Event Detection in Activity Networks

Polina Rozenshtein
Aalto University
Espoo, Finland
polina.rozenshtein@aalto.fi

Aris Anagnostopoulos
Sapienza University of Rome
Italy
aris@dis.uniroma1.it

Aristides Gionis
Aalto University and HIIT
Espoo, Finland
aristides.gionis@aalto.fi

Nikolaj Tatti
Aalto University and HIIT
Espoo, Finland
nikolaj.tatti@aalto.fi

ABSTRACT

With the fast growth of smart devices and social networks, a lot of computing systems collect data that record different types of activities. An important computational challenge is to analyze these data, extract patterns, and understand activity trends. We consider the problem of mining activity networks to identify interesting events, such as a big concert or a demonstration in a city, or a trending keyword in a user community in a social network.

We define an *event* to be a subset of nodes in the network that are close to each other and have high activity levels. We formalize the problem of event detection using two graph-theoretic formulations. The first one captures the compactness of an event using the sum of distances among all pairs of the event nodes. We show that this formulation can be mapped to the MaxCut problem, and thus, it can be solved by applying standard semidefinite programming techniques. The second formulation captures compactness using a minimum-distance tree. This formulation leads to the prize-collecting Steiner-tree problem, which we solve by adapting existing approximation algorithms. For the two problems we introduce, we also propose efficient and effective greedy approaches and we prove performance guarantees for one of them. We experiment with the proposed algorithms on real datasets from a public bicycling system and a geolocation-enabled social network dataset collected from twitter. The results show that our methods are able to detect meaningful events.

Categories and Subject Descriptors

H.3.4 [**Database Management**]: Database Applications—*Data mining*

General Terms

Algorithms, Experimentation

Keywords

Event detection; Submodular function maximization; Prize-collecting Steiner tree; Maximum cut.

1. INTRODUCTION

Detecting events is a fundamental problem in data mining and numerous methods have been applied to a variety of scenarios, including time series and data streams [19], point clouds and vector spaces [12], and networks [11]. The work in this paper concentrates on the latter category: discovering events in networks.

At a high level, our goal is to identify parts of the network with unusual high activity confined in a small space or in a dense part of the graph. We consider a network $G = (V,E)$ whose nodes monitor and record a certain activity. Such a network can represent a sensor network, a social-media site, activity levels from brain imaging, and so on. Given a *time instance* t, each node $v \in V$ in the network keeps a value $w_t(v)$ with the measurement value for the monitored activity. The objective of our approach is to detect an event happening in the network at time instance t. The event is defined with respect to activity values $w_t(\cdot)$ as well as network connectivity. In particular, we aim at finding a subset of network nodes S, such that all nodes in S are close to each other and they all have high levels of activity.

The activity values $w_t(v)$ may be absolute or normalized. Normalization here can be used to capture the abnormality level of a node at a certain time with respect to the routine operation of the network. In general, arbitrarily complex models can be used to define appropriately normalized activity values $w_t(v)$. For instance, periodicity phenomena can be taken into account; for example, when monitoring the traffic activity in a city, the reference values for weekend 8AM traffic are much lower than that of a week day. In this paper, we assume that the activity values $w_t(v)$ are provided as input to our problem. Devising models to obtain finely tuned and appropriately normalized activity values $w_t(v)$ depends on the application at hand and we consider it as an orthogonal problem, outside the scope of this paper.

The problems we define can be applied to a variety of scenarios, such as medical diagnosis, performance monitoring, image or video surveillance, and fraud detection. In this paper we experiment with two applications, event detection in *sensor networks* and *social networks.*

Sensor networks. Consider a network of sensors deployed in a certain region and recording a measurement of interest, such as traffic, pollution, or water-quality level. In some cases, the network nodes are geolocated so that a distance measure is defined among all nodes, whereas in other cases only a list of direct neighbors is known for each node. Events are defined with respect to the activity value monitored: we are interested in finding compact subareas where the traffic measurements are abnormal, the pollution levels are unusually high, and so on.

Social networks. Social networks model social interactions between individuals. The values recorded at each node correspond to a type of activity we are interested in monitoring, for example, the number of messages posted by an individual during the last few hours, an average sentiment score of those messages, or the frequency of keywords associated with a topic of interest. In this application, the events discovered by our methods will correspond to dense network subgraphs that exhibit high values with respect to the activity monitored.

Motivated by the previous discussion, we formalize event detection as a problem of finding a subgraph $S \subseteq V$ in a graph $G = (V,E,w)$ with node weights $w(v)$, for each $v \in V$. The subgraph S needs to be compact in terms of graph distances, and the sum of weights of the nodes of S needs to be large. We express subgraph compactness using two different definitions: *sum of pairs of distances* and *Steiner-tree* cost. The first definition leads to a graph-cut problem. We show that the optimization function is submodular and we provide two simple greedy algorithms with provable approximation guarantees. We also show how we can transform the problem to a variant of the MaxCut problem to obtain an algorithm that is less practical but has better approximation guarantee than the greedy algorithms.

The second subgraph-compactness definition, which is based on the cost of the minimum Steiner tree, leads to a prize-collecting Steiner-tree problem. For that problem we apply a well known 2-approximation algorithm, which is based on the primal–dual paradigm. We also experiment with a greedy heuristic, which in practice gives almost as good solutions as the primal–dual algorithm.

We evaluate our problem formulation and the proposed algorithms on three sensor-network datasets and two social-network datasets. The former are real-world datasets from public bicycling systems from three large cities, Barcelona, Washington D.C., and Minneapolis, while the latter are datasets collected from twitter on two cities, New York City and Los Angeles. Using our methods allows to successfully discover real events in the cities, as reflected on the usage of shared bikes or twitter volume.

1.1 Related work

Statistical methods. A statistical approach for finding anomalies is the *spatial scan statistic* [23]. The method typically assumes that the data are distributed according to some distribution on a Euclidean space. The goal is then to detect whether there exists a subarea where the data are distributed according to the same distribution but with a higher density parameter. This approach is related to our setting but it has important differences. First, the statistical approach defines a null hypothesis, therefore, it assumes an underlying distribution over the data. Instead, our approach is formalized through an optimization function, and no such assumptions are needed. Second, the classical approaches are rather heuristic in nature: the methods detecting the most diverse subareas are usually based on Monte Carlo sampling, even though there have been approaches to formulate them as optimization problems and provide algorithmic solutions [1, 2]. Yet a third difference is that usually these approaches assume that the shape is predefined (e.g., a circle or an axis-parallel rectangle), which allows for the design of algorithms that can search the space of shapes. Statisticians have proposed generalizations where the shape of the dense cluster is not fixed a priori [7, 25, 27], however this forces the solution of the detection problem to be heuristic (e.g., Monte Carlo simulations). Finally, all these approaches assume that there exists an underlying Euclidean geometry on the space. For several of the applications we are interested, such as social networks, such Euclidean geometry is not present, thus requiring alternative approaches.

In summary, the approach of this paper is conceptually different from all the related work on spatial scan statistics. Here we provide a formal graph-theoretic definition of the problem and algorithms that have theoretical quality guarantees and are efficient in practice.

Event detection in social media. Event detection based on geospatial information from social media and twitter is a research area that has attracted significant attention in the last years. Baldwin et al. [8] developed an interactive system, in which a user can insert some queries and obtain the volume of tweets containing these terms at different granularities of space and time. Walther and Kaisser [30] developed a system for detecting events that take place from the twitter stream. It gathers tweets as they are created and it clusters them online based on geolocation. A machine-learning module evaluates whether a cluster of tweets refer to an event. Also Watanabe et al. [31] develop a similar system that identifies tweets that are created close in time and space and by looking at co-occurring terms it attempts to discover if they refer to the same event. Olson et al. [24] designed a system that aims at detecting very rapidly large *rare* events: events that happen with very low frequency but with large consequences, such as an earthquake or a tsunami. The difference of this line of work with our approach is that we offer a graph-theoretic formulation to the problem, which can be applied to any graph, not just geography-induced graphs. Additionally, our method does not uses text, but it assumes numerical measurements on the graph nodes.

Anomaly and outlier detection. Our problem has also resemblance with problems related to outlier and anomaly detection in networks. The main objective of these works is to identify patterns that are different than normal. For instance, Heard et al. [20] apply statistical techniques to discover a subset of the nodes that in a small period of time change significantly their communication patterns. Bhuyan et al. [9] apply clustering on network data for detecting intrusion attacks. The tutorial of Akoglu and Faloutsos [3] has an extensive reference list with related publications.

Dense subgraphs. Our problem is related to finding dense subgraphs [6, 14, 16, 22, 29]. Here, the goal is to find small parts of the network with a high number of edges. We are interested in small subgraphs with high number of nodes, thus obtaining different objective functions.

Finally, in a work related to the Steiner-tree problem formulation presented in this paper, Seufert et al. [26]

consider the problem of finding a subtree with k nodes such that the total node weight is maximized. Their approach also relies on the prize-collecting Steiner-tree, but their methods focus on identifying a heavy tree of exactly k nodes.

2. PROBLEM FORMULATION

We consider a graph $G = (V,E,w,c)$, with V being a set of n vertices and E being a set of m edges. The weight function $w : V \to \mathbb{R}$ assigns a nonnegative value $w(v)$ to each vertex v, whereas the distance function $c : E \to \mathbb{R}$ assigns a distance value $c(u,v)$ to each edge (u,v). The edge distance function c can be used to define a new distance function over all pairs of vertices $(u,v) \in V \times V$. This can be done by considering the *shortest-path distance closure* between vertices. Namely, for each $u,v \in V$ we define $d(u,v)$ to be equal to the distance of a shortest path from u to v using edges of G, or ∞ if no such path exists. Unless specified otherwise, in the rest of the paper we assume that the shortest-path distance d is used, and that a finite distance value is defined for all pairs of vertices in the graph.

Our goal is to find a subset of vertices $S \subseteq V$ that has *large total weight* according to the weight function w, and it is sufficiently *compact*, that is, the vertices in S are close to each other according to the distance function d.

To capture our objective we need to define appropriate weight and distance functions for subsets of vertices. Given $S \subseteq V$ we denote such weight and distance functions by $W(S)$ and $D(S)$, respectively. As a set weight function we consider simply the sum of all the weights in the set, that is,

$$W(S) = \sum_{v \in S} w(v).$$

For measuring the total distance of a set S we consider two options. The first option is to sum the distances of all pairs of vertices in S, namely

$$D_{\mathrm{AP}}(S) = \frac{1}{2} \sum_{v \in S} \sum_{u \in S} d(u,v).$$

This type of objective function is suitable for events that are concentrated in a small area in a round-shaped area, for instance a football game. But events might have different shapes, such as a parade, a street concert, a firework show, and so on; for such events, we need a distance function that does not penalize long distances, as long as points in between are also active. This leads to using the *minimum Steiner tree* of the subgraph induced by the set of vertices S. We denote this total-distance measure by $D_{\mathrm{T}}(S)$. Thus, we have

$$D_{\mathrm{T}}(S) = \min_{T \in \mathcal{T}(G[S])} \sum_{(u,v) \in T} d(u,v),$$

where $G[S]$ denotes the subgraph of G induced by a subset of vertices $S \subseteq V$, and $\mathcal{T}(H)$ denotes the set of all the trees of a graph H.

Subsets of vertices $S \subseteq V$ that correspond to meaningful events need to have *large* weight value $W(S)$ and *small* total distance value $D(S)$. To combine the two measures in a way that allows to maximize simultaneously $W(S)$ and minimize $D(S)$ there are many different options, for example, optimizing the one measure while setting a budget constraint on the other or taking a linear combination of the two measures. The former approach induces difficulties in devising approximation algorithms. To avoid them, we consider a linear combination of the two measures into a single objective with a normalization coefficient λ. The coefficient λ provides an easy way to control the relative importance of the two measures. In addition, as we will see in the next section, such a linear combination leads to neat mathematical forms, which can be viewed as a quadratic integer program or as well studied tree-based problems.

Thus, we consider the following problem formulation.

PROBLEM 1 (EVENT). *Given a graph $G = (V,E,w,c)$ with vertex weights w and edge distance c, and a normalization coefficient λ, find a subset of vertices $S \subseteq V$ that maximizes the objective function*

$$Q(S) = \lambda W(S) - D(S). \tag{1}$$

Problem 1 is a generic problem. The exact structure of the problem and solution methods depend on which distance function D is used. We obtain two instantiations of Problem 1 by considering the two set distance functions D_{AP} and D_{T} that we discussed previously. Thus, we consider the following two specific problems.

PROBLEM 2 (EVENTALLPAIRS). *Given a graph $G = (V, E,w,c)$ with vertex weights w and edge distance c, and a normalization coefficient λ, find a subset of vertices $S \subseteq V$ that maximizes the objective function*

$$Q_{\mathrm{AP}}(S) = \lambda W(S) - D_{\mathrm{AP}}(S). \tag{2}$$

PROBLEM 3 (EVENTTREE). *Given a graph $G = (V,E, w,c)$ with vertex weights w and edge distance c, and a normalization coefficient λ, find a subset of vertices $S \subseteq V$ that maximizes the objective function*

$$Q_{\mathrm{T}}(S) = \lambda W(S) - D_{\mathrm{T}}(S). \tag{3}$$

One should note that in the above problem formulations, the objective functions Q, Q_{AP}, and Q_{T} can take negative values. Negative values do not create any problems as long as one is after exact solutions. However, in the context of approximation algorithms, objective functions with negative values are problematic because it becomes more difficult to apply the concept of multiplicative approximation guarantee. To overcome this problem we modify the objective functions to ensure that they take nonnegative values. We do so by adding a constant term, and we thus consider a *shifted* version of our objective functions.

First, for the problem EVENTALLPAIRS we consider the shifted function $Q^{+}_{\mathrm{AP}}(S) = Q_{\mathrm{AP}}(S) + D_{\mathrm{AP}}(V)$. It is easy to see that the function Q^{+}_{AP} is nonnegative for all $S \subseteq V$. This makes it easier to design approximation algorithms with multiplicative approximation guarantees. A modified version of the EVENTALLPAIRS problem, which we denote by EVENTALLPAIRS+, is now defined as follows.

PROBLEM 4 (EVENTALLPAIRS+). *Given a graph $G = (V,E,w,c)$ with vertex weights w and edge distance c, and a normalization coefficient λ, find a subset of vertices $S \subseteq V$ that maximizes the objective function*

$$\begin{aligned} Q^{+}_{\mathrm{AP}}(S) &= Q_{\mathrm{AP}}(S) + D_{\mathrm{AP}}(V) \\ &= \lambda W(S) - D_{\mathrm{AP}}(S) + D_{\mathrm{AP}}(V). \end{aligned} \tag{4}$$

For the EVENTTREE problem we follow a different approach: first we note that with respect to finding an exact solution, maximizing the function Q_{T} is equivalent to minimizing $-Q_{\mathrm{T}}$. We choose to work with this minimization problem, and we consider minimizing the shifted function $Q^+_{\mathrm{T}}(S) = -Q_{\mathrm{T}}(S) + \lambda W(V)$, which is a nonnegative for all $S \subseteq V$. For this shifted objected function it holds:

$$\begin{aligned} Q^+_{\mathrm{T}}(S) &= \lambda W(V) - \lambda W(S) + D_{\mathrm{T}}(S) \\ &= \lambda W(V \setminus S) + D_{\mathrm{T}}(S). \end{aligned}$$

The interpretation of the above objective function is to find a set S so that the tree cost $D_{\mathrm{T}}(S)$ and the (scaled by λ) weight of the vertices *not included* in S is minimized. This problem is known as the *prize-collecting Steiner-tree* (PCST) problem [4,21]. The term "prize collecting" comes from thinking of the weights on the vertices of the graph as *prizes* and the goal is to find a tree that minimizes the tree cost and the total value of prizes not spanned by it. The shifted version of the EventTree problem, denoted by EventTree+, is now defined as follows.

Problem 5 (EventTree+). *Given a graph $G = (V, E, w, c)$ with vertex weights w and edge distance c, and a normalization coefficient λ, find a subset of vertices $S \subseteq V$ that minimizes the objective function*

$$Q^+_{\mathrm{T}}(S) = \lambda W(V \setminus S) + D_{\mathrm{T}}(S). \qquad (5)$$

For general graphs, the problem EventAllPairs+ is **NP**-hard. The proof of the next lemma is obtained by a reduction from the IndependentSet problem and is omitted because of lack of space.

Lemma 1. *The problem* EventAllPairs+ *is* **NP**-*hard for graphs with general edge distance functions.*

In the more restricted version in which the input graph is a metric the complexity of the problem is open.

On the other hand, the problem EventTree+ is **NP**-hard even for metric edge distances as it generalizes the Steiner-tree problem:

Lemma 2. *The problem* EventTree+ *is* **NP**-*hard.*

3. ALGORITHMS FOR THE EventAllPairs+ PROBLEM

In this section we present our algorihtms for the EventAllPairs+ problem, starting with efficient greedy approaches and continuing with a slower but more effective approach based on the MaxCut problem.

3.1 Greedy algorithms

We start our discussion on the EventAllPairs+ problem by considering the properties of the underlying objective function Q^+_{AP}. In particular, we can show that the function Q^+_{AP} is submodular. Submodularity is a desirable property because a number of approximation algorithms rely on it. The approximability of a submodular function depends on other properties as well, in particular whether the function is monotone and/or symmetric. We recall that if V is a ground set, a set function $f : 2^V \to \mathbb{R}$ is submodular if for all $S,T \subseteq V$ we have that $f(S) + f(T) \geq f(S \cap T) + f(S \cup T)$. The function f is monotone if for all $S \subseteq T \subseteq V$ it holds that $f(S) \leq f(T)$. It is symmetric if for all $S \subseteq V$ we have that $f(S) = f(V \setminus S)$.

Lemma 3. *The function Q^+_{AP} is submodular.*

Algorithm 1: BFNS: Greedy algorithm for EventAllPairs+ using the approach of Buchbinder et al. [13]

```
$X_0 \leftarrow \emptyset$, $Y_0 \leftarrow V$
for $i = 1$ to $n$ do
    $a_i \leftarrow Q^+_{\mathrm{AP}}(X_{i-1} \cup \{u_i\}) - Q^+_{\mathrm{AP}}(X_{i-1})$
    $b_i \leftarrow Q^+_{\mathrm{AP}}(Y_{i-1} \setminus \{u_i\}) - Q^+_{\mathrm{AP}}(Y_{i-1})$
    $a'_i \leftarrow \max\{a_i,0\}$, $b'_i \leftarrow \max\{b_i,0\}$
    with probability $a'_i/(a'_i + b'_i)$ do
        $X_i \leftarrow X_{i-1} \cup \{u_i\}$, $Y_i \leftarrow Y_{i-1}$
    else with compliment probability $b'_i/(a'_i + b'_i)$ do
        $Y_i \leftarrow Y_{i-1} \setminus \{u_i\}$, $X_i \leftarrow X_{i-1}$
return $X_n$ (or equally $Y_n$)
```

Proof. (Sketch) We set $I(S) \triangleq \lambda W(S)$ and $D(S) \triangleq D_{\mathrm{AP}}(V) - D_{\mathrm{AP}}(S)$. It is $Q^+_{\mathrm{AP}}(S) = I(S) + D(S)$. It is easy to see that both functions I and D are positive and submodular. The lemma follows from the fact that the sum of two positive submodular functions is submodular. □

Thus we want to approximate a submodular function without any constraints. A recent paper by Buchbinder et al. [13] provides a linear-time $\frac{1}{2}$-approximation algorithm for this problem, which we explain below.

The algorithm of Buchbinder et al. is based on a *randomized double-greedy* approach. The technique utilizes the fact that for a submodular function f, the function $\bar{f}(S) = f(V \setminus S)$ is also submodular. Furthermore, if a set S^* is optimal for f then $V \setminus S^*$ is also optimal for $\bar{f}$ and the optimal values of the two functions are equal.

The suggested approach is a randomized algorithm, which performs two types of greedy steps, searching for the optimal solution for f and $\bar{f}$. The search strategy can be viewed as running two greedy algorithms simultaneously. We start with an arbitrary order of the elements in V, say $u_1, \ldots, u_n$. The two greedy processes traverse the sequences of sets $\{X_0, X_1, \ldots\}$ and $\{Y_0, Y_1, \ldots\}$, respectively. The one greedy process starts from the empty set $X_0 = \emptyset$ and grows it to optimize f, while the other greedy process starts from the ground set $Y_0 = V$ and shrinks it to optimize $\bar{f}$. The algorithm guarantees that the growing set X_i is always included into the shrinking set Y_i. Which of the two steps (grow X_i or reduce Y_i) is a random choice, which depends on the marginal improvement obtained from each move. The algorithm stops when the sets are equal. A formal description of the algorithm, which we call BFNS, is shown in Algorithm 1. We have the following.

Theorem 1 (Buchbinder et al. [13]). *The* BFNS *algorithm provides a $\frac{1}{2}$-factor approximation for the* EventAllPairs+ *problem.*

For the EventAllPairs+ problem the BFNS algorithm has the following interpretation: It examines each graph vertex v one by one and decides whether to keep v in the solution or remove it. The decision is randomized and the probabilities depend on the marginal gains incurred in the Q^+_{AP} function with respect to the current lower and upper set solutions, X_i and Y_i.

Our objective has additional structure, which leads to a trivial $\frac{1}{2}$-approximation algorithm. Recall the definitions of the functions I and D in the proof of Lemma 3. Both functions are positive and $I(S)$ is increasing in S, whereas

$D(S)$ is decreasing in S. Therefore, for the optimal value S^* we have that

$$\begin{aligned} Q^+_{\mathrm{AP}}(S^*) &= I(S^*) + D(S^*) \\ &\leq (I(V) + D(V)) + (I(\emptyset) + D(\emptyset)) \\ &= Q^+_{\mathrm{AP}}(V) + Q^+_{\mathrm{AP}}(\emptyset) \\ &\leq 2\max\{Q^+_{\mathrm{AP}}(\emptyset), Q^+_{\mathrm{AP}}(V)\}. \end{aligned}$$

This means that simply taking the best of the empty set or the entire node set V also provides a $\frac{1}{2}$ approximation. We call this algorithm Trivial.

We finally propose the standard greedy algorithm, which starts from the empty set, adds one vertex at a time, and stops when the solution cannot be improved. We refer to this simple greedy as GreedyAP.

For general unconstrained submodular functions, the greedy algorithm does not provide any guarantee. Here we are able to exploit the specific structure of the objective, and we prove that also GreedyAP provides a $\frac{1}{2}$ approximation. The result follows from the following lemma.

LEMMA 4. *Consider a submodular function F and let S be the solution given by the greedy algorithm optimizing F. Then, $F(S) \geq F(V)$.*

PROOF. We prove a more general statement. We prove that any hill-climbing algorithm, that is, any algorithm that starts with the empty set and increasingly adds elements for as long as the marginal gain is positive, if it returns solution S we have that $F(S) \geq F(V)$.

Assume that $V \setminus S = \{x_1, x_2, \ldots, x_r\}$. Then we have

$$\begin{aligned} F(S) - F(V) &= \sum_{i=1}^{r} (F(S \cup \{x_1, \ldots, x_{i-1}) - F(S \cup \{x_1, \ldots, x_i)) \\ &\geq \sum_{i=1}^{r} (F(S) - F(S \cup \{x_i\})) \geq 0, \end{aligned}$$

where the first inequality follows from the submodularity of F and the second from the fact that the algorithm returned S without adding any element x_i, so each term in the sum is positive. □

COROLLARY 1. *The* GreedyAP *algorithm provides a $\frac{1}{2}$-factor approximation for the* EVENTALLPAIRS+ *problem. This bound is tight.*

PROOF. By the discussion just before Lemma 4, it suffices to show that $F(S) \geq F(\emptyset)$ and $F(S) \geq F(V)$. The former is trivially true, otherwise the algorithm would have returned $S = \emptyset$ and the latter follows from Lemma 4. The counterexample that shows that the greedy cannot, in the worst case, achieve approximation better than 1/2, will appear in the full version of this work. □

Although all three algorithms, BFNS, Trivial, and GreedyAP, have the same theoretical performance, our experiments show that GreedyAP produces solutions of higher quality, even slightly better or equal to MAXCUT-based algorithm, which is theoretically superior and which we discuss next.

3.2 MaxCut formulation

In this section we reduce the EVENTALLPAIRS+ problem to a variant of the MAXCUT problem. This allows us to use a well known algorithm by Goemans and Williamson [18], which has a 0.868-approximation guarantee. For the reduction we need the following variant of MAXCUT.

PROBLEM 6 ((s,t)-MAXCUT). *Given a graph G and two vertices s and t, partition the vertices of G into two sets A and B such that $s \in A$ and $t \in B$ and the total weight of cross edges is maximized. We denote the cost of such a solution A, B by $Q(A,B)$.*

The only difference between (s,t)-MAXCUT and the traditional MAXCUT is that there are two vertices s and t, for which it is forbidden to be in the same cut. This technicality does not have any complexity consequences.

To reduce EVENTALLPAIRS+ to the (s,t)-MAXCUT problem, assume that we are given a graph $G = (V,E)$ equipped with the distance function c. Assume also that we are given a parameter λ. We then construct a new graph H by adding two special vertices s and t into G. We connect s to each $v \in V$ with a weight $c(s,v) = \sum_{u \in V} d(v,u)$. We connect t to each $v \in V$ with a weight $c(t,v) = 2\lambda w(v)$.

Consider an A, B cut of H such that $s \in A$ and $t \in B$. Let $S = A \setminus \{s\}$ and let $T = B \setminus \{t\}$. We argue that the cost of the cut is twice the cost of EVENTALLPAIRS+, that is, $Q(A,B) = 2\,Q^+_{\mathrm{AP}}(S)$. To see this, notice that each vertex $v \in S$ will contribute $2\,\lambda\, w(v)$ to the cost, and each vertex $v \in T$ will contribute $c(s,v) = \sum_{u \in V} d(v,u)$ to the cost. Additional costs will come from edges (u,v), where $v \in S$ and $u \in T$. Combining these costs gives us

$$\begin{aligned} Q(A,B) &= \sum_{v \in S} 2\,\lambda\, w(v) + \sum_{v \in T}\sum_{u \in V} d(v,u) + \sum_{v \in T}\sum_{u \in S} d(v,u) \\ &= 2\,\lambda\, W(S) + \sum_{v \in T}\sum_{u \in T} d(v,u) \\ &\quad + \sum_{v \in T}\sum_{u \in S} d(v,u) + \sum_{v \in T}\sum_{u \in S} d(v,u) \\ &= 2\,\lambda\, W(S) + \sum_{v \in V}\sum_{u \in V} d(v,u) - \sum_{v \in S}\sum_{u \in S} d(v,u) \\ &= 2\,\lambda\, W(S) + 2\,D_{\mathrm{AP}}(V) - 2\,D_{\mathrm{AP}}(S) \\ &= 2\,Q^+_{\mathrm{AP}}(S). \end{aligned}$$

Thus, solving (s,t)-MAXCUT also solves EVENTALLPAIRS+. Moreover, since the costs of both problems are the same, up to a scaling factor, any approximation guarantee for (s,t)-MAXCUT yields the same approximation guarantee for EVENTALLPAIRS+.

Our final step is to solve (s,t)-MAXCUT. Following the seminal algorithm of Goemans and Williamson [18], the problem can be formulated as the following integer program:

$$\begin{aligned} \max \quad & \sum_{u,v \in V(H)} c(u,v) \frac{1 - x_u x_v}{2} \\ \text{such that} \quad & x_u \in \{-1, +1\}, \text{ for all } u \in V(H), \\ & x_s x_t = -1 \quad . \end{aligned}$$

The only difference with the original MAXCUT formulation is the additional constraint $x_s x_t = -1$. This constraint ensures that the vertices s and t are in different partitions and the resulting cut is a feasible solution for (s,t)-MAXCUT.

The algorithm of Goemans and Williamson proceeds by making a semidefinite relaxation, and then rounds the solution based on a random projection. Fortunately, the additional constraint $x_s x_t = -1$ can be easily added to the semidefinite program, and the constraint has no effect on the rounding step. Consequently, we can use the same semidefinite-programming algorithm to approximate the (s,t)-MaxCut problem, which leads to the following.

Theorem 2. *The (s,t)-*MaxCut *problem can be solved by the Goemans-Williamson algorithm [18] with an approximation guarantee $\beta > 0.868$. This also provides a β-factor approximation to the* EventAllPairs+ *problem.*

The proof for the approximation guarantee for the (s,t)-MaxCut problem is virtually the same as the proof for the MaxCut problem [18], and because of space limitations we omit it from this version.

Even though the aforementioned approach uses the function d, which is a metric function (i.e., it satisfies the triangle inequality), note that the approach is general and holds even if the underlying edge cost is not a metric. Specifically, the approach can be applied to arbitrary distance functions and provides the same approximation guarantee. In the case where d is a metric function, as in our case, we can obtain a stronger approximation guarantee.

Theorem 3. *Assuming that d is a metric, then there exists a polynomial-time approximation scheme (*ptas*) for solving* EventAllPairs+. *In other words, for each $\epsilon > 0$, there is a polynomial algorithm with an approximation guarantee of $1 - \epsilon$.*

The proof of the theorem, which relies on the work of Arora et al. [5] for *dense* MaxCut instances, is omitted from this version. Whereas this theorem provides a more tight approximation bound, the involved algorithm is impractical even for large values of ϵ. Consequently, this result serves mostly as a theoretical contribution.

4. ALGORITHMS FOR THE EventTree+ PROBLEM

Next we discuss the EventTree+ problem. Recall that in this case we want to minimize the objective function

$$Q^{+}_{\mathrm{T}}(S) = \lambda\, W(V \setminus S) + D_{\mathrm{T}}(S).$$

This is known as the *prize-collection Steiner-tree* problem (PCST) [17]: we are given a weighted graph with distances between the nodes and prizes for the nodes. The objective is to select a subset of the nodes such that the cost of the tree to connect them plus the prizes of the nodes not in the tree is minimized.

PCST was first considered by Bienstock et al. [10], who provided a 3-approximation. Goemans and Williamson [17] designed a $(2 - \frac{1}{n-1})$-approximation algorithm using a primal–dual schema. In this paper we use a modified version of the algorithm by Johnson et al. [21], which gives 2-approximation, and we call it PD. We choose this algorithm because it is quite fast (runs in time $O(n^2 \log n)$, whereas Goemans and Williamson's algorithm takes $O(n^3 \log n)$ in time) and it is straightforward to implement. The currently best algorithm [4] improves the approximation ratio to $(2 - \epsilon)$, which is more important rather from a theoretical perspective.

Algorithm PD has two phases. In the first phase it *grows* a tree by maintaining and joining components. Each component has a *surplus* and each edge has a *deficit.* Initially every node v is a single component with surplus equal to its weight, and each edge is initialized with a deficit equal to the corresponding distance. The idea is that the surpluses of the components pay for the deficit in the edges inside. The algorithm starts lowering surpluses and deficits until (1) the deficit of an edge becomes zero, in which case the two endpoints are merged into a single component (if the edge is not part of the same component) with the new surplus being the sum of the surpluses of the two components, or (2) the surplus of a component becomes zero, in which case the component is not considered any more to form part of the Steiner tree.

The second phase involves some *pruning* of the tree constructed in the first phase. Starting from the leaves, we move towards the root of the tree, removing a subtree T if the weight of the nodes in T is higher than the cost to connect T to the rest of the tree. The entire algorithm is explained in detail in [21].

We also experiment with a greedy heuristic, similar to GreedyAP, which we call GreedyT. It works as follows: it maintains a set of nodes S, initialized to the empty set. In each iteration it adds to S the node in $V \setminus S$ that provides the maximum decrease to the objective $Q^{+}_{\mathrm{T}}(S)$. The algorithm terminates when $S = V$, or when every node in $V \setminus S$ leads to an increase of $Q^{+}_{\mathrm{T}}(S)$.

5. EXPERIMENTAL EVALUATION

We evaluate our methods on synthetic and real datasets. The real-world datasets are extracted from city sensors (public-bike sharing) and social media (geolocated tweets). We start by describing our datasets.

5.1 Datasets

Synthetic data. We use three synthetic datasets, which we generate as follows. We place the network nodes uniformly at random into the unit square and set the distance between two nodes to their Euclidean distance. The objective with these synthetic datasets is to study whether our methods can detect the planted ground-truth events.

We plant events to the datasets by setting to 1 the values of nodes that occur within the event and to 0 the volues of nodes outside the planted event. Nodes that are close to the border of the events receive a smaller weight. In *Plant1* we plant one spherical event, in *Plant2* we plant two spherical events, and in *Curve* we plant a snake-like event.

To test the robustness of the algorithms, we choose a fraction of the network nodes at random, and assign them an arbitrary weight from a uniform distribution $[0, 1]$. We refer to the size of the fraction as a level of noise.

In addition, we create semi-synthetic datasets by using the coordinates from real-world datasets (see more detailed description below), and by generating synthetic weights as described above. We plant a cross pattern to the Barcelona data (*PlantB*), and two clusters to the NY data (*PlantNY*).

We show the synthetic datasets with 20% noise level at the top row of Figure 1, where the generated event clusters are colored black.

City-sensor data. We use public-bike sharing data from three systems: Barcelona Bicing,[1] Minneapolis Nice Ride,[2]

[1]http://www.bicing.cat

[2]http://www.niceridemn.org

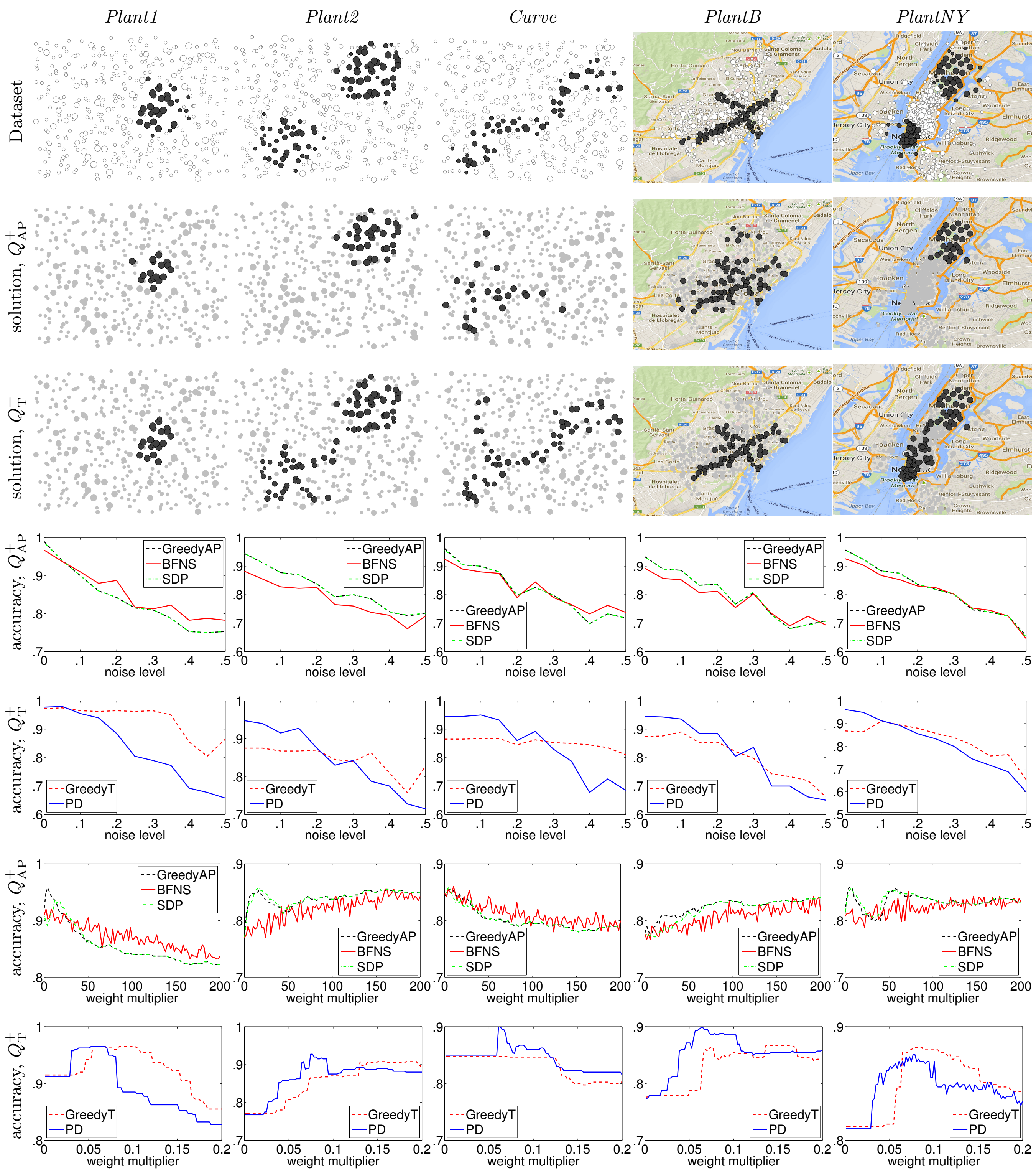


Figure 1: Experiments for synthetic and planted datasets. Each column represents a single dataset. First row contains the datasets and the planted events, contaminated by noise of 20%. Row 2 and 3 row contain events discovered by algorithms for EVENTALLPAIRS+ and EVENTTREE+ respectively that achieve the best accuracy on the particular dataset (with corresponding lambdas). Rows 4–5 contain accuracy as a function of noise level ($\lambda = 100$ for Q^+_{AP} and $\lambda = 0.1$ for Q^+_T). Rows 6–7 contain accuracy as a function of λ (20% noise level).

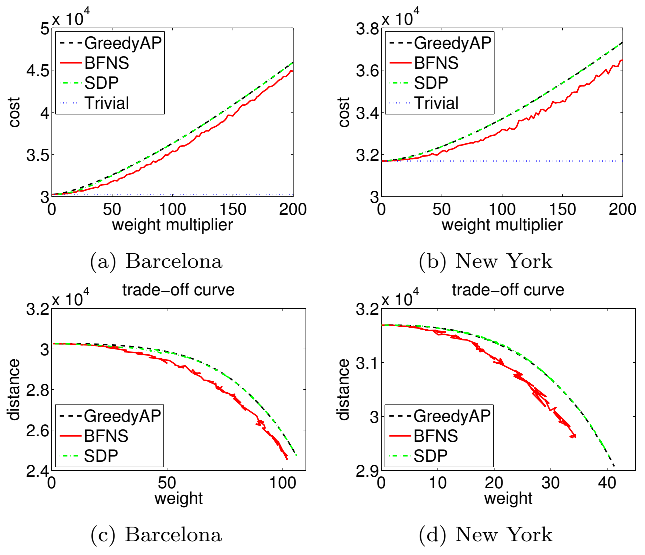


Figure 2: Performance of different algorithms for the EventAllPairs+ problem on Barcelona and NY datasets. The top row shows Q^{+}_{AP} as a function of the weight multiplier λ. The bottom row shows a scatter plot of weight $W(S)$ and the distance $D_{\mathrm{AP}}(V) - D_{\mathrm{AP}}(S)$, parameterized by λ; in both dimensions, larger values are better.

and Washington D.C. Capital bikeshare.[3] In each case, a network node corresponds to a bike station. The activity level of a bike station is the fraction of bikes in the station with respect to the full capacity of the station. To improve the quality of the events discovered by our algorithms we set the weight of a node to be $|x - m|$, where x is the current activity level of a station and m is the typical activity level of the station. As typical activity level we chose the average activity of the station at the same time of the week, over the observation period. The number of stations in the three cities we consider is 420, 145, and 248, respectively, and the data are collected over an observation period of 227, 217, and 273 days, respectively.

Social-media data. We obtain a dataset of geolocated twitter messages for 100 cities. The dataset has been made public by Cheng et al. [15]. For our experiments we focus on the cities of New York and Los Angeles. We start with 302 121 and 254 852 tweets for NY and LA, respectively, which span a period of 329 days. For each city we collapse the locations of the input tweets to centroids, based on proximity, and we use those centroids as network nodes. The activity level of each centroid is the number of tweets during a day. As with the bike-sharing data, we adjust the activity level by considering the difference with respect to the average level at each centroid. The number of centroids is 490 and 487, for NY and LA, respectively. The reason that we collapse tweet locations to centroids is for having enough data at each location to make inferences. The spatial distribution of tweets is highly uneven. To ensure that all the centroids have about the same number of points and, thus, can represent activity of the particular neighborhood correctly, we construct them with the k-means algorithm. The granularity of the result centroids is sufficient to capture accurately the map of activity: in the areas with dense

[3]`http://www.capitalbikeshare.com`

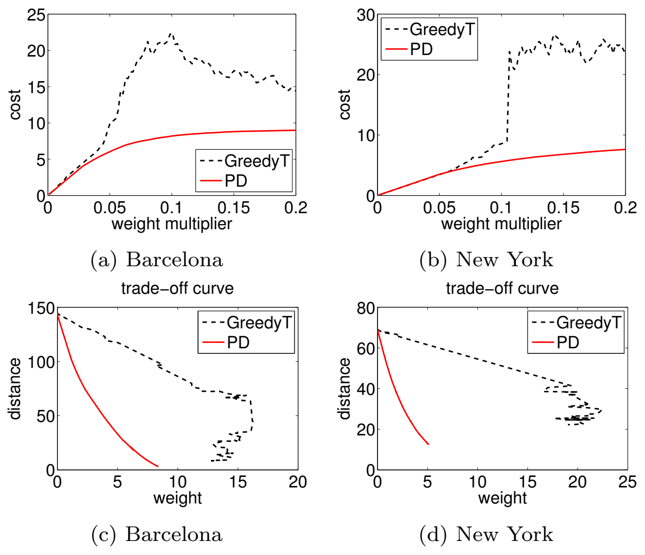


Figure 3: Performance of different algorithms for the EventTree+ problem on Barcelona and New York datasets. The top row contains Q^{+}_{T} as a function of the weight multiplier λ. The bottom row contains a scatter plot of weight $W(V \setminus S)$ and the distance $D_{\mathrm{T}}(S)$, parameterized by λ; in both dimensions, smaller values are better.

activity, such as city centers, each city block is covered by 2-3 centroids, while in the suburbs one centroid represents several blocks.

5.2 Experimental results

Comparison of different algorithms. We have implemented all the algorithms described in the previous sections. For the EventAllPairs+ problem we use the Trivial algorithm as a *baseline*. For the SDP approach, we solve the corresponding semidefinite program using the SDPT3 solver [28], and we round the solution as described in Section 3.2.

We first compare different algorithms in terms of the objective function. The results are shown in Figures 2 and 3 for the Barcelona and NY datasets, respectively. For the EventAllPairs+ problem we see that all algorithms, except Trivial, achieve similar results. In particular, GreedyAP and SDP show almost the same performance and outperform BFNS. For the EventTree+ problem we see that the PD algorithm is far superior to the GreedyT algorithm. Similar trends are observed in all other datasets, and are omitted for paucity of space.

Accuracy on planted events. We test the ability of our algorithms to detect planted events. These events are given at the top row in Figure 1. Because the ground-truth events are known we measure the accuracy of our algorithms, as the fraction of points classified correctly as event points.

Our results, shown in rows 4–5 of Figure 1, provide evidence that our algorithms are capable of discovering the ground-truth events with high accuracy. As expected, accuracy decreases with the noise level, but it remains high, always around 70% even for noise levels of 50%.

With respect to comparing the different algorithms, for the EventAllPairs+ problem, as in the previous experiments, all tested algorithms achieve similar accuracy.

On the other hand, for the EventTree+ problem, the PD algorithm performs better for smaller levels of noise, whereas the GreedyT seems to be more robust for higher noise levels.

We also contrast the solutions found by the two different problem formulations (rows 2–3 of Figure 1), EventAllPairs+ and EventTree+. It is expected that algorithms for the EventAllPairs+ problem will be better suited for round-shaped events, whereas algorithms for the EventTree+ problem will be better in recovering events of arbitrary shape. Our experiments confirm this intuition and show that the algorithms for the tree-based model are more versatile and in general achieve higher accuracy.

Effect of weight multiplier, λ. As we can see from rows 6–7 of Figure 1, the accuracy of the methods depends on the value of the weight multiplier λ. Different values of λ force the algorithms to weight differently the distances with respect to vertex weights, and discover events of different sizes. One way of selecting the parameter is to execute the algorithms with different values of λ, plot the Pareto curve for those values and choose a value that yields the desired trade-off between spanned weight and distance.

Case study. Our next step is to highlight some of the events discovered by our method. For all the events shown as case studies we use the SDP algorithm. To help with selecting λ, we normalize the weights of the vertices so that the maximum weight of each day is equal 1 and the edge distances represent actual kilometric distances. We then use the same value of λ for all the experiments ($\lambda = 20$).

We select top events that correspond to major holidays, shown in Figure 4. The discovered events correspond to places where people would gather in such days, for example, in the city center in Barcelona, and in the National Mall in Washington D.C. In New York City the discovered event is located in the Central Park and the nearby Metropolitan Museum and American Museum of Natural History.

To demonstrate that different days may produce diverse events, we focused on the Barcelona dataset. We aim at reporting k days that have high event score and whose event clusters are spatially non-overlapping (do not have any shared nodes). First we find event clusters and corresponding scores for each day in the dataset and then we sort the days by score in descend order. We select the first day from the list and we keep it in a running set of top days. Then we go through the list and greedily search for a day with an event cluster that does not overlap with any of the clusters of the already obtained top days. If such a day is found, we add it to the top set and we continue traversing the list until the size of the selected top days becomes k. We show the top-3 events in Figure 5. The first event indicates a concert by the Cure music band and the event is around the concert hall, the second event corresponds to the festival of the Poblenou neighborhood, and the third event points out to Halloween activities in el Raval, a famous nightlife spot.

Scalability. The proposed greedy algorithms are efficient and can scale to large networks. We report the scalability behavior of the GreedyAP algorithm. We use the twitter dataset, with tweets from the whole US, to create larger activity networks and we apply the GreedyAP algorithm. Our implementation, executed on an Intel Core i7 machine, with 8 GB RAM and processor running at 2.30GHz, is able to find a solution under 10 seconds when we applied to a fully connected network with 10,000 nodes.

6. CONCLUSION

We formalize the problem of detecting events in activity networks, as a problem of finding compact subgraphs in graphs with vertex weights. Depending on the notion of compactness used—sum of all pairs of distances or Steiner-tree distance—we obtain two different optimization problems. By a reduction to a variant of the MaxCut problem, which we solve with semidefinite programming, and by the use of the primal–dual scheme, we provide approximation algorithms for the two problems considered. In addition, we provide simpler and faster greedy algorithms, for one of which we are able to show approximation guarantees, which rely on the submodularity property of the objective function. Our experiments show that the greedy approaches are more lightweight and perform as well as the more sophisticated approximation algorithms.

The event-detection setting that we consider has many applications. Here we experimented with real-world datasets from city sensors and social media, and we showed that our methods were able to discover successfully real events.

Our work opens many interesting directions for future research. One such direction is to incorporate the temporal dimension of the activity network in the graph-theoretic framework and discover events of varying temporal support.

Acknowledgements

This work was supported by the Academy of Finland grant 118653 (algodan), the EU IP project MULTIPLEX 317532, and the EU FET project SIMPOL 610704. Part of the work was done while Aris Anagnostopoulos was visiting Aalto University under the Aalto Science Institute visiting professor programme.

7. REFERENCES

[1] D. Agarwal, A. McGregor, J. M. Phillips, S. Venkatasubramanian, and Z. Zhu. Spatial scan statistics: Approximations and performance study. *KDD*, 2006.

[2] D. Agarwal, J. M. Phillips, and S. Venkatasubramanian. The hunting of the bump: On maximizing statistical discrepancy. *SODA*, 2006.

[3] L. Akoglu and C. Faloutsos. What is strange in large networks? graph-based irregularity and fraud detection, 2012. Tutorial presented at ICDM.

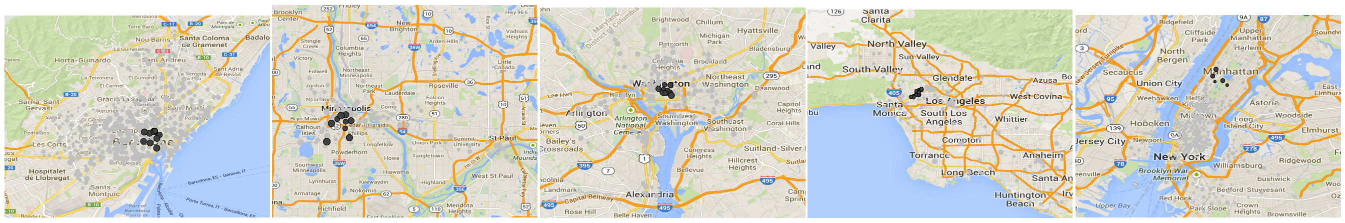
[4] A. Archer, M. H. Bateni, M. T. Hajiaghayi, and H. Karloff. Improved approximation algorithms for prize-collecting Steiner tree and TSP. *SIAM Journal on Computing*, 40(2), 2011.

[5] S. Arora, D. Karger, and M. Karpinski. Polynomial time approximation schemes for dense instances of NP-hard problems. *JCSS*, 58(1), 1999.

[6] Y. Asahiro, K. Iwama, H. Tamaki, and T. Tokuyama. Greedily finding a dense subgraph. *SWAT*, pages 136–148, 1996.

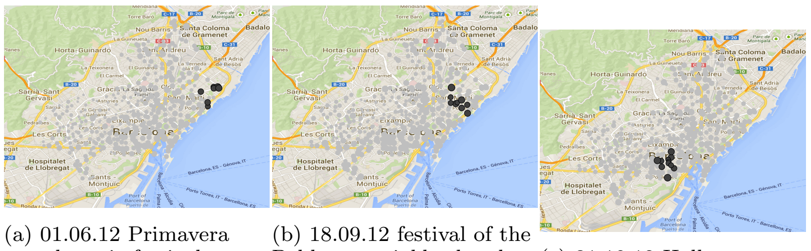
[7] R. Assunção, M. Costa, A. Tavares, and S. Ferreira. Fast detection of arbitrarily shaped disease clusters. *Statistics in medicine*, 25, 2006.

[8] T. Baldwin, P. Cook, B. Han, A. Harwood, S. Karunasekera, and M. Moshtaghi. A support platform for event detection using social intelligence. *EACL*, 2012.



(a) Barcelona: 11.09.12 National Day of Catalonia (b) Minneapolis: 4.07.12 Independence Day (c) Washington, DC: 27.05.13 Memorial Day (d) Los Angeles: 31.05.10 Memorial Day (e) New York: 6.09.10 Labor Day

Figure 4: Public holiday city-events discovered using the SDP algorithm.



(a) 01.06.12 Primavera sound music festival (b) 18.09.12 festival of the Poblenou neighborhood (c) 31.10.12 Halloween

Figure 5: Top-3 diverse events discovered from Barcelona bicing data using the SDP algorithm.

[9] M. H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita. An effective unsupervised network anomaly detection method. *ICACCI*, 2012.

[10] D. Bienstock, M. X. Goemans, D. Simchi-Levi, and D. Williamson. A note on the prize-collecting traveling salesman problem. *Mathematical programming*, 59(1-3), 1993.

[11] B. Boden, S. Günnemann, H. Hoffmann, and T. Seidl. Mining coherent subgraphs in multi-layer graphs with edge labels. *KDD*, 2012.

[12] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander. LOF: identifying density-based local outliers. *SIGMOD*, 2000.

[13] N. Buchbinder, M. Feldman, J. S. Naor, and R. Schwartz. A tight linear time (1/2)-approximation for unconstrained submodular maximization. *FOCS*, 2012.

[14] M. Charikar. Greedy approximation algorithms for finding dense components in a graph. *APPROX*, 2000.

[15] Z. Cheng, J. Caverlee, K. Lee, and D. Z. Sui. Exploring millions of footprints in location sharing services. *ICWSM*, 2011.

[16] U. Feige, G. Kortsarz, and D. Peleg. The dense k-subgraph problem. *Algorithmica*, 29(3), 2001.

[17] M. X. Goemans and D. P. Williamson. A general approximation technique for constrained forest problems. *SIAM Journal on Computing*, 24(2), 1995.

[18] M. X. Goemans and D. P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *JACM*, 42(6), 1995.

[19] V. Guralnik and J. Srivastava. Event detection from time series data. *KDD*, 1999.

[20] N. A. Heard, D. J. Weston, K. Platanioti, and D. J. Hand. Bayesian anomaly detection methods for social networks. *Ann. Appl. Stat.*, 4(2), 2010.

[21] D. S. Johnson, M. Minkoff, and S. Phillips. The prize-collecting Steiner tree problem: theory and practice. *SODA*, 2000.

[22] S. Khuller and B. Saha. On finding dense subgraphs. *ICALP*, 2009.

[23] M. Kulldorff. A spatial scan statistic. *Communications in Statistics-Theory and Methods*, 26(6), 1997.

[24] M. Olson, A. Liu, M. Faulkner, and K. M. Chandy. Rapid detection of rare geospatial events: earthquake warning applications. *DEBS*, 2011.

[25] G. P. Patil and C. Taillie. Upper level set scan statistic for detecting arbitrarily shaped hotspots. *Environmental and Ecological Statistics*, 11, 2004.

[26] S. Seufert, S. J. Bedathur, J. Mestre, and G. Weikum. Bonsai: Growing interesting small trees. In *ICDM*, pages 1013–1018, 2010.

[27] T. Tango and K. Takahashi. A flexibly shaped spatial scan statistic for detecting clusters. *International Journal of Health Geographics*, 4-11, 2005.

[28] K.-C. Toh, M. J. Todd, and R. H. Tütüncü. SDPT3 — a Matlab software package for semidefinite programming, version 1.3. *Optimization methods and software*, 11(1-4), 1999.

[29] C. Tsourakakis, F. Bonchi, A. Gionis, F. Gullo, and M. Tsiarli. Denser than the densest subgraph: extracting optimal quasi-cliques with quality guarantees. *KDD*, 2013.

[30] M. Walther and M. Kaisser. Geo-spatial event detection in the twitter stream. *ECIR*, 2013.

[31] K. Watanabe, M. Ochi, M. Okabe, and R. Onai. Jasmine: a real-time local-event detection system based on geolocation information propagated to microblogs. *CIKM*, 2011.