GAPs: Geospatial Abduction Problems

PAULO SHAKARIAN and V.S. SUBRAHMANIAN
University of Maryland
and
MARIA LUISA SAPINO
Università di Torino

There are many applications where we observe various phenomena in space (e.g. locations of victims of a serial killer), and where we want to infer "partner" locations (e.g. the location where the killer lives) that are geospatially related to the observed phenomena. In this paper, we define *geospatial abduction problems* (GAPs for short). We analyze the complexity of GAPs, develop exact and approximate algorithms (often with approximation guarantees) for these problems together with analyses of these algorithms, and develop a prototype implementation of our GAP framework. We demonstrate accuracy of our algorithms on a real world data set consisting of insurgent IED (improvised explosive device) attacks against US forces in Iraq (the observations were the locations of the attacks, while the "partner" locations we were trying to infer were the locations of IED weapons caches).

Categories and Subject Descriptors: I.2.3 [**Artificial Intelligence**]: Deduction and Theorem Proving—*Nonmonotonic reasoning and belief revision*; I.2.3 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search—*Heuristic methods*; I.2.1 [**Artificial Intelligence**]: Applications and Expert Systems—*Cartography*

General Terms: Theory, Algorithms, Experimentation

Additional Key Words and Phrases: Abduction, Complexity Analysis, Heuristic Algorithms

1. INTRODUCTION

There are numerous applications where we wish to draw geospatial inferences from observations. For example, criminologists [Rossmo and Rombouts 2008; Brantingham and Brantingham 2008] have found that there are spatial relationships between a serial killer's house (the geospatial inference we wish to make), and locations where the crimes were committed (the observations). A marine archaeologist who finds parts of a wrecked ship or its cargo at various locations (the observations) is interested in determining where the main portion of the wreck lies (the geospatial inference). Wildlife experts might find droppings of an endangered species such as the Malayan sun bear (observations) and might want to determine where the bear's den is (the geospatial inference to be made). *In all these cases, we are trying to find a single location that best explains the observations (or the k locations that best*

University of Maryland, College Park, MD 20742, USA. Email: {*pshak*, *vs*}@cs.umd.edu
Universita di Torino, Torino, Italy. Email: mlsapino@di.unito.it
Some of the authors of this paper were funded in part by AFOSR grant FA95500610405 and ARO grants W911NF0910206 and W911NF0910525.

explain the observations). There are two common elements in such applications.

First, there is a set $\mathcal{O}$ of *observations* of the phenomena under study. For the sake of simplicity, we assume that these observations are points where the phenomenon being studied was known to have been present. Second, there is some *domain knowledge* $\mathcal{D}$ specifying known relationships between the geospatial location we are trying to find and the observations. For instance, in the serial killer application, the domain knowledge might tell us that serial killers usually select locations for their crimes that are at least 1.2 km from their homes and at most 3 km from their homes. In the case of the sun bear, the domain knowledge might state that the sun bear usually prefers to have a den in a cave, while in the case of the wreck, it might be usually within a radius of 10 miles of the artifacts that have been found.

The *geospatial abduction problem* (GAP for short) is the problem of finding the most likely *set of locations* that is compatible with the domain knowledge $\mathcal{D}$ and that best "explains" the observations in $\mathcal{O}$. To see why we look for a *set* of locations, we note that the serial killer might be using both his home and his office as launching pads for his attacks. In this case, no single location may best account for the observations. In this paper, we show that many natural problems associated with geospatial abduction are NP-Complete, which cause us to resort to approximation techniques. We then show that certain geospatial abduction problems reduce to several well-studied combinatorial problems that have viable approximation algorithms. We implement some of the more viable approaches with heuristics suitable for geospatial abduction, and test them on a real-world data-set. The organization and main contributions of this paper are as follows.

—Section 2 formally defines geospatial abduction problems (GAPs for short) and Section 3 analyzes their complexity.

—Section 4 develops a "naive" algorithm for a basic geospatial abduction problem called k-**SEP** and shows reductions to set-covering, dominating set, and linear-integer programming that allow well-known algorithms for these problems to be applied to GAPs.

—Section 5 describes two greedy algorithms for k-**SEP** and compares them to a reduction to the set-covering problem.

—Section 6 describes our implementation and shows that our greedy algorithms outperform the set-covering reduction in a real-world application on identifying weapons caches associated with Improvised Explosive Device (IED) attacks on US troops in Iraq. We show that even if we simplify k-**SEP** to only cases where k-means classification algorithms work, our algorithms outperform those.

—Section 7 compares our approach with related work.

2. GEOSPATIAL ABDUCTION PROBLEM (GAP) DEFINITION

Throughout this paper, we assume the existence of a finite, 2-dimensional $M \times N$ space $\mathcal{S}$[1] for some integers $M, N \geq 1$ called the geospatial universe (or just universe). Each point $p \in \mathcal{S}$ is of the form (x, y) where x, y are integers and $0 \leq x \leq M$ and $0 \leq y \leq N$. We assume that all observations we make occur within

[1]We use integer coordinates as most real world geospatial information systems (GIS) systems use discrete spatial representations.

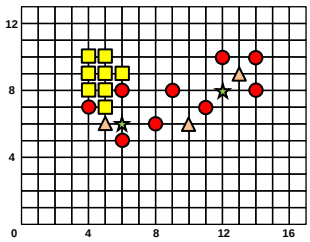

Fig. 1. A space. Red dots denote observations. Yellow squares denote infeasible locations. Green stars show one (0,3) explanation, while pink triangles show another (0,3) explanation.

space $\mathcal{S}$. We use the space shown in Figure 1 throughout this paper to illustrate the concepts we introduce. We assume that $\mathcal{S}$ has an associated distance function d which assigns a non-negative distance to any two points and satisfies the usual distance axioms.[2]

DEFINITION 2.1 OBSERVATION. *An* observation $\mathcal{O}$ *is any finite subset of* $\mathcal{S}$.

Consider the geospatial universe shown in Figure 1. In the serial killer application, the red dots would indicate the locations of the murders, while in the ship-wreck example, they would indicate the locations where artifacts were found. We wish to identify the killer's location (or the sunken ship or the sun bear's den).

As mentioned earlier, there are many constraints that govern where such locations might be. For instance, it is unlikely that the sun-bear's den (or the killer's house or office) is in the water, while the sunken ship is unlikely to be on land.

DEFINITION 2.2 FEASIBILITY PREDICATE. *A feasibility predicate* feas *is a function from* $\mathcal{S}$ *to* $\{TRUE, FALSE\}$.

Thus, $\mathsf{feas}(p) = \mathsf{TRUE}$ means that point p is feasible and must be considered in the search. Figure 1, denotes infeasible places via a yellow square. Throughout this paper, we assume that feas is an arbitrary, but fixed predicate.[3] Further, as feas is defined as a function over $\{\mathsf{TRUE}, \mathsf{FALSE}\}$, it can allow for user input based on analytical processes currently in place. For instance, in the military, analysts often create "MCOO" overlays where "restricted terrain" is deemed infeasible [US Army 1994]. We can also easily express feasibility predicates in a Prolog-style language – we can easily state (in the serial killer example) that point p is considered feasible if p is within R units of distance from some observation and p is not in the water. Likewise, in the case of the sun bear example, the same language might state that p is considered feasible if p is within R_1 units of distance from marks on trees, within R_2 units of scat, and if p has some landcover that would allow the bear to hide. A Prolog-style language that can express such notions of feasibility is the hybrid knowledge base paradigm [Lu et al. 1996] in which Prolog style rules can directly invoke a GIS system.

[2] $d(x,x) = 0; d(x,y) = d(y,x); d(x,y) + d(y,z) \geq d(x,z)$.

[3] We also assume throughout the paper that feas is computable in constant time. This is a realistic assumption, as for most applications, we assume feas to be user-defined. Hence, we can leverage a data-structure indexed with the coordinates of $\mathcal{S}$ to allow for constant-time computation.

DEFINITION 2.3 (α, β) EXPLANATION. *Suppose $\mathcal{O}$ is a finite set of observations, $\mathcal{E}$ is a finite set of points in $\mathcal{S}$, and $\alpha \geq 0$, $\beta > 0$ are some real numbers. $\mathcal{E}$ is said to be an* (α, β) explanation *of $\mathcal{O}$ iff:*

—*$p \in \mathcal{E}$ implies that* $\mathsf{feas}(p) = \mathit{TRUE}$*, i.e. all points in $\mathcal{E}$ are feasible and*

—*$(\forall o \in \mathcal{O})(\exists p \in \mathcal{E})\ \alpha \leq d(p, o) \leq \beta$, i.e. every observation is neither too close nor too far from some point in $\mathcal{E}$.*

Thus, an (α, β) explanation is a set of points (e.g. denoting the possible locations of the home/office of the serial killer or the possible locations of the bear's den). Each point must be feasible and every observation must have an analogous point in the explanation which is neither too close nor too far.

Given an (α, β) explanation $\mathcal{E}$, there may be an observation $o \in \mathcal{O}$ such that there are two (or more) points $p_1, p_2 \in \mathcal{E}$ satisfying the conditions of the second bullet above. If $\mathcal{E}$ is an explanation for $\mathcal{O}$, a *partnering function* $\wp_{\mathcal{E}}$ is a function from $\mathcal{O}$ to $\mathcal{E}$ such that for all $o \in \mathcal{O}$, $\alpha \leq d(\wp_{\mathcal{E}}(o), o) \leq \beta$. $\wp_{\mathcal{E}}(o)$ is said to be o's *partner* according to the partnering function $\wp_{\mathcal{E}}$. We now present a simple example of (α, β) explanations.

EXAMPLE 2.1. *Consider the observations in Figure 1 and suppose $\alpha = 0, \beta = 3$. Then the two green stars denote an (α, β) explanation, i.e. the set $\{(6, 6), (12, 8)\}$ is a $(0, 3)$ explanation. So is the set of three pink triangles, i.e. the set $\{(5, 6), (10, 6), (13, 9)\}$ is also an $(0, 3)$ explanation.*

The basic problem that we wish to solve in this paper is the following.

The Simple (α, β) Explanation Problem (SEP).
INPUT: Space $\mathcal{S}$, a set $\mathcal{O}$ of observations, a feasibility predicate feas, and numbers $\alpha \geq 0$, $\beta > 0$.
OUTPUT: "Yes" if there exists an (α, β) explanation for $\mathcal{O}$ — "no" otherwise.

A variant of this problem is the k-**SEP** problem which requires, in addition, that $\mathcal{E}$ contains k elements or less, for $k < |\mathcal{O}|$. Yet another variant of the problem tries to find an explanation $\mathcal{E}$ that is "best" according to some cost function.

DEFINITION 2.4 COST FUNCTION χ. *A cost function χ is a mapping from explanations to non-negative reals.*

We will assume that cost functions are designed so that the smaller the value they return, the more desirable an explanation is. Some example cost functions are given below. The simple one below merely looks at the mean distances between observations and their partners.

EXAMPLE 2.2 MEAN-DISTANCE. *Suppose $\mathcal{S}, \mathcal{O}, \mathsf{feas}, \alpha, \beta$ are all given and suppose $\mathcal{E}$ is an (α, β) explanation for $\mathcal{O}$ and $\wp_{\mathcal{E}}$ is a partnering function. We could initially set the cost of an explanation $\mathcal{E}$ (with respect to this partnering function) to be:*

$$\chi_{\wp\varepsilon}(\mathcal{E}) = \frac{\Sigma_{o \in \mathcal{O}}\, d(o, \wp_{\mathcal{E}}(o))}{|\mathcal{O}|}.$$

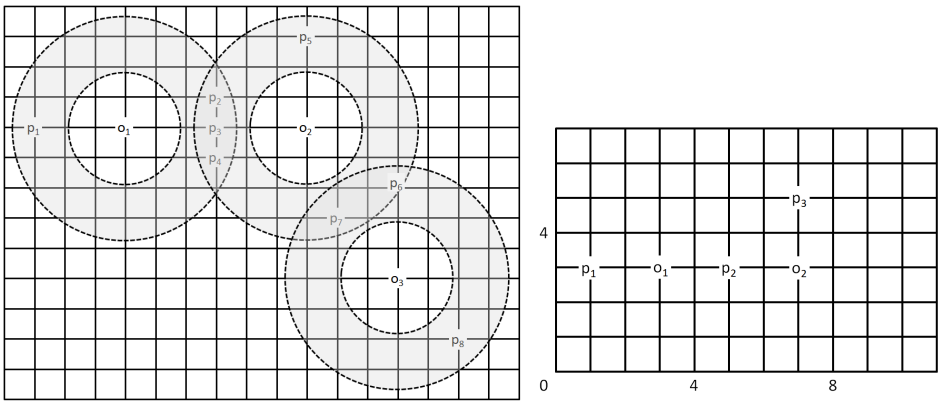

Fig. 2. **Left:** Points $\{o_1, o_2, o_3\}$ indicate locations of evidence of the Malayan sun bear (we shall refer to these as set $\mathcal{O}$). Points $\{p_1, p_2, \ldots, p_8\}$ indicate feasible dwellings for the bear. The concentric rings around each element of $\mathcal{O}$ indicate the distance $\alpha = 1.7km$ and $\beta = 3.7km$. **Right:** Points $\{p_1, p_2, p_3\}$ are feasible for crime-scenes $\{o_1, o_2\}$. $\{p_1, p_2\}$ are safe-houses within a distance of $[1, 2]$ km. from crime scene o_1 and $\{p_2, p_3\}$ are safe-houses within a distance of $[1, 2]$ km. from crime scene o_2.

Suppose $ptn(\mathcal{E})$ is the set of all partner functions for $\mathcal{E}$ in the above setting. Then we can set the cost of $\mathcal{E}$ as:

$$\chi_{mean}(\mathcal{E}) \;=\; inf\{\chi_{\wp\mathcal{E}}(\mathcal{E}) \mid \wp_{\mathcal{E}} \in ptn(\mathcal{E})\}.$$

The above definition removes reliance on a single partnering function as there may be several partnering functions associated with a single explanation. We illustrate this definition using our sun bear example.

EXAMPLE 2.3. *Wildlife experts have found droppings and other evidence of the Malayan sun bear in a given space, $\mathcal{S}$, depicted in Figure 2. Points $\{o_1, o_2, o_3\}$ indicate locations of evidence of the Malayan sun bear (we shall refer to these as set $\mathcal{O}$). Points $\{p_1, p_2, \ldots, p_8\}$ indicate feasible dwellings for the bear. The concentric rings around each element of $\mathcal{O}$ indicate the distance $\alpha = 1.7km$ and $\beta = 3.7km$. The set $\{p_3, p_6\}$ is a valid $(1.7, 3.7)$ explanation for the set of evidence, $\mathcal{O}$. However, we note that observation o_2 can be partnered with either point. If we are looking to minimize distance, we notice that $d(o_2, p_3) = 3km$ and $d(o_2, p_6) = 3.6km$, hence p_3 is the partner for o_2 such that the distance is minimized.*

We now define an "optimal" explanation as one that minimizes cost.

DEFINITION 2.5. *Suppose $\mathcal{O}$ is a finite set of observations, $\mathcal{E}$ is a finite set of points in $\mathcal{S}$, $\alpha \geq 0$, $\beta > 0$ are some real numbers, and χ is a cost function. $\mathcal{E}$ is said to be an* optimal (α, β) *explanation iff $\mathcal{E}$ is an (α, β) explanation for $\mathcal{O}$ and there is no other (α, β) explanation $\mathcal{E}'$ for $\mathcal{O}$ such that $\chi(\mathcal{E}') < \chi(\mathcal{E})$.*

We present an example of optimal (α, β) explanations below.

EXAMPLE 2.4. *Consider the sun bear from Example 2.3 whose behavior is depicted in Figure 2 (left). While $\{p_3, p_6\}$ is a valid solution for the k-**SEP** problem*

($k = 2$), it does not optimize mean distance. In this case the mean distance would be $3km$. However, the solution $\{p_3, p_7\}$ provides a mean-distance of $2.8km$.

Suppose we are tracking a serial killer who has struck at locations $\mathcal{O} = \{o_1, o_2\}$. The points $\{p_1, p_2, p_3\}$ are feasible locations as safe-houses for the killer (partners). This is depicted in Figure 2 (right). Based on historical data, we know that serial killers strikes are at least $1km$ away from a safe-house and at most $2km$ from the safe house ($\alpha = 1$, $\beta = 2$). Thus, for $k = 2$, any valid explanation of size 2 provides an optimal solution wrt mean-distance as every feasible location for a safe-house is within $2km$ of a crime scene.

We are now ready to define the cost-based explanation problem.

The Cost-based (α, β) Explanation Problem.

INPUT: Space $\mathcal{S}$, a set $\mathcal{O}$ of observations, a feasibility predicate feas, numbers $\alpha \geq 0$, $\beta > 0$, a cost function χ and a real number $v > 0$.

OUTPUT: "Yes" if there exists an (α, β) explanation $\mathcal{E}$ for $\mathcal{O}$ such that $\chi(\mathcal{E}) \leq v$ — "no" otherwise.

It is easy to see that standard classification problems like k-means [4] can be captured within our framework by simply assuming that $\alpha = 0$, $\beta > max(M, N)^2$ and that all points are feasible. In contrast, standard classification algorithms cannot take feasibility into account - and this is essential for the above types of applications.

3. COMPLEXITY OF GAP PROBLEMS

SEP can be easily solved in PTIME. Given a set $\mathcal{O}$ of observations, for each $o \in \mathcal{O}$, let $P_o = \{p \in \mathcal{S} \mid feas(p) = \mathsf{TRUE} \wedge \alpha \leq d(p, o) \leq \beta\}$. If $P_o \neq \emptyset$ for each o, we return "yes". We call this algorithm STRAIGHTFORWARD-SEP. Another algorithm would merely find the set F of all feasible points and return "yes" iff for every observation o, there is at least one point $p \in F$ such that $\alpha \leq d(p, o) \leq \beta$. In this case, F is the explanation produced - but it is a very poor explanation. In the serial killer example, F merely tells the police to search all feasible locations without trying to do anything intelligent. k-SEP allows the user to constrain the size of the explanation so that "short and sweet" explanations that are truly meaningful are produced. The following result states that k-**SEP** is NP-Complete - the proof is a reduction from *Geometric Covering by Discs* (GCD) [Johnson 1982].

THEOREM 3.1. *k-**SEP** is NP-Complete.*

In the associated optimization problem with k-**SEP**, we wish to produce an explanation of minimum cardinality. Note that minimum cardinality is a common criterion for parsimony in abduction problems [Reggia and Peng 1990]. We shall refer to this problem as MINSEP. This problem is obviously NP-hard by Theorem 3.1. We can adjust STRAIGHTFORWARD-SEP to find a solution to MINSEP by finding the minimum hitting set of the P_o's.

EXAMPLE 3.1. *Consider the serial killer scenario in Example 2.4 and Figure 2 (right). Crime scene (observation) o_1 can be partnered with two possible safe-houses*

[4] See [Alpaydin 2010] for a survey on classification work.

*$\{p_1, p_2\}$ and crime scene o_2 can be partnered with $\{p_2, p_3\}$. We immediately see that the potential safe house located at p_2 is in both sets. Therefore, p_2 is an explanation for both crime scenes. As this is the only such point, we conclude that $\{p_2\}$ is the minimum-sized solution for the **SEP** problem. However, while it is possible for STRAIGHTFORWARD-SEP to return this set, there are no assurances it does. As we saw in Example 2.4, $\mathcal{E} = \{p_1, p_2\}$ is a solution to **SEP**, although a solution with lower cardinality ($\{p_2\}$) exists. This is why we introduce the MINSEP problem.*

With the complexity of k-**SEP**, the following corollary tells us the complexity class of the Cost-based Explanation problem. We show this reduction by simply setting the cost function $\chi(\mathcal{E}) = |\mathcal{E}|$.

COROLLARY 3.1. *Cost-based Explanation is NP-Complete.*

As described earlier, MINSEP has the feel of a set-covering problem. Although the generalized cost-based explanation cannot be directly viewed with a similar intuition (as the cost maps explanations to reals – not elements of $\mathcal{S}$), there is an important variant of the Cost-based problem that does. We introduce weighted **SEP**, or **WT-SEP** below.

Weighted Spatial Explanation. (**WT-SEP**)
INPUT: A space $\mathcal{S}$, a set $\mathcal{O}$ of observations, a feasibility predicate feas, numbers $\alpha \geq 0$, $\beta > 0$, a weight function $c : \mathcal{S} \rightarrow \Re$, and a real number $v > 0$.
OUTPUT: "Yes" if there exists an (α, β) explanation $\mathcal{E}$ for $\mathcal{O}$ such that $\sum_{p \in \mathcal{E}} c(p) \leq v$ — "no" otherwise.

In this case, we can easily show NP-Completeness by reduction from k-**SEP**, we simply set the weight for each element of $\mathcal{S}$ to be one, causing $\sum_{p \in \mathcal{E}} c(p)$ to equal the cardinality of $\mathcal{E}$.

COROLLARY 3.2. ***WT-SEP** is NP-Complete.*

Cost-based explanation problems presented in this section are very general. While the complexity results hold for an arbitrary function in a general case, we also consider specific functions as well. Below we present the total-distance minimization explanation problem (**TD-SEP**). This is a problem where we seek to minimize the sum of distances between observations and their closest partners while imposing a restriction on cardinality.

Total Distance Minimization Explanation Problem. (**TD-SEP**)
For space $\mathcal{S}$, let $d : \mathcal{S} \times \mathcal{S} \rightarrow \Re$ be the Euclidean distance between two points in $\mathcal{S}$.
INPUT: A space $\mathcal{S}$, a set $\mathcal{O}$ of observations, a feasibility predicate feas, numbers $\alpha \geq 0$, $\beta > 0$, positive integer $k < |\mathcal{O}|$, and real number $v > 0$.
OUTPUT: "Yes" if there exists an (α, β) explanation $\mathcal{E}$ for $\mathcal{O}$ such that $|\mathcal{E}| = k$ and $\sum_{o_i \in \mathcal{O}} \min_{p_j \in \mathcal{E}} d(o_i, p_j) \leq v$ — "no" otherwise.

THEOREM 3.2. ***TD-SEP** is NP-Complete.*

The NP-hardness of the **TD-SEP** is based on a reduction from the k-Median Problem [Papadimitriou 1981]. This particular reduction (details in the appendix)

Algorithm 1 (NAIVE-KSEP-EXACT)

INPUT: Space $\mathcal{S}$, a set $\mathcal{O}$ of observations, a feasibility predicate feas, real numbers $\alpha \geq 0$, $\beta > 0$, and natural number $k > 0$
OUTPUT: Set $\mathcal{E} \subseteq \mathcal{S}$ of size k (or less) that explains $\mathcal{O}$

(1) Let M be a matrix array of pointers to binary string $\{0,1\}^{|\mathcal{O}|}$. M is of the same dimensions as $\mathcal{S}$. Each element in M is initialized to NULL. For a given $p \in \mathcal{S}$, $M[p]$ is the place in the array.
(2) Let L be a list of pointers to binary strings. L is initialized as null.
(3) For each $o_i \in \mathcal{O}$ do the following
  (a) Determine all points $p \in S$ such that $\alpha \leq d(o,p) \leq \beta$ such that feas(p) = TRUE.
  (b) For each of these points, p, if $M[p]$ = NULL then initialize a new array where only bit i is set to 1. Then add a pointer to $M[p]$ in L.
  (c) Otherwise, set bit i of the existing array to 1.
(4) For any k elements of L (actually the k elements pointed to by elements of L), we shall designate $\ell_1, \ldots, \ell_j, \ldots \ell_k$ as the elements. We will refer to the ith bit of element ℓ_j as $\ell_j(i)$.
(5) Exhaustively generate all possible combinations of k elements of L until one such combination is found where $\forall i \in [1, |\mathcal{O}|]$, $\sum_{j=1}^{k}(\ell_j(i)) > 0$
(6) If no such combination is found, return NO. Otherwise, return the first combination that was found.

also illustrates how the k-median problem is a special case of GAPs, but k-median problems cannot handle arbitrary feasibility predicates of the kind that occur in real-life geospatial reasoning. The same argument applies to k-means classifiers as well.

4. EXACT ALGORITHM FOR GAP PROBLEMS

This section presents four exact approaches to solve k-**SEP** and **WT-SEP**. First, we provide an enumerative approach that exhaustively searches for an explanation. Then, we show that the problem reduces to set-cover, dominating set, and linear-integer programming. Existing algorithms for these problems can hence be used directly. Throughout this section, we shall use the symbols Δ to represent the bound on the number of partners that can be associated with a single observation and f to represent the bound on the number of observations supported by a single partner. Note that both values are bounded by $\pi(\beta^2 - \alpha^2)$, however they can be much less in practice – specifically f is normally much smaller than Δ.

4.1 Naive Exact Algorithm

We now show correctness of NAIVE-KSEP-EXACT. This algorithm provides an exact solution to k-**SEP** but takes exponential time (in k). The algorithm first identifies a set L of all elements of $\mathcal{S}$ that could be possible partners for $\mathcal{O}$. Then, it considers all subsets of L of size less than or equal to k. It does this until it identifies one such subset as an explanation.

PROPOSITION 4.1. *If there is a k-sized simple (α, β) explanation for $\mathcal{O}$, then NAIVE-KSEP-EXACT returns an explanation. Otherwise, it returns NO.*

Finally, we have the complexity of the algorithm.

PROPOSITION 4.2. *The complexity of NAIVE-KSEP-EXACT is* $O(\frac{1}{(k-1)!}(\pi(\beta^2 - \alpha^2)|\mathcal{O}|)^{(k+1)})$.

An exact algorithm for the cost-based explanation problems follows trivially from the NAIVE-KSEP-EXACT algorithm by adding the step of computing the value for χ for each combination. Provided this computation takes constant time, this does not affect the $O(\frac{1}{(k-1)!}(\pi(\beta^2 - \alpha^2)|\mathcal{O}|)^{(k+1)})$ run time of that algorithm.

4.2 An Exact Set-Cover Based Approach

We now show that k-**SEP** polynomially reduces to an instance of the popular set-covering problem [Karp 1972] which allows us to directly apply the well-known greedy algorithm reviewed in Paschos [1997]. **Set-Cover** is defined as follows.

The Set-Cover Problem. (**Set-Cover**)
INPUT: Set of elements, E and a family of subsets of E, $F \equiv \{S_1, \ldots, S_{max}\}$, and positive integer k.
OUTPUT: "Yes" if there exists a k-sized subset of F, F_k, such that $\bigcup_{i=1}^{k}\{S_i \in F_k\} \equiv E$.

Through a simple modification of NAIVE-KSEP-EXACT, we can take an instance of k-**SEP** and produce an instance of **Set-Cover**. We run the first four steps, which only takes $O(\Delta \cdot |\mathcal{O}|)$ time by the proof of Proposition 4.2.

THEOREM 4.1. *k-**SEP** polynomially reduces to **Set-Cover**.*

EXAMPLE 4.1. *Consider the serial killer scenario in Example 2.4 and Figure 2 (right). Suppose we want to solve this problem as an instance of k-**SEP** by a reduction to set-cover. We consider the set of crime-scene locations, $\mathcal{O} \equiv \{o_1, o_2\}$ as the set we wish to cover. We obtain our covers from the first four steps of NAIVE-KSEP-EXACT. Let us call the result list L. Hence, we can view the values of the elements in L as the following sets $S_1 \equiv \{o_1\}, S_2 \equiv \{o_1, o_2\}, S_3 \equiv \{o_2\}$. These correspond with points p_1, p_2, p_3 respectively. As S_2 covers $\mathcal{O}$, p_2 is an explanation.*

The traditional approach for approximation of set-cover has a time complexity of $O(|E| \cdot |F| \cdot size)$, where $size$ is the cardinality of the largest set in the family F (i.e. $size = \max_{i \leq |F|} |S_i|$). This approach obtains an approximation ratio of $1 + \ln(size)$ [Paschos 1997]. As f is the quantity of the largest number of observations supported by a single partner, the approximation ratio for k-**SEP** using a greedy-scheme after a reduction from set-cover is $1+\ln(f)$. The NAIVE-KSEP-SC algorithm below leverages the above reduction to solve the k-SEP problem.

PROPOSITION 4.3. *NAIVE-KSEP-SC has a complexity of $O(\Delta \cdot f \cdot |\mathcal{O}|^2)$ and an approximation ratio of $1 + \ln(f)$.*

PROPOSITION 4.4. *A solution $\mathcal{E}$ to NAIVE-KSEP-SC provides a partner to every observation in $\mathcal{O}$ if a partner exists – otherwise, it returns IMPOSSIBLE.*

The algorithm NAIVE-KSEP-SC is a naive, straight-forward application of the $O(|E| \cdot |F| \cdot size)$ greedy approach for set-cover as presented in Paschos [1997].

Algorithm 2 (NAIVE-KSEP-SC)

INPUT: Space $\mathcal{S}$, a set $\mathcal{O}$ of observations, a feasibility predicate feas, and real numbers $\alpha \geq 0$, $\beta > 0$
OUTPUT: Set $\mathcal{E} \subseteq \mathcal{S}$ that explains $\mathcal{O}$

(1) Initialize list $\mathcal{E}$ to null
(2) Let M be a matrix array of the same dimensions as $\mathcal{S}$ of lists of pointers initialized to null. For a given $p \in \mathcal{S}$, $M[p]$ is the place in the array.
(3) Let L be a list of pointers to lists in M, L is initialized to null.
(4) Let $\mathcal{O}'$ be an array of Booleans of length $|\mathcal{O}|$. $\forall i \in [1, |\mathcal{O}|]$, initialize $\mathcal{O}'[i] =$ TRUE. For some element $o \in \mathcal{O}$, $\mathcal{O}'[o]$ is the corresponding space in the array.
(5) Let numObs $= |\mathcal{O}|$
(6) For each element $o \in \mathcal{O}$, do the following.
  (a) Determine all elements $p \in \mathcal{S}$ such that feas$(p) =$ TRUE and $d(o, p) \in [\alpha, \beta]$
  (b) If there does not exist a $p \in \mathcal{S}$ meeting the above criteria, then terminate the program and return IMPOSSIBLE.
  (c) If $M[p] =$ null then add a pointer to $M[p]$ to L
  (d) Add a pointer to o to the list $M[p]$.
(7) While numObs > 0 loop
  (a) Initialize pointer *cur_ptr* to null
  (b) Initialize integer *cur_size* to 0
  (c) For each $ptr \in L$, do the following:
    i. Initialize integer *this_size* to 0
    ii. Let $M[p]$ be the element of M pointed to by *ptr*
    iii. For each *obs_ptr* in the list $M[p]$, do the following
      A. Let i be the corresponding location in array $\mathcal{O}'$ to *obs_ptr*
      B. If $\mathcal{O}'[i] =$ TRUE, increment *this_size* by 1
    iv. If *this_size* $>$ *cur_size*, set *cur_size* = *this_size* and have *cur_ptr* point to $M[p]$
  (d) Add p to $\mathcal{E}$
  (e) For every *obs_ptr* in the list pointed to by *cur_ptr*, do the following:
    i. Let i be the corresponding location in array $\mathcal{O}'$ to *obs_ptr*
    ii. If $\mathcal{O}'[i]$, then set it to FALSE and decrement numObs by 1
  (f) Add the location in space $\mathcal{S}$ pointed to by *cur_ptr* to $\mathcal{E}$
(8) Return $\mathcal{E}$

We note that it is possible to implement a heap to reduce the time-complexity to $O(\Delta \cdot f \cdot |\mathcal{O}| \cdot \lg(\Delta \cdot |\mathcal{O}|))$ - avoiding the cost of iterating through all possible partners in the inner-loop.

In addition to the straightforward greedy algorithm for set-covering, there are several other algorithms that provide different time complexity/approximation ratio combinations. However, with a reduction to the set-covering problem we must consider the result of Lund and Yannakakis [1994] which states that set-cover cannot be approximated within a ratio $c \cdot log(n)$ for any $c < 0.25$ (where n is the number of subsets in the family F) unless $NP \subseteq DTIME[n^{\texttt{poly log } n}]$.

A reduction to set-covering has the advantage of being straightforward. It also allows us to leverage the wealth of approaches developed for this well-known problem. In the next section, we show that k-**SEP** reduces to the dominating set problem as

Algorithm 3 (KSEP-TO-DOMSET)

INPUT: Space $\mathcal{S}$, a set $\mathcal{O}$ of observations, a feasibility predicate feas, and real numbers $\alpha \geq 0$, $\beta > 0$
OUTPUT: Graph $G_{\mathcal{O}}$ for use in an instance of a **DomSet** problem

(1) Let $G_{\mathcal{O}} = (V_{\mathcal{O}}, E_{\mathcal{O}})$ be a graph. Set $V_{\mathcal{O}} = \mathcal{S}$ and $E_{\mathcal{O}} = \emptyset$.
(2) Let S be a mapping defined as $S : \mathcal{S} \rightarrow V_{\mathcal{O}}$. In words, S takes elements of the space and returns nodes from $G_{\mathcal{O}}$ as defined in the first step. This mapping does not change during the course of the algorithm.
(3) For each $o_i \in \mathcal{O}$ do the following
    (a) Determine all points $p \in S$ that are such that $\alpha \leq d(o, p) \leq \beta$. Call this set P_i
    (b) For all $p \in P_i$ calculate feas(p). If feas(p) = FALSE, remove p from P_i.
    (c) Let $V_i = \{v \in V_{\mathcal{O}} | \exists p \in P_i \textit{ such that } S(p) = v\}$.
    (d) Add $|P_i|$ *new* nodes to $V_{\mathcal{O}}$. Add these nodes to V_i as well.
    (e) For every pair of nodes $v_1, v_2 \in V_i$, add edge (v_1, v_2) to $E_{\mathcal{O}}$.
(4) Remove all $v \in V_{\mathcal{O}}$ where there does not exist an v' such that $(v, v') \in E_{\mathcal{O}}$
(5) If any $P_i \equiv \emptyset$ return IMPOSSIBLE. Otherwise return $G_{\mathcal{O}}$.

well. We then explore alternate approximation techniques based on this reduction.

4.3 An Exact Dominating Set Based Approach

We show below that k-**SEP** also reduces to the well known dominating set problem (**DomSet**) [Garey and Johnson 1979] allowing us to potentially leverage fast algorithms such as the randomized-distributed approximation scheme in Jia et al. [2002]. **DomSet** is defined as follows.

Dominating Set. (**DomSet**)
INPUT: Graph $G = (V, E)$ and positive integer $K \leq |V|$.
OUTPUT: "Yes" if there is a subset $V' \subset V$ such that $|V'| \leq K$ and such that every vertex $v \in V - V'$ is joined to at least one member of V' by an edge in E.

As the dominating set problem relies on finding a certain set of nodes in a graph, then, unsurprisingly, our reduction algorithm, Algorithm 3, takes space $\mathcal{S}$, an observation set $\mathcal{O}$, feasibility predicate feas, and numbers α, β and returns graph $G_{\mathcal{O}}$ based on these arguments.

We now present an example to illustrate the relationship between a dominating set of size k in $G_{\mathcal{O}}$ and a k-sized simple (α, β) explanation for $\mathcal{O}$. The following example illustrates the relationship between a k-**SEP** problem and **DomSet**.

EXAMPLE 4.2. *Consider the serial killer scenario in Example 2.4, pictured in Figure 2 (right). Suppose we want to solve this problem as an instance of k-**SEP** by a reduction to **DomSet**. We want to find a 1-sized simple (α, β) explanation (safe-house) for $\mathcal{O}$ (the set of crime scenes, $\{o_1, o_2\}$). Suppose that after running an algorithm such as STRAIGHFORWARD-SEP, we find that $\{p_1, p_2, p_3\}$ are elements of $\mathcal{S}$ that are feasible. $\{p_1, p_2\}$ are all within a distance of α, β from o_1 and $\{p_2, p_3\}$ are all within a distance of α, β from o_2. We run KSEP-TO-DOMSET which creates graph, $G_{\mathcal{O}}$. Refer to Figure 3 for the graph. We can see that $\{p_2\}$ is a 1-sized dominating sets for $G_{\mathcal{O}}$, hence a 1-sized explanation for $\mathcal{O}$.*

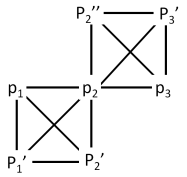

Fig. 3. Results of KSEP-TO-DOMSET based on data seen in Figure 2 (right). Note that $\{p_1, p_2, p_1', p_2'\}$ form a complete graph and $\{p_2, p_3, p_2'', p_3'\}$ also form a complete graph. Note that $\{p_2\}$ is a dominating set of size 1. Hence, $\{p_2\}$ is a 1-sized simple (α, β) explanation for $\mathcal{O}$, as depicted in Figure 2 (right).

We notice that the inner loop of KSEP-TO-DOMSET is bounded by $O(\Delta)$ operations and the outer loop will iterate $|\mathcal{O}|$ times. Thus, the complexity of KSEP-TO-DOMSET is $O(\Delta \cdot |\mathcal{O}|)$.

PROPOSITION 4.5. *The complexity of KSEP-TO-DOMSET is $O(\Delta \cdot |\mathcal{O}|)$.*

Example 4.2 should give us some intuition into why the reduction to **DomSet** works. We provide the formal proof in the Appendix.

THEOREM 4.2. *k-**SEP** is polynomially reducible to **DomSet**.*

The straightforward approximation scheme for **DomSet** is to view the problem as an instance of **Set-Cover** and apply a greedy algorithm. The reduction would view the set of vertices in $G_{\mathcal{O}}$ as the elements, and the family of sets as each vertex and its neighbors. This results in both a greater complexity and a worse approximation ratio when compared with the reduction directly to **Set-Cover**.

PROPOSITION 4.6. *Solving k-**SEP** by a reduction to **DomSet** using a straightforward greedy approach has time-complexity $O(\Delta^3 \cdot f \cdot |\mathcal{O}|^2)$ and an approximation ratio bounded by $O(1 + \ln(2 \cdot f \cdot \Delta))$.*

There are other algorithms to approximate **DomSet** [Jia et al. 2002; Kuhn and Wattenhofer 2003]. By leveraging Jia et al. [2002], we can obtain an improved complexity while retaining the same approximation ratio as the greedy approach.

PROPOSITION 4.7. *Solving k-**SEP** by a reduction to **DomSet** using the distributed, randomized algorithm presented in Jia et al. [2002] has a time complexity $O(\Delta \cdot |\mathcal{O}| + \ln(2 \cdot \Delta \cdot |\mathcal{O}|) \cdot \ln(2 \cdot \Delta \cdot f))$ with high probability and approximation ratio of $O(1 + \ln(2 \cdot f \cdot \Delta))$.*

Hence, although a reduction to dominating set generally gives us a worse approximation guarantee, we can (theoretically) outperform set-cover with the randomized algorithm for dominating set in terms of complexity.

4.4 An Exact Integer Linear Programming based Approach

Given an instance of k-**SEP**, we show how to create a set of integer constraints that if solved, will yield a solution to the problem.

DEFINITION 4.1 OPT-KSEP-IPC. *The k-**SEP** integer programming constraints (OPT-KSEP-IPC) require the following information, obtained in $O(|\mathcal{O}| \cdot \pi(\beta^2 - \alpha^2)$ time:*

—*Let L be the set of all possible partners generated in the first four steps of* NAIVE-KSEP-EXACT.

—*For each $p \in L$, let $str(p)$ be the string of $|\mathcal{O}|$ bits, where bit $str(p)_i$ is 1 if p is a partner of the ith observation (this is also generated in the first four steps of* NAIVE-KSEP-EXACT*).*

For each $p_j \in L$, let $x_j \in \{0,1\}$. $x_j = 1$ iff p_j is in $\mathcal{E}$.
Then KSEP-IPC *consists of the following:*
Minimize $\sum_{p_j \in L} x_j$ ***subject to***

(1) $\forall o_i \in \mathcal{O}$, $\sum_{p_j \in L} x_j \cdot str(p_j)_i \geq 1$

(2) $\forall p_j \in L$, $x_j \in \{0,1\}$ *(for the relaxed linear program: $x_j \leq 1$)*

PROPOSITION 4.8. *OPT-KSEP-IPC consists of $O(|\mathcal{O}|\pi(\beta^2 - \alpha^2))$ variables and $O(|\mathcal{O}| \cdot \pi(\beta^2 - \alpha^2))$ constraints.*

PROPOSITION 4.9. *For a given instance of the optimization version k-**SEP**, if* OPT-KSEP-IPC *is solved, then $\bigcup_{p_j \in L_{x_j=1}} p_j$ is an optimal solution to k-**SEP**.*

EXAMPLE 4.3. *Consider the serial killer scenario in Example 2.4, pictured in Figure 2 (right). Suppose we want to solve this problem as an instance of* MINSEP. *We would set up the constraints as follows:*
Minimize $x_1+x_2+x_3$ ***subject to*** $1 \cdot x_1+1 \cdot x_2+0 \cdot x_3 \geq 1$ *and* $0 \cdot x_1+1 \cdot x_2+1 \cdot x_3 \geq 1$, *where* $x_1, x_2, x_3 \in \{0,1\}$
Obviously, setting $x_1 = 0, x_2 = 1, x_3 = 0$ provides an optimal solution. Hence, as x_2 is the only non-zero variable, p_2 is the explanation for the crime-scenes.

A solution to the constraints OPT-KSEP-IPC can be approximated using the well-known "rounding" technique [Hochbaum 1982; Vazirani 2004] that relaxes constraints. We present an OPT-KSEP-IPC using rounding.

Algorithm 4 (NAIVE-KSEP-ROUND)

INPUT: Space $\mathcal{S}$, a set $\mathcal{O}$ of observations, a feasibility predicate feas, and real numbers $\alpha \geq 0$, $\beta > 0$
OUTPUT: Set $\mathcal{E} \subseteq \mathcal{S}$ that explains $\mathcal{O}$

(1) Run the first four steps of NAIVE-KSEP-EXACT
(2) Solve the relaxation of OPT-KSEP-IPC
(3) For the $o \in \mathcal{O}$ with the most possible partners, let Δ be the number of possible partners associated with o. This can be done in line 1
(4) Return all $p_j \in L$ where $x_j \geq \frac{1}{\Delta}$

PROPOSITION 4.10. *NAIVE-KSEP-ROUND returns an explanation for $\mathcal{O}$ that is within a factor Δ from optimal, where Δ is the maximum number of possible partners associated with any observation.*

Algorithm 5 (GREEDY-KSEP-OPT1)

INPUT: Space $\mathcal{S}$, a set $\mathcal{O}$ of observations, a feasibility predicate feas, and real numbers $\alpha \geq 0$, $\beta > 0$
OUTPUT: Set $\mathcal{E} \subseteq \mathcal{S}$ that explains $\mathcal{O}$

(1) Run lines 1-5 of NAIVE-KSEP-SC
(2) Let OBS be an array, size $|\mathcal{O}|$ of lists to pointers in M. For some observation o, let OBS[o] be the corresponding list in the array.
(3) Run the loop in line 6 of NAIVE-KSEP-SC but when partner p of observation o is considered, add a pointer to $M[p]$ in the list OBS[o]. The list L need not be maintained.
(4) While numObs > 0 loop
  (a) Randomly select an element $o \in \mathcal{O}$ such that $\mathcal{O}'[o] =$ TRUE
  (b) Run the greedy-selection loop of line 7 of NAIVE-KSEP-SC, but consider the list OBS[o] instead of L
(5) Return $\mathcal{E}$

There are several things to note about this approach. First, it can be easily adapted to many of the weighted variants - such as **WT-SEP**. Second, we note that the rounding algorithm is not a randomized rounding algorithm – which often produces a solution that satisfies all of the constraints in the linear-integer program. The above algorithm guarantees that all of the observations will be covered (if an explanation exists). Finally, this approach allows us to leverage numerous software packages for solving linear and linear-integer programs.

5. GREEDY HEURISTICS FOR GAP PROBLEMS

5.1 A Linear Time Greedy Approximation Scheme

In this section, we introduce a greedy approximation scheme for the optimization version of k-**SEP** that has a lower time-complexity than NAIVE-KSEP-SC but still maintains the same approximation ratio. Our GREEDY-KSEP-OPT1 algorithm runs in linear time w.r.t. $\mathcal{O}$. The key intuition is that NAIVE-KSEP-SC iterates through $O(\Delta \cdot |\mathcal{O}|)$ possible partners in line 7. Our algorithm first randomly picks an observation and then greedily selects a partner for it. This results in the greedy step iterating through only $O(\Delta)$ partners.

EXAMPLE 5.1. *Consider the sun bear from Example 2.3 and Figure 2. After initializing the necessary data structures in lines 1-3, GREEDY-KSEP-OPT1 iterates through the observations in $\mathcal{O}$ where the associated position in $\mathcal{O}'$ is TRUE. Suppose the algorithm picks o_1 first. It now accesses the list pointed to from OBS[o_1]. This gives us a set of pointers to the following elements of $\mathcal{S}$: $\{p_1, p_2, p_3, p_4\}$. Following the greedy selection outlined in line 7 of NAIVE-KSEP-SC, the algorithm iterates through these points, visiting the list of observations associated with each one in the matrix array M.*

First, the algorithm accesses the list pointed to by $M[p_1]$. Figure 4 (left) shows the observations considered when p_1 is selected. As there is only one observation in list $M[p_1]$ whose associated Boolean in $\mathcal{O}'$ is TRUE, the variable cur_size is set to 1 (see line 7(c)iv of NAIVE-KSEP-SC). cur_ptr is then set to $M[p_1]$.

Now we consider the next element, p_2. Figure 4 (right) shows the list pointed to

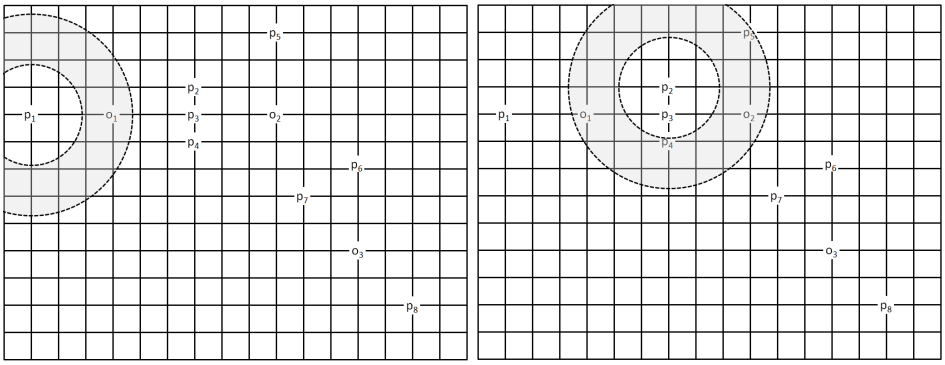

Fig. 4. **Left:** GREEDY-KSEP-OPT1 accesses the list pointed to by $M[p_1]$ thus considering all observations available to p_1. **Right:** GREEDY-KSEP-OPT1 accesses the list pointed to by $M[p_2]$ and finds it has more active observations than it found in the list pointed to by $M[p_1]$.

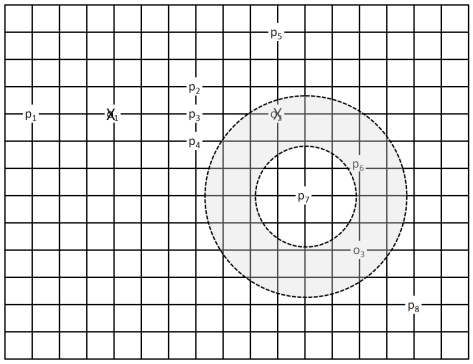

Fig. 5. GREEDY-KSEP-OPT1 considers the observations available to p_7. The X's on o_1 and o_2 signify that OBS$[o_1]$ and OBS$[o_2]$ are set to FALSE.

by $M[p_2]$. As $M[p_2]$ points to more observations whose associated $\mathcal{O}'$ Boolean is TRUE, we update cur_size to 2 and cur_ptr to $M[p_2]$.

The algorithm then iterates through p_3 and p_4, but finds they do not offer more observations than p_2. Hence, p_2 is added to the solution set ($\mathcal{E}$). The algorithm updates the array of Booleans, $\mathcal{O}'$ and sets $\mathcal{O}'[o_1]$ and $\mathcal{O}'[o_2]$ to FALSE (depicted by X's over those observations in subsequent figures). numObs is decremented by 2.

Now, we enter the second iteration of line 4. The only element for the algorithm to pick at this point is o_3, as only $\mathcal{O}'[o_3]$ is TRUE. The list OBS$[o_3]$ points to the positions $\{p_6, p_7, p_8\}$. In Figure 5 we look at what happens as the algorithm considers the p_7. As OBS$[o_2]$ = FALSE, it only considers o_3 when computing this_size.

When the algorithm finishes its consideration of all the elements pointed to by OBS$[o_3]$, it will return the first element of that set (p_6) as neither p_7 nor p_8 were partners to more available observations than p_6 (in our implementation of this algorithm, we use a coin-flip to break ties among partners with the same number of observations). GREEDY-KSEP-OPT1 then adds p_6 to $\mathcal{E}$ and terminates. The final

solution returned, $\{p_2, p_6\}$, is a valid (and in this case, optimal) explanation.

PROPOSITION 5.1 COMPLEXITY OF GREEDY-KSEP-OPT1. *GREEDY-KSEP-OPT1 has a complexity of $O(\Delta \cdot f \cdot |\mathcal{O}|)$ and an approximation ratio of $1 + \ln(f)$.*

PROPOSITION 5.2. *GREEDY-KSEP-OPT1 returns a $|\mathcal{E}|$-sized (α, β) explanation for $\mathcal{O}$.*
GREEDY-KSEP-OPT1 returns IMPOSSIBLE if there is no explanation for $\mathcal{O}$.

We can bound the approximation ratio for GREEDY-KSEP-OPT1 by $O(1+\ln(f))$, as it is still essentially a greedy algorithm for a covering problem. The main difference between GREEDY-KSEP-OPT1 is the way it greedily chooses covers (partners). This algorithm randomly picks an uncovered observation in each loop and then greedily chooses a cover that covers that observation. Improving the accuracy of this algorithm (in practice) is tied directly to the selection criteria used to pick observations, which is random in GREEDY-KSEP-OPT1. In Section 5.2 we develop an algorithm that "smartly" picks observations with a dynamic ranking scheme while maintaining a time complexity lower than the standard set-covering approach.

5.2 Greedy Observation Selection

GREEDY-KSEP-OPT1 randomly selects observations although subsequent partner selection was greedy. It is easy to implement an *a-priori* ranking of observations based on something like the maximum number of other observations which share a partner with it. Such a ranking could be implemented at the start of GREEDY-KSEP-OPT1 with no effect on complexity, but the ranking would be static and may lose its meaning after several iterations of the algorithm. We could also implement a dynamic ranking. We present a version of GREEDY-KSEP-OPT1 that we call GREEDY-KSEP-OPT2 that picks the observations based on dynamic ranking, runs in time $O(\Delta \cdot f^2 \cdot |\mathcal{O}| + |\mathcal{O}| \cdot \ln(|\mathcal{O}|))$, and maintains the usual approximation ratio of $1 + \ln(f)$ for greedy algorithms. Our key intuition was to use a Fibonacci heap [Fredman and Tarjan 1987]. With such a data structure, we can update the rankings of observations at constant amortized cost per observation being updated. The most expensive operation is to remove an observation from the heap - which costs an amortized $O(\ln(|\mathcal{O}|))$, however as we can never remove more than $|\mathcal{O}|$ items from the heap, this cost is most likely dominated by the cost of the rest of the algorithm, which is more expensive than GREEDY-KSEP-OPT1 by a factor of f. Recall that f is the bound on the number of observations supported by a single partner - and is often very small in practice.

In order to leverage the Fibonacci heap, there are some restrictions on how the ranking can be implemented. First, the heap puts an element with the minimal key on top, and can only decrease the key of elements - an element in the heap can never have its key increased. Additionally, there is a need for some auxiliary data structures as searching for an element in the heap is very expensive. Fortunately, the k-**SEP** problem is amenable to these type of data structures.

We based the key (ranking) on a simple heuristic for each observation. The key for a given observation o is the number of unique observations that share a partner with o. As we are extracting the minimum-keyed observation, we are taking the observation that has the "least in common" with the other observations. The

Algorithm 6 GREEDY-KSEP-OPT2

INPUT: Space $\mathcal{S}$, a set $\mathcal{O}$ of observations, a feasibility predicate feas, and real numbers $\alpha \geq 0$, $\beta > 0$
OUTPUT: Set $\mathcal{E} \subseteq \mathcal{S}$ that explains $\mathcal{O}$

(1) Run lines 1-3 of GREEDY-KSEP-OPT1.
(2) Let $key_1, \ldots key_{|\mathcal{O}|}$ be natural numbers associated with each observation. Initially, they are set to 0. For some $o \in \mathcal{O}$ let key_o be the associated number.
(3) Let REL_OBS be an array of lists of pointers to elements of $\mathcal{O}$. The size of the array is $\mathcal{O}$. For element $o \in \mathcal{O}$, let REL_OBS[o] be the corresponding space in the array.
(4) For each $o \in \mathcal{O}$, do the following:
    (a) For each element $p \in$ OBS[o], do the following.
        i. For each element *obs_ptr* of the list pointed to by $M[p]$, do the following
            A. If *obs_ptr* points to an element of $\mathcal{O}$ not pointed to in the list REL_OBS[o], then add *obs_ptr* to REL_OBS[o] and increment key_o by 1.
(5) Let OBS_HEAP be a Fibonacci heap. Let QUICK_LOOK be an array (size $\mathcal{O}$) of pointers to elements of the heap. For each $o \in \mathcal{O}$, add the tuple $\langle o, key_o \rangle$ to the heap, along with a pointer to the tuple to QUICK_LOOK[o]. Note we are using key_o as the key for each element in the heap.
(6) While OBS_HEAP is not empty, loop
    (a) Take the minimum element of OBS_HEAP, let o be the associated observation with this element.
    (b) Greedily select an element of OBS[o] as done in the loop at line 4 of GREEDY-KSEP-OPT1. We shall call this element p.
    (c) For every $o' \in \mathcal{O}$ pointed to by a pointer in $M[p]$, such that $\mathcal{O}'[o'] =$ TRUE, do the following.
        i. Set $\mathcal{O}'[o'] =$ FALSE
        ii. Remove the element pointed to by QUICK_LOOK[o'] from OBS_HEAP
        iii. For every element $o'' \in \mathcal{O}$ pointed to by an element of REL_OBS[o'] where $\mathcal{O}'[o''] =$ TRUE do the following.
            A. Decrease the $key_{o''}$ by 1.
(7) Return $\mathcal{E}$

intuition of choosing an observation with "less in common" with other observations ensures that outliers get covered with larger covers. Meanwhile, elements with a higher rank in this scheme are covered last, which may lead to a more efficient cover. In Section 6 we show experimentally that this heuristic was viable for the data-set we considered - providing more accurate results than the reduction from set-covering.

EXAMPLE 5.2. *The basic intuition behind GREEDY-KSEP-OPT2 is similar to GREEDY-KSEP-OPT1 in that it iterates through the observations and greedily chooses a partner. The main difference is that it ranks the observations instead of just randomly selecting them. Consider the sun bear from Example 2.3 whose behavior is depicted in Figure 2. In Example 5.1, we used GREEDY-KSEP-OPT1 to solve the associated k-**SEP** problem for this situation. We shall discuss how GREEDY-KSEP-OPT2 differs.*
The first main difference is that the algorithm assigns a rank to each observation

Observation	key_i	REL_OBS$[o_i]$
o_1	2	$\{o_1, o_2\}$
o_2	2	$\{o_1, o_2\}$
o_3	2	$\{o_2, o_3\}$

Table I. *key* values and related observations for observations in the sun bear scenario introduced in Example 2.3.

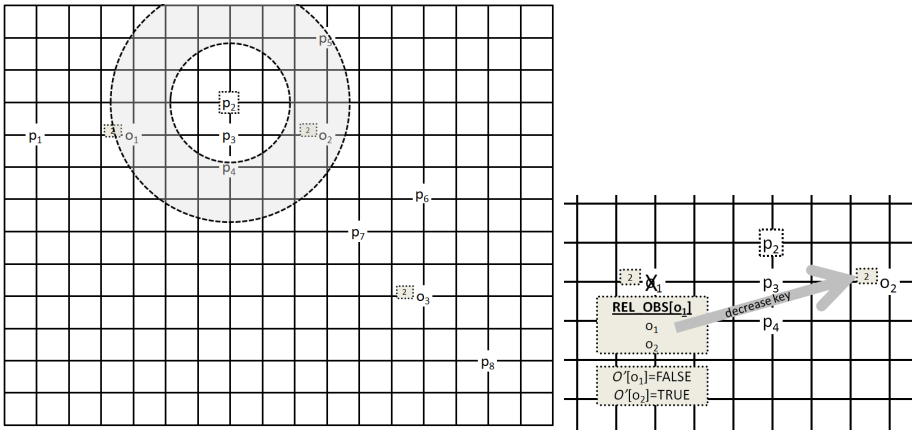

Fig. 6. **Left:** GREEDY-KESP-OPT2 considers all observations that can be partnered with p_2. Notice that in this figure by each observation we show a box that represents the key of the observation in the Fibonacci heap. **Right:** GREEDY-KSEP-OPT2 removes o_1 from the heap, and iterates through the elements in REL_OBS$[o_1]$, causing it to decrease the key of o_2.

o_i, called key_i, which is also the key used in the Fibonacci heap. This is done in the loop at line 4. It not only calculates key_i for each observation, but it also records the elements "related" to it in the array REL_OBS. Note that a "related" observation needs only to share a partner with a given observation. Not all related observations need to have the same partner. For the sun bear scenario, we show the keys and related observations in Table I.

As the key values are the same for all elements of $\mathcal{O}$, let's assume the algorithm first considers o_1 as in Example 5.1. As written, we would take the minimum element in the Fibonacci heap (a constant time operation). We would then consider the partners for o_1 which would result in the greedy selection of p_2, (just as in GREEDY-KSEP-OPT1 and NAIVE-KSEP-SC. Also notice we retain the array of Booleans, $\mathcal{O}'$ as well as the array of lists, M to help us with these operations.).

Now the issue arises that we must update the keys for the remaining observations, as well as remove observations covered by p_2. As we maintain REL_OBS and $\mathcal{O}'$, the procedure quickly iterates through the elements covered by p_2: o_1 and o_2. Figure 6 shows the status of the observations at this point.

We remove o_1 from the heap, and set $\mathcal{O}'[o_1]$ to FALSE. This prevents us from considering it in the future. We now iterate through each o'' in the list pointed to by REL_OBS$[o_1]$ where $\mathcal{O}'[o'']$ is TRUE and decrease the key of each by one. As per table I, REL_OBS$[o_1] = \{o_1, o_2\}$. As $\mathcal{O}'[o_1] =$ FALSE we do nothing. As $\mathcal{O}'[o_2] =$

TRUE, we decrease the key of the associated node in the Fibonacci heap. The array QUICK_LOOK ensures we can access that element in constant time. Figure 6 (left) graphically depicts this action.

Next, we consider the other element covered by partner p_2: o_2. After removing this element from the heap and setting $\mathcal{O}'[o_2]$ to FALSE, we can easily see that there does not exist any $o'' \in$ REL_OBS$[o_2]$ where $\mathcal{O}'[o''] =$ TRUE. Hence, we can proceed to pick a new minimum observation from the heap - which is o_3 in this case. The greedy selection proceeds (resulting in the choice of p_6), followed by the update procedure (which simply removes the node associated with o_3 from the heap and sets $\mathcal{O}'[o_3] =$ FALSE). As there are no more elements in the heap, GREEDY-KSEP-OPT2 returns the solution $\{p_2, p_6\}$.

THEOREM 5.1 COMPLEXITY OF GREEDY-KSEP-OPT2. *GREEDY-KSEP-OPT2 has a complexity of $O(\Delta \cdot f^2 \cdot |\mathcal{O}| + |\mathcal{O}| \cdot \ln(|\mathcal{O}|))$ and an approximation ratio of $1 + \ln(f)$.*

PROPOSITION 5.3. *GREEDY-KSEP-OPT2 returns a $|\mathcal{E}|$-sized (α, β) explanation for $\mathcal{O}$.*
GREEDY-KSEP-OPT2 returns IMPOSSIBLE if there is no explanation for $\mathcal{O}$.

6. IMPLEMENTATION AND EXPERIMENTS

In this section, we show that our geospatial abduction framework and algorithms are viable in solving real-world geospatial abduction problems. Using a real-world data set consisting of counter-insurgency information from Iraq, we were able to accurately locate insurgent weapons cache sites (partners) given previous attacks (observations) and some additional data (used for feas and α, β). This validates our primary research goal for the experiments - to show that geospatial abduction can be used to solve problems in the real-world.

We considered the naive set-covering approach along with GREEDY-KSEP-OPT1 and GREEDY-KSEP-OPT2, which according to our analytical results, had the best approximation ratios and time-complexities. We implemented these algorithms in 4000 lines of Java code, running on a Lenovo T400 ThinkPad laptop running Vista with an Intel Core 2 Duo T9400 2.53 GHz processor and 4.0 GB of RAM. Our SCARE (Social-Cultural Abductive Reasoning Engine) system [Shakarian et al. 2009] enabled us to carry out tests on real-world data. This data includes 21 months of Improvised Explosive Device or IED attacks in Baghdad[5] (a 25x27 km region) – these constitute our observations. It also included information on locations of caches associated with those attacks discovered by US forces. The locations of the caches constitute the (α, β) explanation we want to learn. We used data from the International Medical Corps to define feasibility predicates which took the following factors into account: (i) the ethnic makeup of neighborhoods in Baghdad - specifically, Sunni locations were deemed infeasible for cache locations, (ii) the locations of US bases in Baghdad were also considered infeasible and (iii) bodies of water were also deemed infeasible. We also separately ran tests on that part of the above data focused on Sadr City (a 7x7 km district in Baghdad) alone. On both

[5]Attack and cache location data was provided by the Institute for the Study of War

Algorithm 7 (FIND-BOUNDS)

INPUT: Historical, time-stamped observations $\mathcal{O}_h$, historical, time-stamped partners, $\mathcal{E}_h$, real number (distance threshold) β_{max}
OUTPUT: Real numbers α, β

(1) Set $\alpha = 0$ and $\beta = \beta_{max}$
(2) Set Boolean variable $flag$ to TRUE
(3) For each $o \in \mathcal{O}_h$, do the following:
  (a) For each $p \in \mathcal{E}_h$ that occurs after o, do the following.
    i. Let d be the Euclidean distance function.
    ii. If $flag$, and $d(o,p) \leq \beta_{max}$ then set $\alpha = d(o,p)$ and $\beta = d(o,p)$
    iii. If not $flag$, then do the following:
      A. If $d(o,p) < \alpha$ then set $\alpha = d(o,p)$
      B. If $d(o,p) > \beta$ and $d(o,p) \leq \beta_{max}$ then set $\beta = d(o,p)$
(4) Return reals α, β

Area	Algorithm	Sample Mean Solution Size	Sample Mean Number of Partners ≤ 0.5 km to actual cache
Baghdad	NAIVE-KSEP-SC	14.53	8.13
	GREEDY-KSEP-OPT1	15.02	7.89
	GREEDY-KSEP-OPT2	14.00	7.49
Sadr City	NAIVE-KSEP-SC	8.00	3.00
	GREEDY-KSEP-OPT1	6.61	4.44
	GREEDY-KSEP-OPT2	6.00	5.28

Table II. k-**SEP** Algorithm Results - Solution Size

these regions, we overlaid a grid whose cells were 100m x 100m each — about the size of a standard US city block. All timings were averaged over 100 runs.

We split the data into 2 parts — the first 7 months of data was used as a "training" set and the next 14 months of data was used for experimental evaluation. We used the following simple algorithm, FIND-BOUNDS, to determine the α, β values. We set β_{max} to 2.5 km. We leave more advanced procedures for learning these parameters to future work. Such parameters could also come from an expert.

Accuracy. Our primary goal in the experiments was to determine if the geospatial abduction framework and algorithms could provide viable results in a real-world setting. "Accuracy" in this section refers to two aspects - size of the solution, and the distance to the nearest actual cache site. The distance to nearest cache site was measured by taking the straight-line Euclidean distance to the nearest cache site that was found after the first attack supported by the projected cache site. We used the raw coordinate for the actual cache in the data set - not the position closest to the nearest point in the 100 m resolution grid that we overlaid on the areas. The accuracy results are summarized in Tables II-III.

Overall, GREEDY-KSEP-OPT2 consistently found the smallest solution - of cardinality 14 for Baghdad and 6 for Sadr City - on all 100 trials. For Baghdad, the other two algorithms both found a solution of size 14, but both averaged a higher

Area	Algorithm	Sample Mean Avg Dist to actual cache	Sample Std Dev of Avg Dist to actual cache	Sample Mean Std Dev of Dist to actual cache
Baghdad	NAIVE-KSEP-SC	0.79 km	0.02	0.64
	GREEDY-KSEP-OPT1	0.76 km	0.07	0.60
	GREEDY-KSEP-OPT2	0.72 km	0.03	0.63
Sadr City	NAIVE-KSEP-SC	0.72 km	0.03	0.46
	GREEDY-KSEP-OPT1	0.45 km	0.03	0.46
	GREEDY-KSEP-OPT2	0.35 km	0.03	0.47

Table III. k-**SEP** Algorithm Results - Distances to Actual Cache Sites

solution. For Sadr City, GREEDY-KSEP-OPT1 often did find a solution of 6 caches while NAIVE-KSEP-SC only found solutions of size 8. Additionally, in both tests, the solution sizes for GREEDY-KSEP-OPT1 varied more than the other two algorithms. Moreover, the HSD for both Baghdad and Sadr City indicated significant difference between all pairs of algorithms *wrt* solution size.

Of the partners in a given solution, we also recorded the number of partners less than 0.5 km away from an actual cache. For Baghdad, NAIVE-KSEP-SC performed best in this regard - averaging 8.13 partners less than 0.5 km from an actual cache site. Although this result for Baghdad is significant based on an analysis of variance (ANOVA) and honest significant differences (HSD) (p-value of $2.3 \cdot 10^{-9}$), we also note that the greatest difference among averages was still less than one partner. This same result for Sadr City, however, tells a different story. For this test, NAIVE-KSEP-SC performed poorly with regard to the other two algorithms - only finding 3 partners meeting these criteria for each of the 100 trials. GREEDY-KSEP-OPT2 performed very well in this aspect (for Sadr City). It averaged over 5 partners less than 0.5 km from an actual cache. Further, for Sadr City, *all* partners found by GREEDY-KSEP-OPT2 were within 600 m of an actual cache site. The ANOVA (p-value of $2.2 \cdot 10^{-16}$) and HSD of partners less than 0.5 km from an actual cache for the Sadr City trials indicate that these results are significant.

Our primary metric of accuracy was average distance to actual cache. In this regard, GREEDY-KSEP-OPT2 performed the best. It obtained an average distance of 0.72 km for Baghdad and 0.35 km for Sadr City. This number was 40 m less for Baghdad and 100 m less for Sadr City when compared to GREEDY-KSEP-OPT1, whose average distance varied widely among the trials. With regard to this metric, NAIVE-KSEP-SC performed the worst - particularly in Sadr City, where it predicted caches over twice as far from actual caches as GREEDY-KSEP-OPT2 (on average). For both Baghdad and Sadr City, the simple ANOVA yielded a p-value of $2.2 \cdot 10^{-16}$, which suggests with a 99% probability that there is a difference among the algorithms. Also, for both areas, Tukey's HSD indicates significant difference between each pair-wise comparison of algorithms.

Algorithm run times. Table IV shows the run-times of our algorithms. In order to validate the findings suggested by Table IV statistically, we ran analysis of variance (ANOVA) and Tukey's Honest Significant Difference test (HSD) for pair-wise comparisons [Freedman et al. 2007]. An ANOVA for the Baghdad run-times gave a p-value of $2.2 \cdot 10^{-16}$, which suggests with well over 99% probability that

Area	Algorithm	Sample Mean Run-Time	Sample Run-Time Standard Deviation
Baghdad	NAIVE-KSEP-SC	354.75 ms	12.86
	GREEDY-KSEP-OPT1	162.08 ms	40.83
	GREEDY-KSEP-OPT2	201.40 ms	36.44
Sadr City	NAIVE-KSEP-SC	28.85 ms	10.52
	GREEDY-KSEP-OPT1	25.44 ms	9.33
	GREEDY-KSEP-OPT2	24.64 ms	8.95

Table IV. k-**SEP** Algorithm Performance Results

GREEDY-KSEP-OPT1 is statistically faster than GREEDY-KSEP-OPT2. The HSD for Baghdad indicates that, with regard to run-times, all pair-wise-comparison of the three algorithms are significantly different. For Sadr City, the ANOVA gave a p-value of $4.9 \cdot 10^{-3}$, which suggests with a 99% probability that the algorithms differ in run-times. However, the HSD indicates, with an 82% probability, that there is no difference among GREEDY-KSEP-OPT1 and GREEDY-KSEP-OPT2, while both differ significantly from NAIVE-KSEP-SC.

6.1 A Simple Heuristic to Improve Accuracy

In our implementation of all three algorithms, "ties" in greedy selection of partners were determined by a "coin toss." Specifically, we are considering the case where $this\_size = cur\_size$ in line 7(c)iv of NAIVE-KSEP-SC in Section 4.2. Let us rephrase the situation as follows. Let $\mathcal{O}$ be the entire set of observations and $\mathcal{O}' \subseteq \mathcal{O}$ be the set of observations currently not assigned a partner. Let p be the current partner that best meets the criteria for greedy selection and p' be the partner we are considering. We define P and P' as subsets of $\mathcal{O}$ that are the observations associated with p and p' respectively. Hence, if $|P' \cap \mathcal{O}'| > |P \cap \mathcal{O}'|$, we pick p'. As implemented, if $|P' \cap \mathcal{O}'| = |P \cap \mathcal{O}'|$, we flip a coin. We add a simple heuristic that simply states that "partners that cover more observations are preferred." We change the criteria as follows:

—If $|P' \cap \mathcal{O}'| = |P \cap \mathcal{O}'|$, then do the following:
  —If $|P'| > |P|$, pick p'
  —If $|P| > |P'|$, pick p
  —If $|P| = |P'|$, flip a coin

We shall refer to this as the "tie-breaker" heuristic. The result is that the solution set of partners covers more observations and hence provides a more "dense" solution.

We added this heuristic to our existing code for all three algorithms and ran each one 100 times for both the Baghdad and Sadr City areas. Unsurprisingly, as this is a constant-time operation, run-times were not affected. However, accuracy improved in all cases. As GREEDY-KSEP-OPT2 still provided the most accurate results, the following exposition shall focus on how the heuristics affected the solution size and accuracy for this algorithm.

Because the tie-breaker heuristic only adjusts how two partners are chosen - both of which can be paired with the same uncovered observations - the size of the solution was unaffected in both the Baghdad and Sadr City trials. However, the number of predicted cache sites less than 500 m from an actual site increased

Area	Tie-Breaker Heuristic	Sample Mean Solution Size	Sample Mean Number of Partners ≤ 0.5 km to actual cache
Baghdad	No	14.00	7.49
	Yes	14.00	7.87
Sadr City	No	6.00	5.28
	Yes	6.00	6.00

Table V. The Tie-Breaker heuristic on GREEDY-KSEP-OPT2 - Solution Size

Area	Tie-Breaker Heuristic	Sample Mean Avg Dist to actual cache	Sample Std Dev of Avg Dist to actual cache	Sample Mean Std Dev of Dist to actual cache
Baghdad	No	0.72 km	0.03	0.63
	Yes	0.69 km	0.02	0.64
Sadr City	No	0.35 km	0.03	0.47
	Yes	0.28 km	0.02	0.11

Table VI. The Tie-Breaker heuristic on GREEDY-KSEP-OPT2 - Distances to Actual Cache Sites

for both the Baghdad and Sadr City tests. For Baghdad, more trials returned solutions with 8 predictions less than 500 m from an actual site than returned 7 - the opposite being the case without the tie-breaker heuristic. For Sadr City, all elements of every solution set returned was less than 500 m from an actual cache site. Using the well known T-Test [Freedman et al. 2007], we showed that these results are statistically significant as this test returned a p-value of $6.2 \cdot 10^{-8}$ for Baghdad and $2.2 \cdot 10^{-16}$ for Sadr City.

Summary. The above experiments demonstrate statistically that GREEDY-KSEP-OPT2 provides a viable solution - consistently producing the smaller solution sets which were closer to actual cache sites faster than the basic set-covering approach, at times approaching the faster, although less-accurate GREEDY-KSEP-OPT1. The proximity of the elements of the solution set to actual cache sites is encouraging for real-world use. The results are strong enough that two US Army units used SCARE to aide in locating IED caches.

7. RELATED WORK

In this section we present related work of three different varieties. We compare GAPs to other forms of abduction, facility location, *k-means* clustering, and constrained clustering. As an aside, readers interested in a discussion of the SCARE software from the perspective of military analysis or social science can refer to [Shakarian et al. 2009] where the software package was introduced. However, that work does not include any formal technical details on the framework of geospatial abduction, complexity results, or algorithm analysis.

GAPs and other forms of Abduction. Abduction [Peirce 1955] has been extensively studied in medicine [Reggia and Peng 1990; Peng and Reggia 1986], fault diagnosis [Console et al.], belief revision [Pagnucco 1996], database updates [Kakas and Mancarella 1990; Console et al. 1995] and AI planning [do Lago Pereira and

de Barros 2004]. Two major existing theories of abduction include logic-based abduction [Eiter and Gottlob 1995] and set-covering abduction [Bylander et al. 1991]. Though none of the above papers deals with spatial inference, Shanahan [1996] presents a logical formalism dealing with objects' spatial occupancy, while Santos and Shanahan [2002] describe the construction of a qualitative spatial reasoning system based on sensor data from a mobile robot. In Santos and Shanahan [2002], sensor data are explained by hypothesizing the existence of physical objects along with the dynamic relationships that hold between them, all with respect to a (possibly moving) viewpoint. This approach combines both space and time. Kuipers [1996] describes the *Spatial Semantic Hierarchy* which formalizes, the spatial context in which a robot moves. In the hierarchy, the topological level defines a map which describes the environment as a collection of places, paths, and regions, linked by topological relations such as connectivity, order, containment, boundary, and abstraction. Places (i.e., zero-dimensional points), paths (i.e., one dimensional subspaces, denoting for example a street in a city, possibly represented as an ordering relation on the places they contain), and boundary regions (i.e., two-dimensional subspaces of the robot environment) are created from experience represented as a sequence of views and actions. They are created by abduction, positing the minimal additional set of places, paths, and regions required to explain the sequence of observed views and actions.

Set-covering abduction [Bylander et al. 1991] assumes the existence of a function determining the observable effects of a set of hypotheses, and is based on inverting such function. Given a set of hypotheses H and a set of observations O, the domain knowledge is represented by a function e that takes as an argument a set of hypotheses and gives as a result the corresponding set of observations. Thus, for every subset of the hypotheses $H' \subseteq H$, their effects are known to be $e(H')$. In this case, abduction finds a set $H' \subseteq H$ such that $O \subseteq e(H')$, that is, it finds a set of hypotheses H' whose effects $e(H')$ include all observations in O. A common assumption is that the effects of the hypotheses are independent, that is, for every $H' \subseteq H$, it holds that $e(H') = \bigcup_{h \in H'} e(\{h\})$. If this condition is met, abduction can be seen as a form of set-covering. No spatial reasoning is done here.

Comparison with facility location. There are several important ways in which GAPs differ from facility location problems.

—Although it is possible to specify a distance-based cost function, in a GAP problem, the distances between observations and partners are constraints (α and β in this paper) whereas facility location problems usually attempt to minimize the distance between producers and consumers.

—In this paper, GAP problems have a minimum distance between observations and partners that must be exceeded. In many respects, this requirement makes GAP problems more difficult than facility location and other computational geometry problems as the set of possible partners that cover a given observation is a non-convex ring. Further, the feasibility function (feas) adds non-uniform holes to such a ring. [Maass 1986] addresses the complexity of non-convex covering and highlights issues with problems such as this.

—The feasibility predicate, feas is not part of a facility location problem. This gives

us the ability to restrict certain locations that can be partners.

—In general, the relation between observations and partners can be viewed to be a set of constraints. In this paper, we only used α, β,and feas. However, in the future, we could add additional constraints. Further, as our formalism represents space as a set of discrete points (also not typically done with facility location), we can easily specify certain properties of these points to apply such constraints (such as feas).

Comparsion with *k*-means clustering. A well-known and studied problem in clustering location is the *k-means* problem [MacQueen 1967]. This problem can be expressed as follows:

k-means:
INPUT: Coordinates on a plane C and natural number k
OUTPUT: k disjoint sets of C, $C'_1, \ldots, C'_k$ such that for each C_i, all the mean Euclidean distance among all $c \in C_i$ is minimized.

Clustering problems group points into clusters, associating each cluster with a center. At first glance, one may think that the points are equivalent to observations and the "centers are equivalent to partners. However, this is not so. Most versions of the clustering problem seek only to arrange points in groups – with "centers" being a side-effect of the algorithm. *Geospatial abduction problem seeks to find partners that support observations and places constraints on the location of the partners - this is a key difference from "centers" in clustering problems.* Clustering algorithms cannot handle the generality of our feasibility predicate or the (α, β) constraints.

In addition to these obvious differences, we experimentally compared an implementation of *k-means* with GREEDY-KSEP-OPT2 on the Sadr City data. Even when we ignore the obvious value of α, β and the feasibility predicate, GREEDY-KSEP-OPT2 outperforms the SimpleKMeans solver in WEKA version 3.7.0 [WEKA 2009]. Note that the exclusion of these parameters makes GREEDY-KSEP-OPT2 perform worse than it performs with these parameters – yet, it performed better than k-means in terms of accuracy. Our experiment was set-up as follows:

—We used the same area for the Sadr City tests, as the α value was 0 in these tests and there were virtually no non-feasible points near the observations. This allowed us to use WEKA's k-means implementation "out-of-the-box" as we did not have to implement any extra infrastructure to deal with feasibility and $\alpha = 0$.

—We set $k = 6$, the number of partners consistently found by GREEDY-KSEP-OPT2. Normally, we would rather have the algorithm determine this size. Note that supplying the algorithm with a size already determined by GREEDY-KSEP-OPT2 (and, also the smallest size of any explanation for Sadr City we found in our trials) gives an advantage to *k-means*. Hence, we did not compare solution sizes.

—We clustered the observations with *k-means* and considered the "center" of each cluster the cache location for the cluster.

—We did not compare timing results, as we ran WEKA in its GUI environment.

We ran 500 iterations of the SimpleKMeans and worked with the average centers for the clusters as reported by WEKA. Multiple runs of the 500 iterations yielded

the same centers.

Average Distance Using WEKA, we obtained an average accuracy of 0.38 km, which is worse than GREEDY-KSEP-OPT2 (average over 100 trials, 0.28 km).

Worst-Case Distance WEKA's SimpleKMeans returned 2 of the 6 points with a distance of greater than 600 meters from a cache site. Without the "tie-breaking" heuristic, GREEDY-KSEP-OPT2 never reported a prediction over 600 meters from a cache site (all reported partners over 100 trials). With the heuristic, GREEDY-KSEP-OPT2 never reported a prediction over 500 meters from a cache site.

Best-Case Distance The closest partners ever returned by GREEDY-KSEP-OPT2 (either with our without the heuristic) were 200 m away from an actual cache site (on average, the closest partner per explanation was 220 m away). WEKA's SimpleKMeans did return two partners less than 200 m - each only 100 m away from an actual cache site.

These results suggest that *k-means* may not be the optimal method for GAP problems. Further, it does not support feasibility and α. The results do hold some promise for some variants of cost-based spatial explanation problems that require a k input from one of our greedy-approaches. However, even in this case, there would be modification required of the *k-means* algorithm to support feasibility and α.

Comparison with Constrained clustering. *Constrained clustering* [Wagstaff et al. 2001] studies clustering where, in addition to the points to be clustered, there are constraints that either force two points in the same cluster (must-link) or force two points to be in different clusters (cannot-link). Later work on constrained clustering has focused on distance constraints between elements of C or distance constraints between clusters [Davidson and Ravi 2005]. Much of the work in this area is summarized in [Basu et al. 2008].

At first glance, it may appear that spatial abduction can be expressed as a cannot-link constrained clustering problem as follows: For each $o, o' \in \mathcal{O}$ if $\nexists p \in \mathcal{S}$ s.t. $d(o,p) \in [\alpha, \beta]$, $d(o',p) \in [\alpha, \beta]$, and $\mathsf{feas}(p)$, then create a cannot-link constraint for o, o'.

However, such a mapping cannot be guaranteed to provide a correct result. For example, take o_1, o_2, o_3 and p_{12}, p_{23}, p_{13}. Suppose o_1 and o_2 share just partner p_{12}, o_2 and o_3 share just partner p_{23} and o_1, o_3 share just partner p_{13}. This is entirely possible given the generality of feas. In such a case, all three observations could be incorrectly grouped into a single cluster - although it is obvious there is no single partner that supports all of them. Hence, such a mapping would not be trivial. Further, most clustering algorithms are not seeking to constructively find centers that are constrained. We leave the study of constrained clustering to solve GAP problems (i.e. an adaption of the k-means algorithm) to future work. However, it is also worth noting that solving constrained clustering problems given cannot-link constraints is NP-complete, so the application of clustering techniques to this problem does not imply a more tractable version of geospatial abduction, but rather

an alternative heuristic.

8. CONCLUSIONS

There are a wide variety of problems where we can make geo-located observations "on the ground" and where we want to infer a partner location. In this paper, we have presented four examples of such problems — one dealing with serial killers, another dealing with wildlife studies, and a third (perhaps more fun) application dealing with finding sunken ships. A fourth real world application we have looked at is that of finding weapons caches associated with Improvised Explosive Device (IED) attacks in Iraq where we were able to use real world, open source data. It is clear that many other applications exist as well. For example, a bizarre (but real world) combination of two of our examples involves frequent attacks by man-eating leopards on children in certain parts of greater Bombay in India. This situation is analogous to the serial killer scenario where the leopard is the serial killer. We want to find the leopard's favorite "hang outs", capture it, and solve the problem.

In this paper, we have made an attempt to formally define a class of *geospatial abduction problems* (GAPs for short). We specifically made the following contributions.

—We developed formal mathematical definitions of geospatial abduction problems, including several variants of the above problems. We conducted a detailed analysis of the complexity of these problems.

—We developed exact algorithms for many of the problems, including a straightforward enumeration approach (NAIVE-KSEP-EXACT), by showing and leveraging reductions to both the set-covering and dominating set problems, and by articulating these geospatial abduction problems via integer linear programs.

—As the complexity of most of the problems we have studied is NP-hard, we developed two greedy approximation schemes for the k-**SEP** problem (other than set-covering) and illustrated a scheme to quickly find a solution using randomized approaches to the dominating set problem.

—We have implemented these algorithms and conducted experimental comparisons of the reduction to set-covering and two other greedy approaches - GREEDY-KSEP-OPT1 and GREEDY-KSEP-OPT2. Both of these algorithms outperformed the set-covering reduction in an experiment on the *Understanding War Special Groups* data set. We also implemented a "tie-breaker" heuristic that further improved the accuracy of the algorithms.

—We have also developed approximation schemes using relaxations of the linear-integer program for k-**SEP** and the cost-based variant **WT-SEP**.

There are many interesting directions for future work. For example, spatial abduction in dimensions greater than two might be explored. A probabilistic variant might replace the feasibility predicate with a probability distribution function, or express the relationship between observations and partners as a PDF based on distance rather than rely on α, β. Also, the use of randomization in the approximation algorithms may improve results for both the greedy and linear programming approaches presented in this paper.

One aspect to explore in future work is the relationship between observations and partners. k-**SEP** and its cost based variants only rely on α, β. However, many applications may have other constraints. Perhaps there is a direction associated with each observation (as in identifying where an artillery round originated from), which would limit the locations of the partner. Another possibility is to add geographic constraints. Perhaps the observation cannot have a partner across a body of water, or beyond the edge of a cliff.

ACKNOWLEDGMENT

Some of the authors of this paper were funded in part by AFOSR grant FA95500610405 and ARO grants W911NF0910206 and W911NF0910525.

REFERENCES

Alpaydin, E. 2010. *Introduction to Machine Learning*, 2 ed. MIT Press.

Basu, S., Davidson, I., and Wagstaff, K. 2008. *Constrained Clustering: Advances in Algorithms, Theory, and Applications*. Chapman & Hall/CRC.

Brantingham, P. and Brantingham, P. 2008. Crime Pattern Theory. In *Enviromental Criminology and Crime Analysis*, R. Wortley and L. Mazerolle, Eds. 78–93.

Bylander, T., Allemang, D., Tanner, M. C., and Josephson, J. R. 1991. The computational complexity of abduction. *Artificial Intelligence 49,* 1-3, 25–60.

Console, L., Portinale, L., and Theseider Dupré, D. Focussing Abductive Diagnosis. *Artificial Intelligence Communications, year=1991, volume = 4, number = 2-3, pages = 88-97*.

Console, L., Sapino, M. L., and Theseider Dupré, D. 1995. The Role of Abduction in Database View Updating. *J. Intelligent Information Systems 4,* 3, 261.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. 2001. *Introduction to Algorithms*, Second ed. MIT Press.

Davidson, I. and Ravi, S. S. 2005. Clustering with constraints: Feasibility issues and the k-means algorithm. In *SIAM Data Mining Conference (SDM)*.

do Lago Pereira, S. and de Barros, L. N. 2004. Planning with abduction: A logical framework to explore extensions to classical planning. In *Lecture Notes in Computer Science Advances in Artificial Intelligence*.

Eiter, T. and Gottlob, G. 1995. The complexity of logic-based abduction. *Journal of the ACM 42,* 1, 3–42.

Fredman, M. L. and Tarjan, R. E. 1987. Fibonacci heaps and their uses in improved network optimization algorithms. *Journal ACM 34,* 3 (July), 596–615.

Freedman, D., Purves, R., and Pisani, R. 2007. *Statistics*, 4 ed. W.W. Norton and Co.

Garey, M. R. and Johnson, D. S. 1979. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA.

Hochbaum, D. S. 1982. Approximation Algorithms for the Set Covering and Vertex Cover Problems. *SIAM Journal on Computing 11,* 3, 555–556.

Jia, L., Rajaraman, R., and Suel, T. 2002. An efficient distributed algorithm for constructing small dominating sets. *Distributed Computing 15,* 4, 193–205.

Johnson, D. 1982. The NP-Completeness Column: An Ongoing Guide. *J. Algorithms 3,* 2, 182–195.

Kakas, A. C. and Mancarella, P. 1990. Database updates through abduction. In *International Conference on Very Large Databases (VLDB)*.

Karp, R. 1972. Reducibility Among Combinatorial Problems. In *Complexity of Computer Computations*, R. E. Miller and J. W. Thatcher, Eds. 85103.

Kuhn, F. and Wattenhofer, R. 2003. Constant-time distributed dominating set approximation. In *Proc. of the 22 nd ACM Symposium on the Principles of Distributed Computing (PODC)*. 25–32.

Kuipers, B. 1996. A hierarchy of qualitative representations for space. In *Working papers of the Tenth International Workshop on Qualitative Reasoning about Physical Systems*.

Lu, J. J., Nerode, A., and Subrahmanian, V. 1996. Hybrid knowledge bases. *IEEE Transactions on Knowledge and Data Engineering 8,* 5, 773–785.

Lund, C. and Yannakakis, M. 1994. On the hardness of approximating minimization problems. *Journal of the ACM 41,* 5, 960–981.

Maass, W. 1986. On the complexity of nonconvex covering. *SIAM Journal on Computing 15,* 2, 453–467.

MacQueen, J. B. 1967. Some methods for classification and analysis of multivariate observations. In *Proc. of the fifth Berkeley Symposium on Mathematical Statistics and Probability*, L. M. L. Cam and J. Neyman, Eds. Vol. 1. University of California Press, 281–297.

Pagnucco, M. 1996. The role of abductive reasoning within the process of belief revision. Ph.D. thesis, Basser Department of Computer Science, University of Sydney, Australia.

Papadimitriou, C. H. 1981. Worst-Case and Probabilistic Analysis of a Geometric Location Problem. *SIAM Journal on Computing 10,* 3, 542–557.

Paschos, V. T. 1997. A survey of approximately optimal solutions to some covering and packing problems. *ACM Computing Surveys 29,* 2, 171–209.

Peirce, C. S. 1955. *Philosophical writings of Peirce, selected and edited with an introd. by Justus Buchler*. Dover Publications New York,.

Peng, Y. and Reggia, J. A. 1986. Plausibility of Diagnostic Hypotheses. In *Proceedings, 5th National Conference on AI (AAAI)*. 140–145.

Reggia, J. A. and Peng, Y. 1990. *Abductive inference models for diagnostic problem-solving*. Springer-Verlag New York, Inc., New York, NY, USA.

Rossmo, D. K. and Rombouts, S. 2008. Geographic Profiling. In *Enviromental Criminology and Crime Analysis*, R. Wortley and L. Mazerolle, Eds. 136–149.

Santos, P. and Shanahan, M. 2002. Hypothesising object relations from image transitions. In *Proc. European Conference on Artificial Intelligence (ECAI)*.

Shakarian, P., Subrahmanian, V., and Sapino, M. L. 2009. SCARE: A Case Study with Baghdad. In *Proceedings of the Third International Conference on Computational Cultural Dynamics (ICCCD)*. AAAI.

Shanahan, M. 1996. Noise and the common sense informatic situation. In *Proc. AAAI*. 1098.

US Army. 1994. *Intelligence Preparation of the Battlefiled (US Army Field Manual)*, FM 34-130 ed.

Vazirani, V. V. 2004. *Approximation Algorithms*. Springer.

Wagstaff, K., Cardie, C., Rogers, S., and Schrödl, S. 2001. Constrained k-means clustering with background knowledge. In *Proceedings of the Eighteenth International Conference on Machine Learning (ICML)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 577–584.

WEKA. 2009. WEKA 3 Data Mining, http://www.cs.waikato.ac.nz/ml/weka/.

Algorithm 8 (GCD-TO-KSEP)

INPUT: Instance of GCD $\langle S, P, b, k\rangle$
OUTPUT: Instance of k-**SEP** $\langle \mathcal{S}, \mathcal{O}, \mathsf{feas}, \alpha, \beta, k'\rangle$

(1) Set $\mathcal{S}$ to be a set of lattice points in the Euclidean plane that include all points in P
(2) Set $\mathcal{O} = P$
(3) Let $feas(x) =$ TRUE iff $x \in P$
(4) Set $\alpha = 0$
(5) Set $\beta = b/2$
(6) Set $k' = k$

A. PROOFS

A.1 Proof of Theorem 3.1

k-**SEP** is NP-Complete.

PROOF. **Geometric Covering by Discs.** (GCD)
INPUT: A set P of integer-coordinate points in a Euclidean plane, positive integers $b > 0$ and $k < |P|$.
OUTPUT: "Yes" if there exist k discs of diameter b centered on points in P such that there is a disc covering each point in P — "no" otherwise.
CLAIM 1: k-**SEP** is in the complexity class NP.
Suppose a non-deterministic algorithm can guess a set $\mathcal{E}$ that is a k-sized simple (α, β) explanation for $\mathcal{O}$. We can check the feasibility of every element in $\mathcal{E}$ in $O(|\mathcal{E}|)$ time and compare every element of $\mathcal{E}$ to every element of $\mathcal{O}$ in $O(|\mathcal{O}|^2)$ time. Hence, k-**SEP** is in the complexity class NP as we can check the solution in polynomial time.

CLAIM 2: k-**SEP** is NP-Hard.
We use the polynomial algorithm GCD-TO-KSEP to take an instance of GCD and create an instance of k-**SEP**.

CLAIM 2.1: If there is a k'-sized simple (α, β) explanation for $\mathcal{O}$, then there are k discs, each centered on a point in P of diameter b that cover all points in P.
Let $\mathcal{E}$ be the k'-sized simple (α, β) explanation for $\mathcal{O}$. Suppose by way of contradiction, that there are not k discs, each centered on a point in P of diameter b that cover all points in P. As $k' = k$, and all elements of $\mathcal{E}$ must be in P by the definition of feas, let us consider the k discs of diameter b centered on each element of $\mathcal{E}$. So, for these discs to not cover all elements of P, there must exist an element of P, that is not covered by a disc. As $P \equiv \mathcal{O}$, then there must exist an element of $\mathcal{O}$ outside of one of the discs. Note that all elements of $\mathcal{O}$ are within a distance β of an element of $\mathcal{E}$ by the definition of a k'-sized simple (α, β) explanation (as $\alpha = 0$). As $\beta = b/2$, each element of $\mathcal{O}$ falls inside a disc of diameter b centered on an element of $\mathcal{E}$, thus falling within a disc and we have a contradiction.

CLAIM 2.2: If there are k discs, each centered on a point in P of diameter b that cover all points in P then there is a k'-sized simple (α, β) explanation for $\mathcal{O}$.
Let set E be the set of points that are centers of the k discs. We note that $E \subseteq P$. Assume by way of contradiction, that there is no k'-sized simple (α, β) explanation for $\mathcal{O}$. Let us consider if E is a k'-sized simple (α, β) explanation for $\mathcal{O}$. As $k = k'$, $\alpha = 0$, and all points of E are feasible, there must be some $o \in \mathcal{O}$ such that $\forall e \in E$, $d(e, o) > \beta$. As $\mathcal{O} \equiv P$, we know that all points in $\mathcal{O}$ fall in a disc centered on a point in E, hence each $o \in \mathcal{O}$ must be a distance of $b/2$ or less from a point in E. As $\beta = b/2$, we have a contradiction. □

A.2 Proof of Corollary 3.1

Cost-based Explanation is NP-Complete.

PROOF. CLAIM 1: Cost-based Explanation is in the complexity class NP.
This follows directly from Theorem 3.1, instead of checking the size of $\mathcal{E}$, we only need to apply the function χ to the $\mathcal{E}$ produced by the non-deterministic algorithm to ensure that $\chi(\mathcal{E}) \leq v$.

CLAIM 2: Cost-based Explanation is NP-Hard.
We show k-**SEP**$\leq_p$ CBE. Given an instance of k-**SEP**, we transform it into an instance of CBE in polynomial time where $\chi(\mathcal{E}) = |\mathcal{E}|$ and $v = k$.

CLAIM 2.1: If there is a set $\mathcal{E}$ such that $\chi(\mathcal{E}) \leq v$ then $|\mathcal{E}| \leq k$.
Straightforward.

CLAIM 2.2: If there is a set $\mathcal{E}$ of size k or less then $\chi(\mathcal{E}) \leq v$
Straightforward. □

A.3 Proof of Corollary 3.2

WT-SEP is NP-Complete.

PROOF. Membership in the complexity class NP follows directly from Theorem 3.1, instead of checking the size of $\mathcal{E}$, we check if $\sum_{p \in \mathcal{E}} c(p) \leq v$. We also note that the construction for cost-based explanation in Theorem 3.1 is also an instance of WT-SEP, hence NP-hardness follows immediately.

□

A.4 Proof of Theroem 3.2

TD-SEP is NP-Complete.

PROOF. CLAIM 1: TD-SEP is in the complexity class NP.
Given a set $\mathcal{E}$, we can easily determine in polynomial time that it meets the standards of the output specified in the problem statement.

CLAIM 2: TD-SEP is NP-hard.
Consider Euclidean k-Median Problem, as presented and shown to be NP-Complete in Papadimitriou [1981], defined as follows:
INPUT: A set P of integer-coordinate points in a Euclidean plane, positive integer

$k' < |P|$, real number $v' > 0$.
OUTPUT: "Yes" if there is a set of points, $S \subseteq P$ such that $|S| = k'$ and $\sum_{x_i \in X} \min_{s_j \in S} d(x_i, s_j) \leq v'$ — "no" otherwise.

Given an instance of the Euclidean k-Median Problem, we create an instance of TD-SEP as follows:

—Set $\mathcal{S}$ to be a set of lattice points in the Euclidean plane that include all points in P
—Set $\mathcal{O} = P$
—Let $feas(x) =$ TRUE iff $x \in P$
—Set $\alpha = 0$
—Set β greater than the diagonal of $\mathcal{S}'$
—Set $k = k'$
—Set $v = v'$

CLAIM 2.1: If there is $\mathcal{E}$, a k-sized explanation for $\mathcal{O}$ such that $\sum_{o_i \in \mathcal{O}} \min_{p_j \in \mathcal{E}} d(o_i, p_j) \leq v$, then there is a set $S \subseteq P$ such that $|S| = k'$ and $\sum_{x_i \in P} \min_{s_j \in S} d(x_i, s_j) \leq v'$.
Because of how we set feas and $\mathcal{O}$, $\mathcal{E} \subseteq P$. As α and β do not affect $\mathcal{E}$, the only real restrictions on $\mathcal{E}$ is that its cardinality is k and that $\sum_{o_i \in \mathcal{O}} \min_{p_j \in \mathcal{E}} d(o_i, p_j) \leq v$. Because of how we set k and v, we can see that $\mathcal{E}$ meets all the conditions to be a solution to the Euclidean k-Median problem, hence the claim follows.

CLAIM 2.2: If there is set $S \subseteq P$ such that $|S| = k'$ and $\sum_{x_i \in P} \min_{s_j \in S} d(x_i, s_j) \leq v'$, then there is set $\mathcal{E}$, a k-sized explanation for $\mathcal{O}$ such that $\sum_{o_i \in \mathcal{O}} \min_{p_j \in \mathcal{E}} d(o_i, p_j) \leq v$.
In the construction, the arguments α, β and feas allow any element of a solution to the k-Median problem to be a partner for any observation in $\mathcal{O}$. By how we set k and v, we can easily see that S is a valid solution to TD-SEP. The claim follows.

The statement of the theorem follows directly from claims 1-2. □

A.5 Proof of Proposition 4.1

If there is a k-sized simple (α, β) explanation for $\mathcal{O}$, then NAIVE-KSEP-EXACT returns an explanation. Otherwise, it returns NO.

Proof. CLAIM 1: If there is a k-sized simple (α, β) explanation for $\mathcal{O}$, then NAIVE-KSEP-EXACT returns an explantion.
Suppose, by way of contradiction, that there is a k-sized simple (α, β) explanation for $\mathcal{O}$ and NAIVE-KSEP-EXACT returns NO. Then there does not exist k bit strings such that for all o_i, $\sum_{j=1}^{k} (\ell_j(i)) \geq 1$. As each bit string is associated with a point in $\mathcal{S}$, then by the construction of the bit strings, there are not k points in $\mathcal{S}$ such that each point is feasible and falls no closer than α and no further than β distance away from each point in $\mathcal{O}$. This is a contradiction.

CLAIM 2: If there is no k-sized simple (α, β) explanation for $\mathcal{O}$, then NAIVE-KSEP-EXACT returns NO.

Suppose, by way of contradiction, that there is no k-sized simple (α, β) explanation for $\mathcal{O}$ and NAIVE-KSEP-EXACT returns an explanation. Then there must exist k bit strings such that $\bigvee_{j=1}^{k}(\ell_j(i)) = 1$. As each bit string is associated with a point in $\mathcal{S}$, then by the construction of the bit strings, there must exist k points in $\mathcal{S}$ such that each point is feasible and falls no closer than α and no further than β distance away from each point in $\mathcal{O}$. This is a contradiction.

□

A.6 Proof of Proposition 4.2

The complexity of NAIVE-KSEP-EXACT is $O(\frac{1}{(k-1)!}(\pi(\beta^2 - \alpha^2)|\mathcal{O}|)^{(k+1)})$.

PROOF. Note that as all pointers in M are initially null, thus there is no need to iterate through every element in M - rather lists in M can only be initialized as needed. Hence, the cost to set-up M in $O(1)$ and not the size of the matrix.
As each $o \in \mathcal{O}$ has, at most $\pi(\beta^2 - \alpha^2)$ partners, the total complexity of the inner loop is $\pi(\beta^2 - \alpha^2)|\mathcal{O}|$.
As we have, at most, $\pi(\beta^2 - \alpha^2)|\mathcal{O}|$ elements in L (recall that L is the subset of $\mathcal{S}$ that can be partnered with elements in $\mathcal{O}$), then there are $\binom{\pi(\beta^2-\alpha^2)|\mathcal{O}|}{k}$ iterations taking place in step 5. Each iteration costs $k \cdot |\mathcal{O}|$ as we must compare the $|\mathcal{O}|$ bits of each k bit string. So,

$$\binom{\pi(\beta^2 - \alpha^2)|\mathcal{O}|}{k} \cdot k \cdot |\mathcal{O}|$$

$$= \frac{(\pi(\beta^2 - \alpha^2)|\mathcal{O}|) \cdot (\pi(\beta^2 - \alpha^2)|\mathcal{O}| - 1) \cdot \ldots \cdot (\pi(\beta^2 - \alpha^2)|\mathcal{O}| - (k-1))}{k!} \cdot k \cdot |\mathcal{O}|$$

$$< O(\frac{1}{(k-1)!}(\pi(\beta^2 - \alpha^2)|\mathcal{O}|)^{(k+1)})$$

As this term dominates the complexity of the inner loop, the statement follows. □

A.7 Proof of Theorem 4.1

k-SEP$\leq_p$ Set-Cover

PROOF. We employ the first four steps of NAIVE-KSEP-EXACT. We view the bit-strings in list L as subsets of $\mathcal{O}$ where if the ith bit of the string is 1, o_i of $\mathcal{O}$ is in the set.

CLAIM 1: If there are k subsets of L that cover $\mathcal{O}$, then there is a k-sized simple (α, β) explanation for $\mathcal{O}$.
Suppose, by way of contradiction, that there are k subsets of L that cover $\mathcal{O}$ and there is no k-sized simple (α, β) explanation for $\mathcal{O}$. Then, by Proposition 4.1, for every combination of k bit strings, there is some bit i such that $\bigvee_{j=1}^{k}(\ell_j(i)) = 1$ does not hold. Hence, by the reduction, a set cover with k sets from L would be impossible. This is a contradiction.

CLAIM 2: If there there is a k-sized simple (α, β) explanation for $\mathcal{O}$, then there are k subsets of L that cover $\mathcal{O}$.
Suppose, by way of contradiction, there is a k-sized simple (α, β) explanation for $\mathcal{O}$ and there are not k subsets of L that cover $\mathcal{O}$. Then, for any combination of k subsets of L, there is at least one element of $\mathcal{O}$ not included. Hence, for any bit-string representation of an element in L, for some bit i, $\bigvee_{j=1}^{k}(\ell_j(i)) = 1$ does not hold. However, by Proposition 4.1, this must hold or there is no k-sized simple (α, β) explanation for $\mathcal{O}$. This is a contradiction. □

A.8 Proof of Proposition 4.3

NAIVE-KSEP-SC has a complexity of $O(\Delta \cdot f \cdot |\mathcal{O}|^2)$ and an approximation ratio of $1 + ln(f)$.

PROOF. CLAIM 1: NAIVE-KSEP-SC has a complexity of $O(\Delta \cdot f \cdot |\mathcal{O}|^2)$.
The loop at line 6, which reduces the problem to set-covering, takes $O(\Delta \cdot |\mathcal{O}|)$ time.
The loop at line 7 iterates, at most, $|\mathcal{O}|$ times.
The first nested loop at line 7c iterates, at most, $\Delta \cdot |\mathcal{O}|$ times.
The second nested loop at line 7(c)iii iterates, at most, f times.
The updating procedure at line 7e, which is still inside the loop at line 7, iterates, at most, f times.
Hence, by the above statements, the total complexity of NAIVE-KSEP-SC is $O(|\mathcal{O}| \cdot (\Delta \cdot |\mathcal{O}| \cdot f + f) + \Delta \cdot |\mathcal{O}|)$, hence the statement follows.
CLAIM 2: NAIVE-KSEP-SC has an approximation ratio of $1 + ln(f)$.
Viewing list L as a family of subsets, each subset is the set of observations associated with a potential partner, hence the size of the subsets is bounded by f. The approximation ratio follows directly from the analysis of the set-covering problem. □

A.9 Proof of Proposition 4.4

A solution $\mathcal{E}$ to NAIVE-KSEP-SC provides a partner to every observation in $\mathcal{O}$ if a partner exists.

PROOF. Follows directly from Theorem 4.1. □

A.10 Proof of Proposition 4.5

The complexity of KSEP-TO-DOMSET is $O(\Delta \cdot |\mathcal{O}|)$.

PROOF. Notice that the number of points in $\mathcal{S}$ considered for each $o \in \mathcal{O}$ examined in the inner loop is bounded by $O(\Delta)$. As the outer loop is bounded by the size of $\mathcal{O}$, the complexity of KSEP-TO-DOMSET is $O(|\mathcal{O}|)$. □

A.11 Proof of Theorem 4.2

k-**SEP**$\leq_p$ **DomSet**.

PROOF. We can run KSEP-TO-DOMSET that creates graph $G_{\mathcal{O}} = (V_{\mathcal{O}}, E_{\mathcal{O}})$ based on the set of observations. We show that $G_{\mathcal{O}}$ has a dominating set of size k iff there is a k-sized simple (α, β) explanation for $\mathcal{O}$.

Item	Quantity
Number of elements to be covered (number of nodes in $G_{\mathcal{O}}$)	$2 \cdot \Delta \cdot \lvert\mathcal{O}\rvert$
Number of subsets (number of nodes in $G_{\mathcal{O}}$)	$2 \cdot \Delta \cdot \lvert\mathcal{O}\rvert$
Number of elements per subset (Maximum degree of nodes in $G_{\mathcal{O}}$ determined by the produce of partners per observation and observations per partner	$2 \cdot \Delta \cdot f$

Table VII. Quantities for the Greedy-Approach in the **DomSet** reduction.

CLAIM 1: If $G_{\mathcal{O}}$ has a dominating set of size k or less, then there is a k-sized (or less) simple (α, β) explanation for $\mathcal{O}$.
Suppose, by way of contradiction, that $G_{\mathcal{O}}$ has a dominating set of size k and there is not a k-sized simple (α, β) explanation for $\mathcal{O}$. Then, there has to be at least one element $o_i \in \mathcal{O}$ such that there is no feasible $p \in \mathcal{S}$ where $\alpha \leq d(o_i, p) \leq \beta$. Consider the nodes V_i from the inner loop of KSEP-TO-DOMSET that are associated with o_i. Note that these nodes form a complete subgraph. As each node in V_i is associated with o_i, no node in V_i can be in the dominating set of $G_{\mathcal{O}}$ (if one were, then we would have a contradiction). However, note that half of the nodes in V_i only have edges to other nodes in V_i, so there must be an element of V_i in the dominating set. This is a contradiction.
CLAIM 2: If there is a k-sized simple (α, β) explanation for $\mathcal{O}$, then $G_{\mathcal{O}}$ has a dominating set of size k or less.
Suppose, by way of contradiction, that there is a k-sized simple (α, β) explanation for $\mathcal{O}$, and $G_{\mathcal{O}}$ has does not have a dominating set of size k or less. Let $\mathcal{E}$ be a k-sized simple (α, β) explanation for $\mathcal{O}$. Let this also be a subset of the nodes in $G_{\mathcal{O}}$. By the KSEP-TO-DOMSET, in each set of nodes V_i, there must be at least one element of $\mathcal{E}$. As each set of vertices V_i is a complete graph, then we have a dominating set of size k. Hence, a contradiction.

□

A.12 Proof of Proposition 4.6

Solving k-**SEP** by a reduction to **DomSet** utilizing a straight-forward greedy approach has time-complexity $O(\Delta^3 \cdot f \cdot |\mathcal{O}|^2)$ and an approximation ratio bounded by $O(1 + \ln(2 \cdot f \cdot \Delta))$.

Proof. This is done by a well-known reduction of an instance of **DomSet** into an instance of **Set-Cover**. In the reduction, each node is an element, and the subsets are formed by each node and its neighbors. The Table VII shows the quantities:

Hence, the total time complexity of the algorithm is $O(8 \cdot \Delta^3 \cdot f \cdot |\mathcal{O}|^2)$ and the complexity part of the statement follows. As the maximum number of elements per subset, the approximation ratio $O(1 + \ln(2 \cdot f \cdot \Delta))$ follows by the well-known analysis of the greedy set-covering algorithm. □

A.13 Proof of Proposition 4.7

Solving k-**SEP** by a reduction to **DomSet** utilizing the distributed, randomized algorithm presented in [Jia et al. 2002] has a time complexity $O(\Delta \cdot |\mathcal{O}| + ln(2 \cdot \Delta \cdot |\mathcal{O}|) \cdot ln(2 \cdot \Delta \cdot f))$ with high probability and approximation ratio of $O(1+\ln(2 \cdot f \cdot \Delta))$.

PROOF. By Proposition 4.5, the complexity of KSEP-TO-DOMSET is $O(\Delta \cdot |\mathcal{O}|))$. The graph $G_{\mathcal{O}}$ has $O(2 \cdot \Delta \cdot |\mathcal{O}|)$ nodes, and the maximum degree of each node is bounded $2 \cdot \Delta \cdot f$ as per Proposition 4.6. As the algorithm in [Jia et al. 2002] has a complexity of $O(lg(n) \cdot lg(d))$ (with high probability) where n is the number of nodes and d is the maximum degree, the complexity of this approach requires $O(\Delta \cdot |\mathcal{O}| + ln(2 \cdot \Delta \cdot |\mathcal{O}|) \cdot ln(2 \cdot \Delta \cdot f))$ with high probability (the statement follows).

As the approach in [Jia et al. 2002] is greedy, it maintains the $O(1 + \ln(2 \cdot f \cdot \Delta))$ (Proposition 4.6) (the approximation ratio in this case being a factor of the optimal in expectation). □

A.14 Proof of Proposition 4.8

OPT-KSEP-IPC consists of $O(|\mathcal{O}|\pi(\beta^2 - \alpha^2))$ variables and $1 + |\mathcal{O}|$ constraints.

PROOF. Follows directly from Definition 4.1. □

A.15 Proof of Proposition 4.9

For a given instance of the optimization version k-**SEP**, if OPT-KSEP-IPC is solved, then $\bigcup_{p_j \in L_{x_j=1}} p_j$ is an optimal solution to k-**SEP**.

PROOF. Suppose, by way of contradiction, that $\bigcup_{p_j \in L_{x_j=1}} p_j$ is not an optimal solution to k-**SEP**. By the constraint, $\forall o_i \in \mathcal{O}$, $\sum_{p_j \in L} x_j \cdot str(p_j)_i \geq 0$, we are ensured that for each observation, there is a partner p_j such that $x_j = 1$. Further, if we associate x_j with the selected parter p_j for any solution $\mathcal{E}$ to k-**SEP**, then this constraint must hold. Hence, $\bigcup_{p_j \in L_{x_j=1}} p_j$ is a valid explanation. Therefore, the optimal solution to the instance of k-**SEP**, we shall call $\mathcal{E}_{OPT}$, must be smaller than $\bigcup_{p_j \in L_{x_j=1}} p_j$. As the minimization of $\sum_{p_j \in L} x_j$ ensures that the cardinality of $\bigcup_{p_j \in L_{x_j=1}} p_j$ is minimized. Therefore, $|\mathcal{E}_{OPT}|$ cannot be smaller than $|\bigcup_{p_j \in L_{x_j=1}} p_j|$, as the constraint $\forall o_i \in \mathcal{O}$, $\sum_{p_j \in L} x_j \cdot str(p_j)_i \geq 0$ holds for any solution to k-**SEP**. This is a contradiction. □

A.16 Proof of Proposition 4.10

NAIVE-KSEP-ROUND returns an explanation for $\mathcal{O}$ that is within a factor f of optimal, where f is the maximum number of possible partners associated with any observation.

PROOF. [Hochbaum 1982] shows that the solution to the relaxation of the integer program representation of set-cover approximates the optimal solution within a factor of f, which is the greatest number of sets an element can be found in. For k-**SEP**, this would be the greatest number of partners for any given observation, which is bounded by $O(\pi(\beta^2 - \alpha^2))$, but may be much lower in practice - particularly

if the $\equiv_{\mathcal{O}}$ heuristic is employed. As OPT-KSEP-IPC employs this technique, the statement follows directly. □

A.17 Proof of Proposition 5.1

GREEDY-KSEP-OPT1 has a complexity of $O(\Delta \cdot f \cdot |\mathcal{O}|)$ and an approximation ratio of $1 + ln(f)$.

PROOF. CLAIM 1: GREEDY-KSEP-OPT1 has a complexity of $O(\Delta \cdot f \cdot |\mathcal{O}|)$. This follows the same analysis of NAIVE-KSEP-SC in Proposition 4.3, except that line 4 iterates only Δ times rather than $\Delta \cdot |\mathcal{O}|$ times. Hence, the total complexity is $O(|\mathcal{O}| \cdot (\Delta \cdot f + f) + \Delta \cdot |\mathcal{O}|)$ and the statement follows.

CLAIM 2: GREEDY-KSEP-OPT1 has an approximation ratio of $1 + ln(f)$.
The proof of this claim resembles the approximation proof of the standard greedy algorithm for set-cover (i.e. see [Cormen et al. 2001] page 1036).

Let $p_1, \ldots, p_i, \ldots, p_n$ be the elements of $\mathcal{E}$, the solution to GREEDY-KSEP-OPT1, numbered by the order in which they were selected. For each iteration, let set COV_i be the subset of observations that are partnered for the first time with point p_i. Note that each element of $\mathcal{O}$ is in exactly one COV_i. For each $o_j \in \mathcal{O}$, we define $cost_j$ to be $\frac{1}{|COV_i|}$ where $o_j \in COV_i$.

CLAIM 2.1: $\sum_{p_i \in \mathcal{E}^*} \sum_{o_j \in \mathcal{O}_{p_i,o_j} \textit{ are partners}} cost_j \geq |\mathcal{E}|$
By the definition of $cost_j$, exactly one unit of cost is assigned every time a point is picked for the solution $\mathcal{E}$. Hence,

$$COST(\mathcal{E}) = |\mathcal{E}| = \sum_{o_j \in \mathcal{O}} cost_j$$

The statement of the claim follows.

CLAIM 2.2: For some point $p \in L$, $\sum_{o_j \in \mathcal{O}_{p,o_j} \textit{ are partners}} cost_j \leq 1 + \ln(f)$.
Let P be the subset of $\mathcal{O}$ that can be partners with p. At each iteration i of the algorithm, let $uncov_i$ be the number of elements in P that do not have a partner. Let $last$ be the smallest number such that $uncov_{last} = 0$. Let $\mathcal{E}_P = \{p_i \in \mathcal{E} | (i \leq last) \wedge (COV_i \cap P \not\equiv \emptyset)\}$. From here on, we shall renumber each element in $\mathcal{E}_P$ as $p_1, \ldots, p_{|\mathcal{E}_P|}$ by the order they are picked in the algorithm (i.e. if an element is picked that cannot partner with anything in P, we ignore it and continue numbering with the next available number, we will use this new numbering for COV_i and the iterations of the algorithm as well, but do not re-define the set based on the new numbering).

We note that for each iteration i, the number of items in P that are partnered is equal to $uncov_{i-1} - uncov_i$. Hence,

$$\sum_{\substack{o_j \in \mathcal{O} \\ p,o_j \textit{ are partners}}} cost_j = \sum_{i=1}^{last} \frac{uncov_{i-1} - uncov_i}{|COV_i|}$$

At each iteration of the algorithm, let $PCOV_i$ be the subset of observations that are covered for the first time if point p is picked instead of point p_i. We note, that for all iterations in $1, \ldots, last$, the point p is considered by the algorithm as one of its options for greedy selection. Therefore, as p is not chosen, we know that $|COV_i| \geq |PCOV_i|$. Also, by the definition of $ucov_i$, we know that $|PCOV_i| = ucov_{i-1}$. This gives us:

$$\sum_{\substack{o_j \in \mathcal{O} \\ p, o_j \text{ are partners}}} cost_j \leq \sum_{i=1}^{last} \frac{uncov_{i-1} - uncov_i}{ucov_{i-1}}$$

Using the algebraic manipulations of [Cormen et al. 2001] (page 1037), we get the following:

$$\sum_{\substack{o_j \in \mathcal{O} \\ p, o_j \text{ are partners}}} cost_j \leq H_{|P|}$$

Where H_j is the jth harmonic number. By definition of the symbol f (maximum number of observations supported by a single partner), we obtain the statement of the claim.

(Proof of claim 2): Combining claims 1-2, we get $|\mathcal{E}| \leq \sum_{p_i \in \mathcal{E}^*}(1 + \ln(f))$, which gives us the claim. □

A.18 Proof of Proposition 5.2

GREEDY-KSEP-OPT1 returns a $|\mathcal{E}|$-sized (α, β) explanation for $\mathcal{O}$.
GREEDY-KSEP-OPT1 returns IMPOSSIBLE if there is no explanation for $\mathcal{O}$.

Proof. Suppose by way of contradiction that there exists and element $o \in \mathcal{O}$ such that there is no in $\mathcal{E}$. We note that set $\mathcal{O}'$ contains all elements of $\mathcal{O}$ and the only way for an element to be removed from $\mathcal{O}'$ is if a partner for that element is added to $\mathcal{E}$. Hence, if the program returns a set $\mathcal{E}$, we are guaranteed that each $o \in \mathcal{O}$ has a partner in $\mathcal{E}$.

Suppose by way of contradiction that GREEDY-KSEP-OPT1 returns IMPOSSIBLE and there exists a set $\mathcal{E}$ that is a valid (α, β) explanation for $\mathcal{O}$. Then, for every element of $\mathcal{O}$, there exists a valid partner. However, this contradicts line 6b of NAIVE-KSEP-SC (called by line 4b of GREEDY-KSEP-OPT1) which causes the program to return IMPOSSIBLE only if an element of $\mathcal{O}$ is found without any possible partner. □

A.19 Proof of Theorem 5.1

GREEDY-KSEP-OPT2 has a complexity of $O(\Delta \cdot f^2 \cdot |\mathcal{O}| + |\mathcal{O}| \cdot \ln(|\mathcal{O}|))$ and an approximation ratio of $1 + \ln(f)$.

Proof. CLAIM 1: GREEDY-KSEP-OPT2 has a complexity of $O(\Delta \cdot f^2 \cdot |\mathcal{O}| + |\mathcal{O}| \cdot \ln(|\mathcal{O}|))$.
Line 1 takes $O(\Delta \cdot |\mathcal{O}|)$ time.
The loop starting at line 4 iterates $|\mathcal{O}|$ times.

The nested loop at line 4a iterates Δ times.
The second nested loop at line 4(a)i iterates f times. The inner body of this loop can be accomplished in constant time.
In line 5, initializing the Fibonacci heap takes constant time, as does inserting elements, hence this line takes only $O(|\mathcal{O}|)$ time.
The loop at line 6 iterates, at most, $|\mathcal{O}|$ times.
Viewing the minimum of a Fibonacci heap, as in line 6a can be done in constant time.
As per the analysis of GREEDY-KSEP-OPT1, line 6b takes $\Delta \cdot f$ iterations. The updating procedure starts with line 6c which iterates f times.
The removal of an elements in line 6(c)ii from a Fibonacci heap costs $O(\ln(|\mathcal{O})$ amortized time. However, we perform this operation no more than $|\mathcal{O}|$ times, hence we can add $|\mathcal{O}| \cdot \ln(|\mathcal{O}|))$ to the complexity.
Note that the size of a list pointed to by REL_OBS$[o']$ is bounded by $\Delta \cdot f$ - f observations associated with each of Δ partners - hence line 6(c)iii iterates, at most, $\Delta \cdot f$ times.
We note that decreasing the key of an item in the Fibonacci heap (in line 6(c)iii) takes constant time (amortized).
Therefore, by the above statements, the complexity of GREEDY-KSEP-OPT2 is $O(|\mathcal{O}| \cdot (\Delta \cdot f + \Delta \cdot f^2) + |\mathcal{O}| \cdot \ln(|\mathcal{O}|) + \Delta \cdot f \cdot |\mathcal{O}| + \Delta \cdot |\mathcal{O}|)$ and the statement follows.
CLAIM 2: GREEDY-KSEP-OPT2 has an approximation ratio of $1 + \ln(f)$.
Follows directly from Proposition 5.1. □

A.20 Proof of Proposition 5.3

GREEDY-KSEP-OPT2 returns a $|\mathcal{E}|$-sized (α, β) explanation for $\mathcal{O}$.
GREEDY-KSEP-OPT2 returns a IMPOSSIBLE if there is no explanation for $\mathcal{O}$.

Proof. Mirrors that of Proposition 5.2. □