization algorithms.

The solution space of the objective in (4) might have a unique (optimum) or multiple solutions (local optima) given an arbitrary  $y_R$ . Based on Problem 1, given a  $y_R$ , the probability of finding a *viable* region is

$$\mathbb{P}\{f(\mathbf{x}, \mathbf{l}) > y_R\} = 1 - F_Y(y_R), \quad (5)$$

where  $F_Y$  is the cumulative distribution function (CDF) of  $y$ . Since,  $\lim_{y_R \rightarrow +\infty} F_Y(y_R) = 1$ , it indicates that the objective function will have less viable solutions because  $\mathbb{P}\{f(\mathbf{x}, \mathbf{l}) > y_R\} \rightarrow 0$ , i.e., the probability of a viable solution diminishes. In the case  $f(\mathbf{x}, \mathbf{l}) < y_R$ , we obtain  $\lim_{y_R \rightarrow -\infty} \mathbb{P}\{f(\mathbf{x}, \mathbf{l}) < y_R\} = \lim_{y_R \rightarrow -\infty} F_Y(y_R) = 0$ . Hence, with an *appropriate*  $y_R$ , i.e., strictly non-zero probability (5), we expect to find multiple regions (local optimal) satisfying (1), i.e.,  $k \geq 1$  regions. It is highly plausible that given an *appropriate*  $y_R$ , multiple regions  $k$  exist satisfying  $f(\mathbf{x}_k, \mathbf{l}_k) > y_R$ . Therefore we make use of a *multimodal optimization* algorithm capable of finding all the possible solutions for Problem 1.

#### A. Multimodal Evolutionary Optimization

Due to the multimodal nature of Problem 1, we cannot adopt optimization methods which return a single optimal solution (=region) given  $y_R$ . Therefore, we cast our optimization problem as an evolutionary multimodal optimization problem [19] adopting methodologies from Swarm Intelligence. We adopt the Glowworm Swarm Optimization (GSO), which is a multimodal variant of the well-known Particle Swarm Optimization (PSO) method [17]. Both GSO and PSO methods are computationally light providing near-optimal solutions (regions in our context) in the face of non-differentiable *fitness* objective functions. Notably, GSO optimizes multimodal fitness functions as it converges towards multiple local-optima, thus considered a good candidate optimizer for our problem.

GSO makes use of *particles*, which are represented as multidimensional candidate solutions in the solution space. The particles move around the solution space and eventually converge to local-optima. A candidate solution particle  $\mathbf{p} = [\mathbf{x}, \mathbf{l}] \in \mathbb{R}^{2d}$  refers to a region defined by  $[\mathbf{x}, \mathbf{l}]$  in the  $(2d)$ -dimensional solution space. The fitness objective function that we use for GSO is the objective  $\mathcal{J}$  in (4), which encapsulates the function  $f$ . However, given an arbitrary  $y_R$ , our method avoids the evaluation of  $f$  over all the possible viable solutions. The fitness function of GSO becomes the objective  $\hat{\mathcal{J}}$  derived from (4) by replacing  $f$  with the estimate  $\hat{f}$ . Hence, the solutions are evaluated using  $\hat{\mathcal{J}}$  given  $\hat{f}$ .

In short, GSO initializes a number of particles  $\{\mathbf{p}_i\}$  at random positions in  $\mathbb{R}^{2d}$ . Each particle  $\mathbf{p}_i$  is associated with a *luciferin* value  $\ell_i$  emulating glowworms. The glowworms have an adaptive neighborhood that helps them identify their neighbours and move towards other particles that have higher luciferin values. Because their movements are based only on local interactions and in small neighbourhoods it allows the swarm of glowworms to partition into disjoint groups and converge to multiple local optima. The GSO algorithm is executed iteratively with discrete steps  $t = \{1, 2, \dots\}$  and is split into two phases. The first phase updates the luciferin  $\ell_i(t)$  at step  $t$  for each particle  $\mathbf{p}_i = [\mathbf{x}_i, \mathbf{l}_i]$  in the swarm using:

$$\ell_i(t) = (1 - \rho)\ell_i(t-1) + \gamma\hat{\mathcal{J}}(\mathbf{x}_i, \mathbf{l}_i) \quad (6)$$

The factor  $\rho$  in (6) is the luciferin decay, which reduces attraction to particles that are not moving towards local-optima. The factor  $\gamma$  in (6) is the luciferin enhancement and increases attraction of particles close to local-optima dictated by the current evaluation of  $\hat{\mathcal{J}}$ . The second phase updates the (position) vector  $\mathbf{p}_i$  of each particle w.r.t to a neighbourhood of particles  $\mathcal{N}_i(t) = \{\mathbf{p}_j : \|\mathbf{p}_i - \mathbf{p}_j\| \leq r_i(t) \wedge \ell_j(t) > \ell_i(t)\}$  in which the selected neighbours have higher luciferin values and are within a current radius  $r_i(t)$  in  $L_2$  (Euclidean) distance. GSO then adapts the (position) vector  $\mathbf{p}_i$  towards a neighbour  $\mathbf{p}_j \in \mathcal{N}_i(t)$  with the maximum selection probability:

$$\mathbb{P}\{\mathbf{p}_j\} = \frac{\ell_j(t) - \ell_i(t)}{\sum_{k \in \mathcal{N}_i(t)} \ell_k(t) - \ell_i(t)} \quad (7)$$

Fig. 1 illustrates the final (converged) positions of the particles over a 2-dim. region space. The x-axis denotes the center of region  $x$  and the y-axis denotes the side length  $l$ . Hence each particle is a region defined over this space, with the intensity of the color at Figure 1 being the value of the objective function (4) across the space. The final positions are illustrated as red “x” and the slightly shaded blue dots are previous positions held by those particles. In this example, 84% of the particles have converged to regions satisfying the constraint set here ( $f(\mathbf{x}, \mathbf{l}) > 1080$ ),  $y_R = 1080$ . As witnessed a large number of particles have converged to the objective’s peaks which suggest better regions. Indeed the regions at the bottom (the peaks) constitute pre-defined *ground-truth* regions (explained in our evaluation section). There are also particles that seem stationary as they are in a space *undefined* by our objective

(4), where ( $f(\mathbf{x}, \mathbf{l}) < 1080$ ). We also explain this in more detail in our Evaluation section.

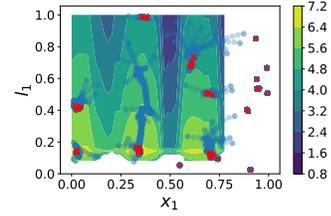


Fig. 1. Final positions of particles (optimal regions) in the 2-dim. region solution space. The objective’s (4) value is the color’s intensity with the peaks shown at the bottom of the plot.

### B. Constraining the Regions Solution Space

We introduce surrogate models in our problem to expedite the process of evaluating viable regions. However, the surrogate models are not restricted within a specific domain. Although the underlying  $f$  is essentially undefined in regions with no data points in  $\mathcal{B}$ , surrogate models are not. The purpose of ML models, is to generalize to unknown regions. Hence, even if the function  $f$  is undefined in areas with no data points,  $\hat{f}$  will still return a result. If the surrogate model is not provided with training examples denoting where the function is undefined then the obtained result might not reflect reality. Therefore, we adapt our algorithm to account for this fact.

The particles in GSO are initially randomly spread across the solution space. Again, the valid solution space (space where data points and thus regions exist) is not reflected and particles only have their neighbours’ luciferin values to guide them. These are inherently associated with the fitness value  $\hat{\mathcal{J}}$ , which then goes back to our initial concern about the validity of the surrogate models. To alleviate this, we first approximate the distribution of the data points  $p_A(\mathbf{a})$  (over a sample for large-scale datasets) in  $\mathcal{B}$  adopting Kernel Density Estimation (KDE) [11] and then we obtain the probability of a region containing any number of data points, i.e., from  $\mathbf{x} - \mathbf{l}$  to  $\mathbf{x} + \mathbf{l}$ . We use this as a guide for particles when selecting which direction to explore. Therefore, given (7), we alternate the selection probability by multiplying with the density (probability) of data points around the particle  $\mathbf{p}_j$ ’s data component  $\mathbf{x}_j$ :

$$\mathbb{P}'\{\mathbf{p}_j\} = \frac{\mathbb{P}\{\mathbf{p}_j\} \cdot \int_{\mathbf{x}_j - \mathbf{l}_j}^{\mathbf{x}_j + \mathbf{l}_j} p_A(\mathbf{a}) d\mathbf{a}}{\sum_{k \in \mathcal{N}_i} \mathbb{P}\{\mathbf{p}_k\} \cdot \int_{\mathbf{x}_k - \mathbf{l}_k}^{\mathbf{x}_k + \mathbf{l}_k} p_A(\mathbf{a}) d\mathbf{a}} \quad (8)$$

### C. Complexity of Multimodal Optimization

As reported earlier, the baseline complexity of our problem is  $\mathcal{O}((n \times m)^d \times N)$ . By adopting GSO and surrogate models  $\hat{f}$ , we expedite this process, obtaining viable solution(s) in  $\mathcal{O}(TL^2d)$ , where  $T$  is the number of iterations and  $L$  is the number of particles for GSO. As a rule of thumb, GSO requires less than  $T \approx 100$  iterations and  $L \approx 100$  glowworms to converge (evidenced also in our experiments shown later). On

the contrary, using the naive approach with just  $n = m = 6$  and  $d = 5$ , one needs to evaluate more than  $6 \cdot 10^7$  possible regions over  $N$  data points. On the other hand, GSO has to execute only  $100 \times 100 = 10^4$  evaluations, just 0.016% of the evaluations needed by the baseline approach.<sup>1</sup> Therefore, by using GSO, the complexity is now of polynomial nature as not all parameter values, spanning uniformly across the entire domain space, have to be examined. In addition, the use of surrogate models has eliminated the need to examine  $N$  data points as the regions no longer have to be evaluated using  $f$ . In the next section, we report on how to approximate  $f$  using ML models. This gives a near-constant time (w.r.t the chosen model) performance for evaluating obtaining the region’s statistic  $y$ .

#### IV. SURROGATE MODEL ESTIMATE

We could approximate  $f$  via various ML models<sup>2</sup> trained to associate a region  $[\mathbf{x}, \mathbf{l}]$  with its corresponding statistic  $y = f(\mathbf{x}, \mathbf{l})$  using a set of past function  $f$  evaluations in  $\mathcal{Q} = \{\mathbf{q}_m = [\mathbf{x}_m, \mathbf{l}_m, y_m]\}_{m=1}^M$ . Using these *training* examples, ML models approximate the actual  $f$ . In general, ML algorithms try to minimize the Expected Prediction Error (EPE)  $\min_{\hat{f}} \mathbb{E}[(f(\mathbf{x}, \mathbf{l}) - \hat{f}(\mathbf{x}, \mathbf{l}))^2]$  which is estimated using an out-of-sample dataset different from  $\mathcal{Q}$ . They also try to find models which are complex enough to minimize this EPE and simple enough to ensure good generalizability to never before seen examples: they tune what is called the Bias-Variance trade-off to ensure the derived model is neither under-fitting nor over-fitting [11]. However, our task is to approximate the behavior of the actual  $f$  applied over regions of data subsets in  $\mathcal{B}$ . Hence our primary concern is *not* to generalize well to new examples. Instead, it is to find a surrogate model  $\hat{f}$ , which follows the *trend* of  $f$  over random regions given an arbitrary  $y_R$ . In other words, our desideratum of  $\hat{f}$  is that given a random region  $[\mathbf{x}, \mathbf{l}]$ , if the statistic  $y = f(\mathbf{x}, \mathbf{l})$  and  $f(\mathbf{x}, \mathbf{l}) < y_R$  then (and only then) the estimate  $\hat{y} = \hat{f}(\mathbf{x}, \mathbf{l})$ , and  $\hat{f}(\mathbf{x}, \mathbf{l}) < y_R$ . That is both  $f$  and  $\hat{f}$  should *agree* on the constraint  $< y_R$  for any random region. This, clearly by definition, does not imply that  $|y - \hat{y}|$  is desired to be as small as possible (i.e., minimizing the prediction error). Instead, we would like to obtain a model  $\hat{f}$  such that whenever  $y < y_R$  holds then,  $\hat{y} < y_R$  holds true, too. Surely, if  $\hat{f}$  minimizes the EPE then we may statistically expect that the two above-mentioned conditions hold true. Nonetheless, both conditions can hold true even if it is not the case that  $y \approx \hat{y}$ . To reflect this objective, we would require to find an estimate  $\hat{f}$ , which minimizes the  $L_2$  norm difference of gradients at any region:

$$\min_{\hat{f}} \mathbb{E}[\|\nabla \hat{f} - \nabla f\|_2] \quad (9)$$

Minimizing the gradient difference we expect that a surrogate model  $\hat{f}$  resembles the behavior of the true underlying function

<sup>1</sup>**Note:** Although the complexity contains  $T \times L^2$ , the number of region evaluations by the algorithm is, in fact,  $T \times L$  [19].

<sup>2</sup>We restrict to a single class of ML models in our experimentation, however this is not necessary and alternative ML models could be employed.

$f$ . However, a number of problems arise if we seek to minimize (9). We have no way of knowing if the true function  $f$  is differentiable and we also do not restrict our choice of ML models to differentiable ones. We could approximate the gradient using a finite number of training samples that are equally spaced in  $(\mathbf{x}, \mathbf{l})$ . But this would mean that we cannot take advantage of past function evaluations, issued by analysts/applications, as an assumption that these examples are equally spaced is invalid.

In this paper, we do not use a specific class of ML models that minimizes (9) and is left as our future work for further investigation. Nevertheless, we adopt conventional ML models minimizing the EPE, which can be directly used for providing robust (in terms of predictability) surrogate estimate model  $\hat{f}$ .

#### V. PERFORMANCE EVALUATION

In our evaluation, we seek to answer the following:

- 1) What is the impact on accuracy, for finding interesting regions per user/application request using ML-approximated surrogate models  $\hat{f}$ ?
- 2) What are the performance benefits of SuRF over the baseline approach and other methods?
- 3) How is the efficiency and accuracy affected by SuRF-GSO, ML-approximate surrogate models and objective functions?

We begin by outlining the implementation details & setup, discussing our methodology and establish evaluation metrics in Section V-A. We showcase the accuracy of SuRF in comparison to other methods using a variety of synthetic datasets in Section V-B. A qualitative analysis over real datasets, showing the applicability of SuRF is presented in Section V-C. The performance benefits of SuRF are discussed in Section V-D. The aforementioned sections provide the answers to questions (1) and (2). Finally, we answer question (3) by evaluating the sensitivity of objective functions, GSO and surrogate ML models in Sections V-F, V-G, and V-H, respectively.

##### A. Implementation Details & Setup

We implemented our algorithms using scikit-learn [24] and adopted the XGBoost (XGB) [8] ML model for our ML-approximated surrogate models  $\hat{f}$ . We implemented Glow-Worm [19] as our optimization algorithm. We performed our experiments using Python 3.5 running on a desktop machine with an Intel(R) Core(TM) i7-6700 CPU @ 3.40GHz and 16GB RAM. The surrogate models used for both synthetic and real datasets were trained using a set of past function evaluations executed across the data space with centers  $\mathbf{x}$  selected uniformly at random and region side lengths  $\mathbf{l}$  set to cover 1% – 15% (uniformly) of the data domain.<sup>3</sup> The surrogate models were hyper-tuned using Grid-Search [24] with  $K$ -fold cross validation. A sensitivity analysis for surrogate models is discussed at Section V-H.

<sup>3</sup>Please note that uniformly sampling regions across the data space with uniform lengths is not the same as obtaining training examples that are *equally spaced* across the complete domain in both  $\mathbf{x}$  and  $\mathbf{l}$

**Methods:** We evaluate the effectiveness and efficiency on mining interesting regions of four different methods: (i) Our framework SuRF which is the ML-approximated surrogate model used with the GSO (ii) *Naive* is the baseline method described in Section II-A<sup>4</sup>, (iii) *f+GlowWorm* is the GSO optimization coupled with the true underlying function which accesses data to evaluate the objective function described in (4), and (iv) PRIM, is an implementation of the algorithm in [12], which is obtained from [1]. PRIM is used to find regions which maximize the result of an output variable. We have found it performs good on our task as well.

**Synthetic Datasets:** We have created 20 synthetic datasets to compare the methods outlined above. The size of the datasets can be arbitrary and it is defined within each experiment. The synthetic datasets have Ground Truth (GT) regions, which are purposely either more dense than the rest of the dataset, or have relatively higher  $y$  values (for the purposes of testing for other statistics). The GT regions are hyper-rectangles constraining a region in all dimensions. Concretely we vary the following settings: number of GT regions  $k = \{1, 3\}$ , statistic type for  $y$  is either: (i) ‘density’ referring to number of data points in subset  $\mathcal{D}$  or (ii) ‘aggregate’ referring to average value of a certain dimension of data points in subset  $\mathcal{D}$ , data dimensions  $d \in \{1, 2, 3, 4, 5\}$ . Each dataset is characterized by a variation of these settings. Note that the statistic could be any other type, e.g., variance, high-order moments. Figure 2 shows four different datasets with varying settings. The sub-figures on the left show data points  $\alpha$  for setting  $d = 1$ , as only the dimension  $a_1$  is to be used to bound the data space. The dimension  $a_1$  has areas with higher values for  $a_i$  and thus the average  $y = \frac{1}{|\mathcal{D}|} \sum_{m=1}^{|\mathcal{D}|} a_{i,m}$  over the highlighted GT regions bounded on  $a_1$  is higher. On the other hand, the sub-figures on the right show the corresponding datasets for the density statistic. The region is bounded by both  $a_1$  and  $a_2$  and for the highlighted (green rectangle) GT area the density of data points is higher. The number of GT regions  $k = 3$  is evident at the bottom sub-figures, in which multiple regions exist for both statistics, and  $k = 1$  at the top sub-figures.

Our goal for each synthetic dataset is to estimate the GT boundaries as close as possible. Let  $\mathcal{R}(\mathbf{x}, \mathbf{l})$  be the hyper-rectangle area corresponding to a random region  $[\mathbf{x}, \mathbf{l}] \in \mathbb{R}^{2d}$  with coordinates:  $[\mathbf{x}-\mathbf{l}, \mathbf{x}+\mathbf{l}]$ . We use a popular metric adopted in data mining, the Intersection over Union (IoU), also known as the *Jaccard Index*. i.e., a ratio where the numerator is the area of overlap between the bounding box (hyper-rectangle)  $\mathcal{R}(\mathbf{x}_k, \mathbf{l}_k)$  of the region  $[\mathbf{x}_k, \mathbf{l}_k]$  mined from any of the outlined methods and the ground-truth bounding box  $\mathcal{G}(\mathbf{x}_0, \mathbf{l}_0)$  corresponding to the GT region  $[\mathbf{x}_0, \mathbf{l}_0]$ . The denominator is the area of union, i.e., the area encompassed by both the  $\mathcal{R}(\mathbf{x}_k, \mathbf{l}_k)$  and the GT bounding box  $\mathcal{G}(\mathbf{x}_0, \mathbf{l}_0)$ , thus, we obtain:

$$\text{IoU} = \frac{\mathcal{R}(\mathbf{x}_k, \mathbf{l}_k) \cap \mathcal{G}(\mathbf{x}_0, \mathbf{l}_0)}{\mathcal{R}(\mathbf{x}_k, \mathbf{l}_k) \cup \mathcal{G}(\mathbf{x}_0, \mathbf{l}_0)}, \quad (10)$$

<sup>4</sup>As the number of function evaluations becomes un-manageable we restrict the discretisation to  $n = m = 6$

where  $\cap$  and  $\cup$  in (10) are adopted as the overlap and union operators over (hyper)-rectangles. One might notice that region dimensionality is not exceedingly high (we experiment up to  $2d = 10$  dimensions in the region solution space for  $d = 5$  data dimensionality). Indeed, at first we conducted experiments by producing synthetic datasets  $\mathcal{U}(0, 1)^d$ ,  $d \gg 5$ , resulting to searching for regions in significantly higher than 10-dimensional spaces. However, due to the effects of curse of dimensionality and as mentioned by Friedman et al. [11], regions (and data points) become increasingly sparse and, thus, the mined regions were returning no data points, thus, no *interesting regions*. The expected length  $l$  of a hyper-cube to retrieve a fraction of data points in unit volume in  $\mathbb{R}^d$  is given by  $\mathbb{E}[r] = r^{\frac{1}{d}}$  [11]. Thus, as dimensionality  $d$  increases, the expected length becomes much larger, covering most of the data domain. Hence, the notion of finding interesting regions becomes meaningless as we would essentially return regions covering most of the data domain. Even though we set the synthetic datasets’ dimensionality up to 5, we highlight the fact that our algorithm deals with  $2d$  dimensions as our regions are expressed as vectors in  $\mathbb{R}^{2d}$  (region solution space).

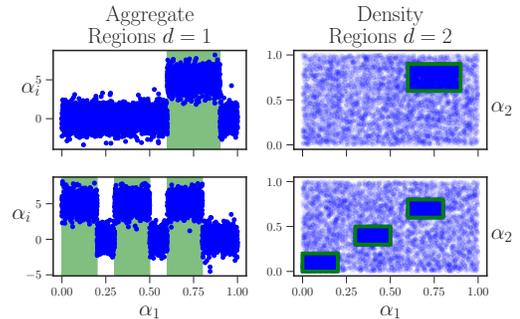


Fig. 2. Synthetic Ground Truth Regions (shaded green) for statistic type ‘aggregate’ and  $d = 1$ (left) and ground truth regions (green rectangles) for statistic type ‘density’ and  $d = 2$  (right), with both a single ground truth region  $k = 1$  (top) and multiple regions  $k = 3$  (bottom).

**Real Datasets:** We use the Crimes [2] and Human Activity datasets [4] publicly available online. As ground-truth regions do not exist for these datasets, we use them to conduct a qualitative analysis experiment, testing the applicability and effectiveness of SuRF to find regions of interest for fixed  $y_R$ . Specifically, we train surrogate models using function evaluations obtained uniformly across the data space with varying lengths and, then, try to find regions of interest given  $y_R$ . Finally, we analyze the obtained regions and confirm that they match to true regions in those datasets. Parameter  $c$  for objective (2) was set to 4.

### B. Accuracy of Interesting Region Identification

All experiments for assessing the accuracy of the interesting region identification were performed on the constraint  $f(\mathbf{x}, \mathbf{l}) > y_R$  with  $y_R$  set to the value close to the extracted statistic given by the GT regions. Specifically  $y_R = 2$  for aggregate statistics and  $y_R = 1000$  for the *density* statistic. As

stated, the surrogate models were trained using past function evaluations, the number of past function evaluations varied as the number of dimensions increases (300 – 300K) to account for the fact that more training examples are required to sufficiently learn a much larger space. The GSO parameters were dynamically adjusted to reach convergence outlined in Section V-G. The objective’s parameter was set to  $c = 4$ . For PRIM, minimum support for the sub-boxes was set to 0.01 and the threshold for aggregate statistics to 2. For Naive as the number of queries becomes prohibitively large we resort to a subset of the total queries that are to be generated. Nevertheless, this is still a good approximation for the method outlined at Section (II-A) and serves as a good baseline. As the synthetic dataset size in this experiment is not important we create synthetic datasets of 7,500 – 12,500 points. Bigger datasets will merely scale the responses. For all algorithms, we obtain the average IoU per dataset by obtaining all the proposed regions given by the algorithms and assessing their IoU with the GT regions.

Figure 3 shows the average IoU over all settings used. As dimensionality increases, the IoU decreases for all methods across all settings. It is worth mentioning that our method is identical to the true underlying function method ( $f$ +GlowWorm) without incurring any of the costs associated with computing the exact results of the statistics. This leads us to believe that the error attributed to the use of an approximation is minimal and, thus, it can be safely used to identify interesting regions with no significant use of computational resources. From all sub-figures, we can deduce that dimensionality plays a crucial role in making this task more challenging. We see a drop in IoU as  $d > 3$ , one contributing factor is that the GT regions cover a much smaller space in higher dimensions. Given a fixed side length of  $l = 0.3$  in uniform space  $\mathcal{U}(0, 1)$ , then the ratio of space covered in  $d = 1$  can be obtained by  $0.3^d = 0.3^1$ . As  $d$  increases then the ratio of space covered becomes much less and thus the possibility of fully intersecting with other hyper-rectangles is relatively small. For instance the ratio of covered space (by the GT) in  $d = 3$  is 2.7% of the total space covered by the unit hyper-cube.

For the aggregate statistic and  $k = 1$  (top-left sub-figure of Figure 3), PRIM outperforms all other methods and is initially invariant by the increase in dimensions. However, for the density statistic (right column in Figure 3), PRIM is unable to spot the GT regions as it is not applicable in such domains. PRIM constructs sub-boxes (hyper-rectangles) by *peeling* across a specific dimension. It sequentially generates smaller sub-boxes  $B$  until the support of current box  $\beta_B$  (i.e.,  $\beta_B = |B|$ ; the number of points belonging in  $B$ ) is below a user-specified threshold  $\beta_0$ . PRIM tries to identify sub-boxes with minimum support  $\beta_0$ , that maximize the average response value of a selected attribute. Formally, PRIM’s objective is:

$$\max_B \mathbb{E}[f(\mathbf{a}) | \mathbf{a} \in B \wedge \beta_B = \beta_0]. \quad (11)$$

The density of a box  $B$  is defined by the support to volume ratio:  $\frac{|B|}{\prod_1^d l_i}$ , where the denominator is the volume of the sub-

box. To this end, there is neither a way to specify *density* as the response variable, nor PRIM takes into consideration the volume of sub-boxes. In addition, PRIM progressively removes sub-boxes such that the expectation in (11) is greater than what it was before the removal of the sub-box. In case where two sub-boxes  $B_i$  and  $B_j$  provide similar gains w.r.t. (11), then the one with *less* support  $\beta_{B_i} < \beta_{B_j}$  is removed. However, in the case of the density statistic and, precisely because PRIM does not consider the region covered by the sub-boxes, a sub-box with higher density might be removed. Of course this should not be considered as a problem of PRIM as we are testing it in settings that was not designed to operate. Its primary use case is to maximize the average response of an attribute by enclosing small sub-boxes in  $d$ -dimensional space.

PRIM also performed less than the rest methods for the aggregate statistic and  $k = 3$  multiple regions (bottom-left in Figure 3)<sup>5</sup>. In general, we are able to get satisfactory IoU with the Naive method, but as we will exhibit in our performance section, its efficiency deteriorates as datasets grow in size and dimension.

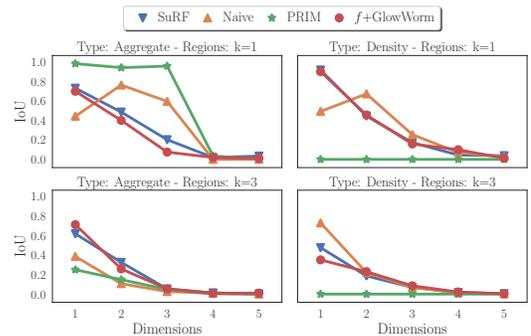


Fig. 3. Average IoU: (Top-Left) for aggregate statistic and  $k = 1$  GT region; (Top-Right) for density statistic and  $k = 1$  GT region; (Bottom-Left) for aggregate statistic and  $k = 3$  GT regions; (Bottom-Right) for density statistic and  $k = 3$  GT regions.

Figure 4 shows the average IoU along with the standard deviation for multiple/single regions (left) and different statistic/aggregate types (right). For multiple regions, Figure 4(left) we note that PRIM has the relatively largest standard deviation and largest decrease in accuracy as we switch from 1 GT regions to 3.

In addition, all other methods seem to be identical, with a decrease experienced from 1 GT region to 3 GT regions. On the other hand, the statistic type (density or aggregate) in Figure 4(right) does not affect accuracy, apart for PRIM’s, which as stated is not able to find regions under this setting. Given our experiments, it is safe to conclude that SuRF is able to detect multiple regions of interest under different types of statistics.

### C. Qualitative Analysis over Real Datasets

We also run a set of experiments over real datasets to exemplify the use cases of SuRF. Using the approach described, we

<sup>5</sup>The IoU for  $k = 3$  is obtained by averaging IoU’s for 3 GT regions.

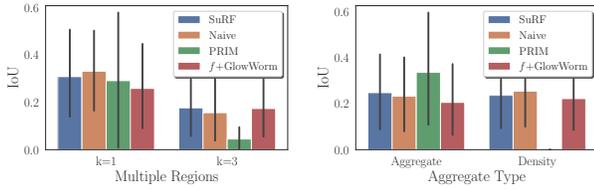


Fig. 4. (Left) Average IoU for multiple regions; (right) Average IoU for different statistics.

examine whether SuRF can indeed identify regions of interest experimenting with `Crimes` [2] and `Human Activity` [4] real datasets. SuRF was trained using synthetically generated past region evaluations. We use SuRF over `Crimes` to identify regions where the crime index is over the 3<sup>rd</sup> quartile of a random set of regions, i.e.,  $y_R = Q_3$  with  $\hat{f}(\mathbf{x}, \mathbf{l}) > y_R$ . Figure 5 shows the number of crimes over X-Y spatial coordinates. The higher the intensity of the color, the higher the crime rate is within the given area. We plot the corresponding density values obtained by the surrogate model  $\hat{f}(\cdot)$ , shown at Figure 5(left), and note that it is a coarse grained approximation to the true density values shown on the right. However, optimizing the objective function using the surrogate model is still sufficient to propose accurate regions in a matter of seconds. The regions shown at Figure 5(left) are the regions that SuRF identified as complying with the constraint  $\hat{f}(\mathbf{x}, \mathbf{l}) > y_R$ . Figure 5(right), shows the same regions over the true density values with 100% of the proposed regions complying with  $f(\mathbf{x}, \mathbf{l}) > y_R$ . This means that the obtained region defined by  $(\mathbf{x}, \mathbf{l})$  complied with the constraint  $> y_R$  at both the surrogate  $\hat{f}$  and the true function  $f$ . Thus, SuRF using approximate surrogate models and GSO is able to pin-point regions of interest in the true data space, complying with the user request  $f(\mathbf{x}, \mathbf{l}) > y_R, y_R = Q_3$ . Moreover, the regions identified are highly parsimonious as the regions denote boundaries in X-Y Coordinates.

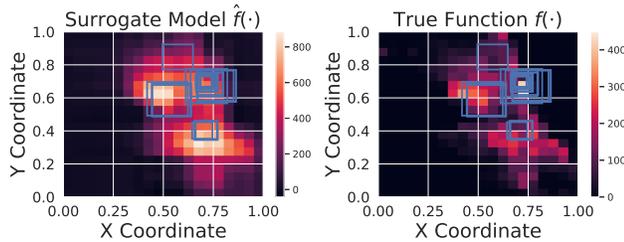


Fig. 5. On the left, identified regions by approximate surrogate function  $\hat{f}$  match to regions identified by the true function  $f$  shown to the right.

Furthermore, the `Human Activity` dataset reports the values for gyrometers and accelerometers. Using the parameters  $(X, Y, Z)$  from the accelerometers, we used SuRF to identify regions with high ratio for a specific activity; for this experiment we used the human activity `stand`. This proactively suggests classification boundaries which the analysts

TABLE I  
COMPARATIVE ASSESSMENT OF DIFFERENT METHODS.

Method	Data size $N$ $d$ dim.	$10^5$	$10^6$	$10^7$
		Time (sec)		
SuRF	1	1.28	1.28	1.3
	2	1.4	1.4	1.4
	3	1.35	1.35	1.35
	4	1.63	1.63	1.64
	5	1.68	1.68	1.69
Naive	1	0.01	0.16	1.94
	2	3.22	33.72	341.7
	3	115.49	1221.6	- (22%)
	4	- (66%)	- (6%)	- (0.5%)
	5	- (1%)	- (0.1%)	- (0.01%)
$f$ +GlowWorm	1	4.71	51.9	601.32
	2	26.7	280.14	2856.02
	3	26.46	289.5	2808.42
	4	27.1	293.62	2981.81
	5	30.21	320.03	-
PRIM	1	0.15	0.4	4.8
	2	0.2	1.9	32.2
	3	0.56	9.3	46.3
	4	0.9	9.5	160.5
	5	1.28	7.36	282.6

can adopt to build a baseline classifier, or further investigate the identified region. SuRF was able to identify regions with ratio of 33% for activity `stand`. Notably, the empirical CDF  $\hat{F}_Y$ , where  $Y$  is the ratio of data points with activity=`stand`, showed that the probability of obtaining  $y_R = 0.3$  was equal to  $\mathbb{P}(f(\mathbf{x}, \mathbf{l}) > y_R) = 1 - \hat{F}_Y(0.3) = 0.0035$ . This denotes a highly unlikely event and also shows that regions with higher ratios are not easy to identify. This denotes the capability of SuRF to mine interesting regions even for cases where the users' requests correspond to highly unlikely regions.

#### D. Models Comparison

We present a comparative assessment with other methods to showcase the efficiency and scalability of SuRF in terms of data size and dimensionality. We also demonstrate the exponential complexity of the considered problem. The performance results are shown in Table I. As shown in Table I, the Naive method is efficient with low dimensional data ( $d = 1$ ). For Naive, we kept  $m = n = 6$ , therefore the number of function evaluations executed were just  $(6 \times 6)^1 = 36$  for  $d = 1$ . However, there is an exponential increase in time as  $d$  increases, and with  $N = 10^7$  data points, Naive times out. The ratio included denotes the number of regions examined before exceeding the time limit, which was set to 3000 seconds. The same trend appears in  $f$ +GlowWorm showing an exponential increase in the amount of time it takes to mine interesting regions. The GSO parameters were set to  $T = 100$  and  $L = 100$  for both  $f$ +GlowWorm and SuRF, with initial swarm neighborhood range  $r_0 = 3$  and constants  $\gamma = 0.6, \rho = 0.4$  as in [19]. For these experiments, we keep GSO's parameters fixed to explore the effects of dimensionality  $d$  and data size  $N$ . At (V-G) we investigate the impact of GSO's parameters on efficiency. PRIM is not affected as much and performs well across all configurations except when

the dimensions  $d$  and data points  $N$  become sufficiently large. On the other hand, SuRF only takes a few seconds across all configurations. Given the same dimensionality  $d$  and a varying dataset size  $N$ , SuRF’s performance remains constant (scales very well) as SuRF does not actually access any data during the mining process. Of course SuRF’s surrogate models are trained before hand for separate statistics. The models will be trained once on a number of past region evaluations and then successively be used for different statistics, thresholds and by different users. Each new request does not need to re-train the model and the overhead for training the surrogate models of SuRF is incurred once. **Note:** It is worth mentioning that all datasets were loaded in memory for performing these experiments. For larger datasets in size  $N$  that do not fit in memory the methods in comparison would have to perform multiple disk accesses, thus, incurring significantly higher costs in solving the discussed mining task. In addition, as stated in [12], PRIM is not equipped to work with disk-access and a common remedy would be to sample the dataset. On the contrary, SuRF models are light enough, to always be loaded in memory and make no use of data at all. For SuRF, it does not matter if the data are stored on a disk or remote data center.

### E. Training Surrogate Models

In this experiment we measure the overhead required to train the surrogate models on a varying number of queries. The results are shown at Figure 6. Using GridSearchCV by [24], we are able to find optimal parameters for our model of choice. For GridSearchCV, we pre-specify a range of parameter values for the parameters of XGBoost. We hypertune the parameters: (i) `learning_rate`  $\in [0.1, 0.01, 0.001]$ , (ii) `max_depth`  $\in [3, 5, 7, 9]$ , (iii) `n_estimators`  $\in [100, 200, 300]$  and, (iv) `reg_lambda`  $\in [1, 0.1, 0.01, 0.001]$ . As expected, this takes more time than only training the models with their values pre-specified, as witnessed at Figure 6. This is because  $3 \times 4 \times 3 \times 4 = 144$  combinations have to be tested on large sets of training examples. We could possibly reduce the number of parameter values, to be tested, to increase efficiency. However, we run the risk of not getting adequate approximations to  $f$ . Surely, this should not be a problem to the analysts as the models will only be trained once. In addition, the models could be trained in a central location on more powerful clusters to expedite this process and then subsequently be used by the analysts.

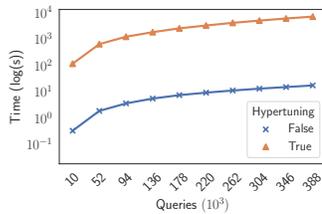


Fig. 6. Training overhead shown in log-scale (y-axis) as the number of queries (x-axis) increase.

### F. Sensitivity on Optimization Functions

We compare the effectiveness of the optimization objectives outlined in (2) and (4) and present the results in Figure 7. The top sub-figures refer to the objective function in (4) and the bottom sub-figures refer to objective function in (2). We used the synthetic data set with  $d = 1$  and  $k = 3$  to be able to visualise the objectives and demonstrate the multimodality in the optimization process. Regarding the objective (4), the use of logarithms explicitly impose that for regions not adhering to the constraint on  $y_R$ , the regions become invalid and the corresponding objective function undefined. Hence, the white area in Figure 7(top) corresponds to those areas. Using this objective, GSO is able to successfully *isolate* glowworms initialized at those areas and eventually adjust their radii to reach glowworms in the valid solution space only. On the other hand, if objective (2) was to be adopted, the glowworms could have formed neighbourhoods in what they would believe are local optima, where in reality, those regions would be invalid. In addition, we conduct an experiment to study the sensitivity of the objective on parameter “ $c$ ”. We keep a fixed number of solutions spread uniformly across the solution space and gradually increase the value of parameter  $c$ . We used the synthetic data set with  $d = 1$  and  $k = 1$ . The results are shown at Figure 8. On the y-axis we plot the number of solutions that are within a given radius (0.2) of the peak (*global optima*). As is evident, the number of viable solutions decreases as  $c$  acts as a regularization variable on the region’s size that is to be accepted.

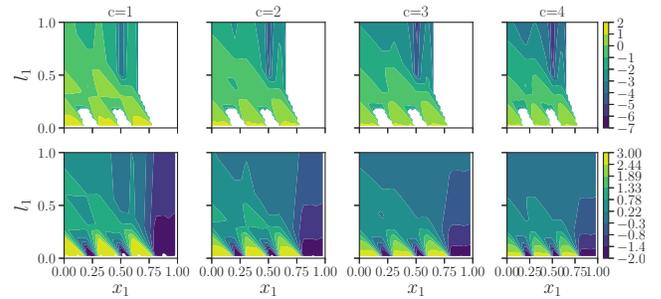


Fig. 7. 2-dim. region solution space examined by (top) objective  $\mathcal{J}$  in (4) and by (bottom) objective  $J$  in (2) as the optimization parameter  $c$  increases.

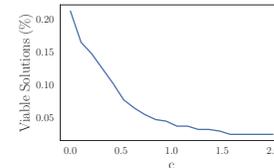


Fig. 8. Sensitivity of parameter  $c$  on identifying viable solutions.

### G. SuRF-GSO Algorithm Sensitivity

We have conducted experiments to evaluate the computational efficiency of GSO and also examined its rate of convergence across different dimensions and parameter settings. Please note that the *Dimensions* parameter has been doubled to reflect the fact that SuRF (and GSO) operate at  $2d$  dimensions per our definition for regions at Def.2. GSO specific parameters such as  $\gamma$ ,  $\rho$  are constant adopted from the respective paper [19]. The results are shown in Figures 9 & 10. Experimentally, we have found that the number of glowworms and neighbourhood radius ( $r_0$  in GSO parameters) have to be adjusted to account for the enlarged region solution space. We increase glowworms using  $L = 50d$  and radius  $r_0 = (1 - \frac{1}{2} \frac{1}{L})^{\frac{1}{d}}$  adopted from [11] Section 2, Equation (2.24). Although the number of needed iterations does vary across settings, as witnessed at Figure 9, the average number of iterations across all settings is 63. Making GSO a robust and efficient algorithm for converging to the various local-optima of the mining task, over different dimensions  $d$  and number of multiple regions  $k$ . Moreover, the average performance for varying number of iterations and glowworms is shown at Figure 10. In Figure 10(left), we increase dimensionality  $d$  and number of glowworms  $L$  as we keep the number of iterations  $T = 100$  constant to measure the impact on performance for a varying number of glowworms. This has minimal effect on the total run-time as it takes no more than 15 seconds for GSO's process to complete (still better than the best competitor shown at V-D). The same holds for the number of iterations in Figure 10(right). Although the average number of iterations required to reach convergence is estimated to be 63 and no setting required more than  $T = 250$  iterations, we measured the performance for up to  $T = 400$  iterations with  $L = 100$ . No more than 10 seconds is required for the largest number of iterations to finish. It appears that both parameters cause an almost linear increase in time for the same number of dimensions even if the stated complexity was  $\mathcal{O}(TL^2d)$ . This is because the number of glowworms is small enough so that the time required is still driven by the prediction time from the approximate  $\hat{f}(\mathbf{x}, \mathbf{l})$  instead of the increase in glowworms.

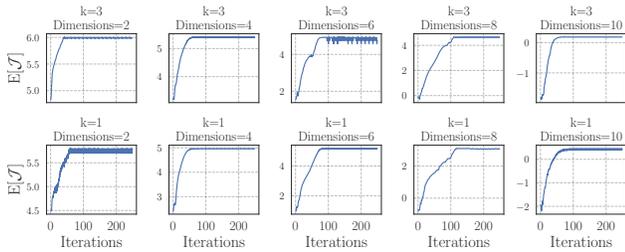


Fig. 9. Expected convergence rates vs iterations  $T$  for different dimensionality  $d$  with  $k \in \{1, 3\}$  multiple regions.

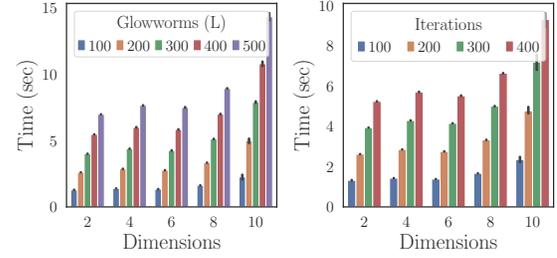


Fig. 10. SuRF-GSO mining performance over dimensionality  $d$  for (left) different number of glowworms  $L$  and (right) iterations  $T$

### H. SuRF-Surrogate Model Sensitivity

In this experiment we evaluate the sensitivity of the surrogate models. Specifically we examine how the number of training samples and out-of-sample generalization error affect the accuracy of the model. In addition, we evaluate how the complexity of the model affects the accuracy of the model and its ability to find obtain good IoU. Figure 11 (left) shows a negative correlation between IoU and Root Mean Squared Error (RMSE) obtained from the ML-trained surrogate models using XGB. For this experiment we use the dataset with a *density* static, dimensions  $d = 3$  and single GT region  $k = 1$ . As the out-of-sample test error (measured by RMSE) increases, the accuracy for IoU drops. This is evidenced by an estimated regression line along with 95% confidence interval and Pearson's Correlation estimated at  $-0.57$ . Therefore, it is important to find ML models that can also act as good statistic estimators. In addition, Figure 11 (right) shows how cross-validated error decreases as the number of training examples for approximating a surrogate function increases. For each ML model at different dimensions, we stop training when no further improvement is measured w.r.t. RMSE. We use datasets with varying dimensions using the *density* statistic and single region  $k = 1$ . The shaded area refer to the error's standard deviation. We note that by  $\sim 1,000$  training examples, i.e., function evaluations, and sufficient hyper-tuning of parameters, the ML models are able to learn the association between region vectors  $[\mathbf{x}, \mathbf{l}]$  and statistic values  $y$  well enough. In our region identification accuracy experiments, we examine the IoU behaviour up to 5-dim. hyper-rectangles corresponding to 10-dim. vectors; recall region is  $[\mathbf{x}, \mathbf{l}]$  with  $\mathbf{x} \in \mathbb{R}^d$  and  $\mathbf{l} \in \mathbb{R}_+^d$ . Hence, the XGB ML models need to learn using  $2 \times d$ -dim. vectors. The number of examples is not at all hard to obtain as in reality multi-dimensional regions are extracted from datasets by a plethora of business intelligence applications. One could also assume that the past function evaluations can be obtained, manually by SuRF, at a regular downtime of the system (where traffic load is low). We also analyze the impact of the XGB-ML model complexity on RMSE and IoU reflected by the maximum depth in regression trees in XGB. The results for both training and cross-validation steps are shown in Figure 12. As expected, RMSE drops as ML model complexity is increased. Although not initially evident,

IoU has a tendency to increase when model complexity increases. However, this might deter the analysts to training a more complicated model as it is evident that they would be able to get a good enough approximation with relatively less complex models.

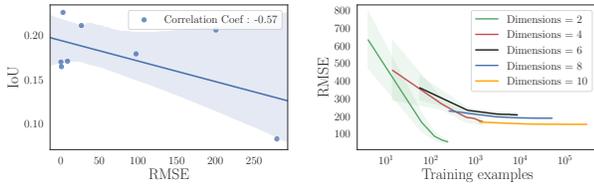


Fig. 11. (Left) Correlation of IoU and RMSE; (right) number of training examples needed to minimize RMSE of XGB ML-approximate surrogate model over different dimensionality  $d$ .

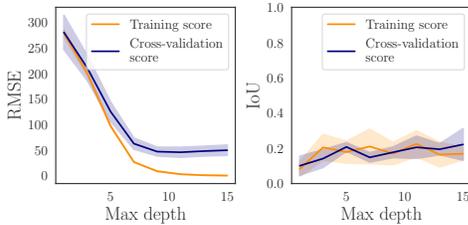


Fig. 12. (Left) RMSE vs ML model complexity (max depth in trees in XGB); (right) IoU vs ML model complexity (max depth in trees in XGB).

## VI. RELATED WORK

Identifying interesting regions can be traced back to Friedman et al. [12] who were interested in finding regions in  $d$ -dimensional spaces that would maximize/minimize a dependent variable  $y$ . Their algorithm processed data sequentially to generate smaller regions and included a pruning step in the end. The computational cost of their algorithm is prohibitive when considering large data sets with respect to dimensionality and number of points. Their objective is different from ours as we do not seek regions that would maximize/minimize  $y$  but regions that satisfy the conditions listed at (3). Our task is also loosely coupled with the objective of Subspace Clustering (SC) [23]. The algorithms proposed for SC aim to identify clusters in low-dimensional sub-spaces by pruning regions and dimensions using some evaluation criteria often being the support of a given region. Other measures of interestingness have also been proposed [28] with the underlying metric still being the number of points. Such methods rely on partitioning schemes, evaluating the density of the found regions and pruning/merging until converging to a region of interest. This is not ideal as the complexity is often exponential w.r.t the number of dimensions [28] as also experimentally evidenced by our Naive/Baseline solution. In addition, although we consider the density of regions as one example use case, in general, we are interested in regions satisfying the constraint outlined in (3) for any given statistic. Thus, our objective is

substantially different, but we regard it as equally important for data mining practitioners. Furthermore, there is large body of work on Subgroup Discovery (SD) [5], [7], [13], of course the list is not exhaustive. The purpose of SD is to find subsets of data that show an interesting behaviour with respect to a given interesting-ness/quality function. It is similar to SC, however SD generalizes the notion of interesting-ness to subsets of data (potentially across all dimensions) of various data types, i.e nominal, binary, numeric etc. Depending on the data type an analogous quality function is employed. Multiple algorithms, both exhaustive [6] and approximate [5] have been developed for this task, however to our knowledge most algorithms are data-driven and do not share our approach to this problem. By data-driven we mean that they employ algorithms that work directly with the underlying data and try to extract subgroups by repeatedly performing region evaluations. We believe that this is costly as data sets become larger.

In other contexts, finding interesting regions was explored for categorical attributes by the construction of OLAP cubes [27] on defined dimensions and hierarchies. As we consider the problem of identifying regions in continuous attributes this approach could not be leveraged as also mentioned in [27], in which they direct to other techniques for continuous data. Alternative formulations, such as posing this problem as finding the top- $k$  regions [22], could also be leveraged and are considered to be complementary to our approach. In the case of, *top-k* formulation, the user has to supply the number of regions, this is often ad-hoc and as evidenced by our examples at Section I a threshold is more intuitive. Hence, each approach can be used in cases when one of the values ( $k$  or threshold) is known. In addition, the complexity of any top- $k$  algorithm inevitably depends on  $N$  (the number of data items),  $d$ , and  $k$ . In intended applications,  $N$  will be very large and (as we argued before, so will be  $k$ ). In contrast, our approach manages to offer performance independent of  $N$ , which is likely to pay off big dividends for big data deployments. Also, note that for the multi-modal case in our experiments, if all top- $k$  regions were to be concentrated in one of those regions (if  $y$  was to be slightly higher for one of the regions) then a top- $k$  approach would effectively identify just one of the regions. Lastly, spatial indexing [14], [22] could also be considered as the indexes produce hyper-rectangular regions over data points which is one of our requirements. However, their goal is not to locate interesting regions with respect to a given statistic but to group data points together for efficient access. Hence, the produced regions only take in mind the spatial distance of data points and produce fixed regions which the user can use. Hence, for any subsequent region that a user wants to identify based on a threshold or a region size requirement the regions produced by these methods are fixed.

Finding regions has been considered in other domains [10], [20], [29], showing an interest for such methods, with different objectives and algorithms which consider smaller data sets  $N < 200,000$ . SuRF is used with an arbitrarily large number of data points  $N$  as effectively makes no use of the underlying database system; instead, SuRF uses ML models to perform

computations over surrogate models.

Machine Learning is increasingly being used to do heavy lifting in data computation where faster and more light-weight models can be leveraged to perform a variety data-intensive tasks. For instance, the authors of [3], [9] trained ML models using past query evaluations to estimate the cardinality of data points returned, given *unseen* queries without performing computations over the data set. In addition other areas include : settings where an estimate of the result of a given aggregate query is requested [21] and for light-weight and efficient indexes over data [18]. Our approach can similarly exploit past issued queries for mining significant statistical information over the underlying data. However, the core objective and context are definitely not the same. We do not wish to train ML models that could answer unseen queries with an arbitrary low error. Instead, we approximate the underlying true function  $f$  using an accurate estimate  $\hat{f}$ , which we then use to inexpensively evaluate (4) and solve (3).

## VII. CONCLUSIONS

We propose SuRF, a solution based on multimodal evolutionary optimization and Machine Learning which efficiently mines regions of interest in multidimensional datasets. Specifically, the regions are associated with a statistic of interest  $y$  computed using the data points included in a region. Thus, the problem of locating regions of interest is formulated as finding regions complying with  $y > y_R$  or  $y < y_R$ , where  $y_R$  is a user defined threshold. Given this constraint an optimization problem is introduced which yields multimodal solution space. SuRF leverages the Glowworm Swarm Optimization built for this class of optimization problems. SuRF also leverages ML models to approximate functions for predicting statistic  $y$  over interesting regions. Therefore, by resorting to these algorithms, SuRF locates regions of interest 150x faster than the best competitor and more than 3 orders of magnitude than the worse, with minimal impact in accuracy. To our knowledge, the problem of finding interesting regions by fusing multimodal optimization with ML has not been investigated before. SuRF is a promising approach in solving a laborious and often manual mining task.

## VIII. ACKNOWLEDGEMENT

This work is partially funded by the European Union EU/H2020 Marie Skłodowska-Curie Action Individual Fellowship (MSCA-IF) under the project INNOVATE (Grant No. 745829). The first author is funded by an EPSRC scholarship, grant number 1804140.

## REFERENCES

- [1] Prim - implementation. URL: <https://github.com/Project-Platypus/PRIM>.
- [2] Crimes - 2001 to present. URL: <https://data.cityofchicago.org/Public-Safety/Crimes-2001-to-present/ijzp-q8t2>, 2018. Accessed: 2018-08-10.
- [3] C. Anagnostopoulos and P. Triantafillou. Query-driven learning for predictive analytics of data subspace cardinality. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 11(4):47, 2017.
- [4] D. Anguita, A. Ghio, L. Oneto, X. Parra, and J. L. Reyes-Ortiz. A public domain dataset for human activity recognition using smartphones. In *Esann*, 2013.
- [5] M. Atzmueller. Subgroup discovery. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 5(1):35–49, 2015.
- [6] M. Atzmueller and F. Puppe. Sd-map—a fast algorithm for exhaustive subgroup discovery. In *European Conference on Principles of Data Mining and Knowledge Discovery*, pages 6–17. Springer, 2006.
- [7] A. Belfodil, A. Belfodil, and M. Kaytoue. Anytime subgroup discovery in numerical domains with guarantees. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 500–516. Springer, 2018.
- [8] T. Chen and C. Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 785–794. ACM, 2016.
- [9] A. Dutt, C. Wang, A. Nazi, S. Kandula, V. Narasayya, and S. Chaudhuri. Selectivity estimation for range predicates using lightweight models. *Proceedings of the VLDB Endowment*, 12(9):1044–1057, 2019.
- [10] J. Eckstein, P. L. Hammer, Y. Liu, M. Nediak, and B. Simeone. The maximum box problem and its application to data analysis. *Computational Optimization and Applications*, 23(3):285–298, 2002.
- [11] J. Friedman, T. Hastie, and R. Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics New York, 2001.
- [12] J. H. Friedman and N. I. Fisher. Bump hunting in high-dimensional data. *Statistics and Computing*, 9(2):123–143, 1999.
- [13] H. Grosskreutz and S. Rüping. On subgroup discovery in numerical domains. *Data mining and knowledge discovery*, 19(2):210–226, 2009.
- [14] A. Guttman and M. Stonebraker. A dynamic index structure for spatial searching. In *Proceedings of the 13th ACM SIGMOD International Conference on Management of Data*, pages 47–57, 1983.
- [15] S. Idreos, O. Papaemmanouil, and S. Chaudhuri. Overview of data exploration techniques. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, pages 277–281. ACM, 2015.
- [16] A. E. Jaffe, P. Murakami, H. Lee, J. T. Leek, M. D. Fallin, A. P. Feinberg, and R. A. Irizarry. Bump hunting to identify differentially methylated regions in epigenetic epidemiology studies. *International journal of epidemiology*, 41(1):200–209, 2012.
- [17] J. Kennedy. Particle swarm optimization. *Encyclopedia of machine learning*, Springer, pages 760–766, 2010.
- [18] T. Kraska, A. Beutel, E. H. Chi, J. Dean, and N. Polyzotis. The case for learned index structures. In *Proceedings of the 2018 ACM SIGMOD International Conference on Management of Data*, pages 489–504. ACM, 2018.
- [19] K. Krishnanand and D. Ghose. Glowworm swarm optimization for simultaneous capture of multiple local optima of multimodal functions. *Swarm intelligence*, 3(2):87–124, 2009.
- [20] B. Liu, L.-P. Ku, and W. Hsu. Discovering interesting holes in data. In *IJCAI (2)*, pages 930–935, 1997.
- [21] Q. Ma and P. Triantafillou. Dbest: Revisiting approximate query processing engines with machine learning models. In *Proceedings of the 2019 International Conference on Management of Data*, pages 1553–1570. ACM, 2019.
- [22] N. Mamoulis, S. Bakiras, and P. Kalnis. Evaluation of top-k olap queries using aggregate r-trees. In *International Symposium on Spatial and Temporal Databases*, pages 236–253. Springer, 2005.
- [23] L. Parsons, E. Haque, and H. Liu. Subspace clustering for high dimensional data: a review. *ACM SIGKDD Explorations Newsletter*, 6(1):90–105, 2004.
- [24] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.
- [25] R. A. Poldrack. Region of interest analysis for fMRI. *Social Cognitive and Affective Neuroscience*, 2(1):67–70, 03 2007.
- [26] Rui Xu and D. Wunsch. Survey of clustering algorithms. *IEEE Transactions on Neural Networks*, 16(3):645–678, May 2005.
- [27] S. Sarawagi, R. Agrawal, and N. Megiddo. Discovery-driven exploration of olap data cubes. In *International Conference on Extending Database Technology*, pages 168–182. Springer, 1998.
- [28] K. Sequeira and M. Zaki. Schism: A new approach for interesting subspace mining. In *Fourth IEEE International Conference on Data Mining (ICDM'04)*, pages 186–193, 2004.
- [29] M. R. Uddin, C. Ravishankar, and V. J. Tsotras. Finding regions of interest from trajectory data. In *2011 IEEE 12th MDM International Conference on Mobile Data Management*, volume 1, pages 39–48, 2011.