Decentralized spatial data mining for geosensor networks

Patrick Laube and Matt Duckham  
Department of Geomatics  
The University of Melbourne  
Victoria 3010  
Australia  
telephone: +61 3 8344 {9185,6935}  
fax: +61 3 9347 2916  
{plaube,mduckham}@unimelb.edu.au  

December 3, 2007

Abstract

Advances in distributed sensing and computing technology offer new, reliable, and cost-effective means to collect fine-grained spatiotemporal data. Conventional spatiotemporal data mining procedures, however, are based on centralized models of information processing, where sophisticated and powerful central systems collate and process global information. By contrast, decentralized spatial computing systems require new techniques for in-network knowledge discovery. This chapter introduces the notion of decentralized spatial data mining, where individual sensor-enabled computing nodes possess only local knowledge about their immediate neighborhood, but derive global knowledge through local collaboration and information exchange. The chapter then presents four strategies for decentralized spatial data mining, illustrating the concept of decentralization with three simple decentralized algorithms for the classical spatial data mining task of clustering.

Keywords: spatial data mining, distributed systems, decentralized spatial computing, geosensor networks, clustering

1 Introduction

Conventional approaches to geographic knowledge discovery and spatial data mining are founded on powerful, centralized algorithms that screen large data sets for interesting patterns and rules. Such global algorithms allow fast detection of interesting patterns if centralized access to the whole data set can be guaranteed.
However, new technologies for distributed spatial data capture and processing, such as geosen-
sor networks, present new challenges to conventional knowledge discovery and data mining al-
gerithms. Increasingly, access to the whole data set cannot be guaranteed; instead multiple com-
puting units, none of which possess global knowledge, must cooperate in knowledge discovery.

This chapter investigates the structure and design of decentralized algorithms for spatial data
mining. Our increasing ability to collect data at finer and finer spatiotemporal granularities has
the potential to generate such overwhelming volumes of data that the paradigm of central pro-
cessing is no longer practicable (Kargupta and Chan 2000). Instead, knowledge discovery must
descend into the network, detecting patterns as the spatial data is captured.

Specifically, the chapter contributes to the theory of geographic knowledge discovery and
spatial data mining by:

- identifying spatial data mining and knowledge discovery as a crucial application layer for
geosensor networks, the latest technology for spatial data capture;

- exploring the notion of decentralized spatial data mining (DSDM) for geographic knowl-
edge discovery; and

- presenting an overview of techniques for DSDM, including an investigation of the potential
of DSDM for classical spatial data mining applications, such as clustering.

This remainder of this chapter is organized as follows. Section 2 surveys the relevant back-
ground literature for DSDM. Section 3 presents the concept of decentralized spatial data mining
and proposes a set of generic strategies for DSDM algorithms. The chapter then investigates a
specific case study of different decentralized algorithms for spatial clustering (sections 4 and 5).
Finally, the chapter concludes with a discussion of the results in section 6 and the formulation of
a research agenda for DSDM (section 7).

2 Background

2.1 Distributed and decentralized spatial computing

A distributed system is defined as a collection of multiple information system units that syn-
chronously cooperate via a communication network to complete some computing task (Worboys
and Duckham 2004). A wireless sensor network (WSN, ad-hoc wireless networks of sensor-
enabled miniature computing platforms, Zhao and Guibas 2004) is a form of distributed system,
where individual sensor nodes cooperate to ensure the network as a whole can meet the require-
ments of the specific application. Applications of WSN in the spatial domain include environ-
mental monitoring (Duckham et al. 2005, Werner-Allen et al. 2006), smart farming (Wark et al.
2007), traffic management (Kellerer et al. 2001), and robotics (Correll and Martinoli 2006).
Considerable recent research activity in the area of WSN has focused on the issues surround-
ing the establishment and maintenance of the communication networks necessary for distributed
computing (e.g., Braginsky and Estrin 2002, Cheng and Heinzelman 2005), including many ingenious techniques using the spatial characteristics of the network for that purpose (e.g., Karp and Kung 2000, Mauve et al. 2001, Yu et al. 2001, Xu et al. 2001).

In many systems that are commonly referred to as “distributed,” the cooperating information systems each take responsibility for logically or functionally distinct sub-tasks. For example, the architecture of distributed client-server systems is typically founded on a clear delineation of the distinct services provided and consumed by different logical units (e.g., the classic three-tier client-server architecture of web browser, web server, and spatial database server used in web mapping applications, Worboys and Duckham 2004). However, in some distributed systems, such as peer-to-peer networks, there is no such partitioning of sub-tasks; multiple units in the distributed system may have similar or equivalent responsibilities. In such system, specific processing tasks may be distributed throughout the network, each individual unit performing a small part of the required processing. Here we reserve the term decentralized for describing these distributed systems and algorithms, where the processing task itself is distributed throughout the network and no component of the distributed system “knows” the entire system state (Lynch 1996).

A geosensor network is defined by Nittel et al. (2004) as a wireless sensor network that monitors phenomena in geographic space. A geosensor networks is, therefore, also a type of distributed system. There are four main reasons why decentralized algorithms are important in geosensor networks.

- **Energy resources**: WSN are highly resource-constrained systems, especially with respect to sensor node energy resources (Zhao and Guibas 2004). Wireless communication is one of the most energy-intensive activities of a sensor node, so continually relaying data to a central system can dramatically shorten the useful lifetime of a WSN.

- **Information overload**: The fine-grained detail becoming available from larger sensor networks means that individual data items become less and less meaningful. Transmitting all data can lead to high levels of redundancy and ultimately information overload (Rabiner et al. 1999, Datta et al. 2006a).

- **Scalability**: As networks scale from tens to thousands to millions of nodes, effective centralized control of the network becomes impossible. The issue of scalability is especially important in geosensor networks, which must by definition contain large number of nodes in order to provide enough spatial detail to monitor geographic phenomena (Estrin et al. 1999).

- **Sensor/actuator networks**: The results of the analysis of sensor network data are often required by the network itself in order modify the behavior of the network (e.g., activate or deactivate sensors to adapt the granularity of monitoring of important events, Duckham et al. 2005). Removing information from the network, processing it centrally, then returning it to the network is an inefficient drain on network resources.

The key challenge of decentralized, in-network processing is to use “decentralized coordination with local decision making to achieve the intended global goal” (Estrin et al. 2000, p. 40);
in other words, to generate global knowledge using local processes (local in this context refers to a node and its immediate neighborhood or locality). Thus, in decentralized spatial computing we are interested in developing algorithms that can operate using purely local knowledge, but are still able to monitor geographic phenomena with global extents. This is very different approach from conventional spatial computing paradigms (exemplified by GIS) where processes (e.g., spatial analysis routines) operate upon entire data sets (e.g., stored in a spatial database).

2.2 Centralized (geographic) knowledge discovery and data mining

Conventional knowledge discovery in databases (KDD), and its most prominent step data mining, rely on data available at a single location. Association rule mining, for example, is based on global counts of frequent item sets in order to compute support for and confidence in a rule (Gidofalvi and Pedersen 2005). Similarly, many point pattern measures used for clustering purposes depend on globally fixed criteria such as “nearest neighbor” or “neighbors within 50m” (O’Sullivan and Unwin 2003). Even though classical spatial data mining patterns fundamentally depend on local spatial relations (Shekhar et al. 2003), the vast majority of current algorithms for detecting these patterns rely on global data structures and algorithms.

For example, clustering is a classic spatial data mining technique that organizes observations into coherent and contrasted groups. Clustering approaches are normally classified into two categories: hierarchical and partitional clustering. Hierarchical clustering techniques establish a nested hierarchy of clusters by successively building new clusters basing on previously merged leaves in the clustering tree (O’Sullivan and Unwin 2003). At every step, merging the closest clusters requires finding the smallest distance of any pair in the distance matrix for the whole data set, and hence relies on global knowledge. The most common partitional algorithm is k-means clustering. The algorithm is based on an initial assignment of all observations to k randomly seeded cluster heads followed by successive improvement of the partitioning by iterative reassessment of computed mean centers of the partitions. In its conventional variants, k-means clustering assumes global knowledge (but see section 2.3).

Spatial and spatiotemporal clustering often takes a different perspective and first asks if there are clusters at all in some given point distribution. In the field of spatial statistics a series of techniques have been developed to quantify the randomness of a point distribution, including density-based methods (quadrat count, kernel estimation) and distance-based methods (nearest-neighbor, distance functions, O’Sullivan and Unwin 2003). The locations of potential clusters are then the focus of a second stage. A common application would be the identification of crime hot spots or the origin of an infectious disease (Shekhar et al. 2003). Again these centralized algorithms require global access to data to operate.

In summary, conventional data mining approaches allow efficient screening for patterns and rules, given the proviso that all data is available at a single location, data structures are centralized, and algorithms omniscient.
2.3 Distributed data mining

The emergence of network-based distributed computing environments has added a new dimension to knowledge discovery in databases and data mining. Distributed data mining (DDM) has evolved over the last decade in an attempt to develop distributed versions of many standard data mining algorithms (Datta et al. 2006a). DDM embraces the growing trend of merging computation with communication. DDM aims at finding patterns and rules from distributed and heterogeneous data using minimal communication (Kargupta and Chan 2000). Privacy concerns, as well as bandwidth and resource constraints in distributed systems, often dictate that data collected at different nodes be analyzed in a decentralized fashion, without collecting everything to a central site (Datta et al. 2006a). The limitations of using purely local knowledge means distributed data mining often focuses on approximate algorithms that may not always match the exact answers provided by conventional centralized data mining algorithms (Datta et al. 2006a). The goal remains to derive new and useful information, but potentially to sacrifice a small degree of certainty for substantial computational gains.

Taking for instance clustering, there exists ample research on distributed clustering algorithms (see Bandyopadhyay et al. 2006 for an introductory overview). For example, clustering of sensor nodes can be used for communication load balancing in ad-hoc sensor networks (Younis and Fahmy 2004). Other authors have explicitly focused on distributed clustering for data mining purposes (Bandyopadhyay et al. 2006) and even dynamic distributed networks (Datta et al. 2006b). However, most distributed clustering approaches so far focus on partitional clustering, assigning the nodes of a network to a given number of \( k \) cluster heads. Apart from clustering there also exists work on distributed in-network association rule mining (Wolff and Schuster 2004) and outlier detection (Branch et al. 2006).

The distributed data mining field shows a growing interest in distributed partitional clustering. However, decentralized spatial data mining, for example decentralized cluster detection and localization in spatial point distributions, remains an open research task.

3 Decentralized spatial data mining (DSDM)

This section discusses the concept of decentralized spatial data mining (DSDM). Many of the patterns and rules of interest in conventional spatial data mining are defined based on local inter-object relationships, including density clusters and co-location patterns. This spatial locality is exploited by decentralized algorithms for DSDM. The section first defines the problem addressed by DSDM, then proposes a series of general strategies for DSDM.

3.1 Problem definition

Distributed data mining (DDM) is the attempt to develop distributed versions of standard data mining algorithms (Datta et al. 2006a). Similarly, the aim of decentralized spatial data mining is the introduction of decentralized algorithms for spatial data mining. Like spatial data mining, DSDM gains its strength from exploiting the special characteristics of spatial information (pri-
marily, spatial autocorrelation). Hence, DSDM is not about functionally distributing a complex task amongst cooperating sub-systems. Instead, DSDM attempts to spatially distribute a global task throughout a decentralized network, each individual computing unit relying on local data and processes to operate.

Consequently, decentralized spatial data mining can be defined as the process of discovering new spatial patterns within a distributed system using decentralized algorithms with no central coordination operating upon locally defined spatial data.

For example, consider the simple spatial cluster detection illustration in Figure 1. The black observations lie in two distinctive density clusters. The task of DSDM is to detect these clusters using decentralized data mining algorithms that operate in computing nodes themselves. Each node is expected to be able to perceive only local information about its own and its immediate neighbors’ geographic environment, but there exists no node that can perceive the entire geographic space.

Figure 1: Decentralized clustering. A set of $V$ observations features two clusters of hot nodes. Detecting such a clustering is easier given a larger neighborhood $N_1$ covering the whole cluster, but more difficult in a smaller neighborhood $N_2$ that only covers parts of the cluster. Applied to a geosensor network scenario, where the observations represent the nodes, the DSDM task is the decentralized detection of the clusters.

Environmental monitoring using a geosensor network would be one specific application of the example above. The observations represent sensor nodes in a geosensor network, physically distributed in space, and monitoring some environmental variable, such as temperature. Given
some threshold \( t \), then the clusters of “hot” nodes might represent temperature “hot spots,” where the temperature is above \( t \). Each individual sensor node is expected to possess only partial spatial knowledge about its own temperature and the temperature of its immediate neighbors. The DSDM task for this geosensor network is then to detect the temperature hot spots using only its local knowledge about its own and its neighbors’ observations.

Obviously, the size of neighborhood is a central parameter in this example. The larger the neighborhood, the more the problem resembles a conventional centralized, global spatial data mining problem. Neighborhood \( N_1 \), for example, covers the entire cluster bottom left and it should potentially be straightforward for a node with only local knowledge of that neighborhood still to detect the pattern. By contrast, neighborhood \( N_2 \) only covers a fraction of the whole cluster top right, making decentralized detection of this cluster much harder. In the context of a geosensor network, the size of the neighborhood will be determined to a large extent by the communication range of individual nodes. The technical and physical limitations of communication in geosensor networks means that in general it is to be expected that each sensor node can only communicate with a tiny fraction of the nodes in the entire network: those in its immediate spatial vicinity.

3.2 Formal problem definition

In this section we more precisely specify the problem outlined above using a formal model of geosensor networks. For simplicity we assume only a static geosensor network, where nodes are immobile. However, later sections indicate how this model can be extended to deal with dynamism.

3.2.1 Geosensor networks

As indicated previously, the key features of a geosensor network are the nodes and short-range radio frequency (RF) communication links between nearby nodes. The most commonly used model of such a network is as a graph, where vertices in the graph model nodes in the geosensor network and edges in the graph model the potential for communication between neighboring nodes. Such a graph is static (nodes do not move and edges are fixed), and can be formally defined as in definition 3.2.1.

**Definition 3.2.1** A geosensor network may be modeled as a graph \( G = (V, E) \), where \( V \) is the set of vertices (sensor nodes) and \( E \subseteq V \times V \) is the set of edges (communication links) between neighboring nodes. For a node \( v \in V \), its neighborhood \( \{ v' \in V \mid \{v, v'\} \in E \} \) is written \( nbr(v) \).

Note that by adopting an *undirected* graph to model a geosensor network (as in definition 3.2.1), we are implicitly assuming symmetric bidirectional communication: if node \( a \) can communicate with node \( b \), then node \( b \) can communicate with node \( a \). While this is a natural and common simplifying assumption in geosensor networks, in actuality it does not always hold (Min and Chandrakasan 2003). In more sophisticated situations, a directed graph might be required to model any communication asymmetry.
We can further model the location of a sensor node as a \textit{locator} function (definition 3.2.2).

\textbf{Definition 3.2.2} A (static) \textit{locator} is a function \( l : V \rightarrow \mathbb{R}^n \), where for any vertex \( v \in V \), \( l(v) \) maps to the coordinate location of that node (where \( n \) is 2 or 3). The distance function \( \delta : V \times V \rightarrow \mathbb{R} \) is the usual metric for Euclidean distance between nodes.

An implicit assumption is commonly made that no two nodes occupy the same location (i.e., the locator function is an injection). Because the communication links between nodes are constrained by the physical limitations of RF communication, the locator function can be used to generate the set of edges \( E \) for a particular set of vertices \( V \) assuming a maximum communication range \( c \). For example, the \textit{unit distance graph} (UDG) is the graph formed when all nodes that are within communication range may potentially communicate (definition 3.2.3).

\textbf{Definition 3.2.3} Given a maximum communication distance \( c \), the \textit{unit distance graph} (UDG) is the geosensor network \( G = (V, E) \) where \( E = \{ (u, v) \in V \times V | 0 < \delta(u, v) \leq c \text{ and } u \neq v \} \).

Note that in addition to assuming symmetric, bi-directional communication, the UDG also assumes a constant communication distance across the entire network. Again, the actual situation may in practice be more complex.

Often in spatial applications it is more useful to assume that only a subset of the communication links in the UDG are available. In particular, subsets of the UDG that form planar graphs (such as triangulations) are commonly used in specific spatial applications (e.g., Karp and Kung 2000, Worboys and Duckham 2006). Common planar subsets of the UDG include the relative neighborhood graph (RNG) and the Gabriel graph (GG) (Zhao and Guibas 2004).

It is important to note that although the underlying communication graph is usually constructed with reference to a locator function \( l \), we do not necessarily assume that an individual node is location-aware (i.e., has access to knowledge about its own location). In some geosensor networks, all nodes may be location aware. However, the technical limitations of achieving high precision and accuracy location of nodes in a geosensor networks mean that where possible it is safer to assume nodes are only able to determine their qualitative location in terms of the nodes in their immediate neighborhood. Despite this limitation, it is possible to generate many interesting spatial properties and behaviors using only such qualitative location information, as we shall see later in section 4.

Finally, the geosensor network is assumed to be monitoring some environmental variable in space using its sensors. The environmental variable may itself be highly structured and complex, however in this chapter we assume the most simple domain for an environmental variable: Boolean values.

\textbf{Definition 3.2.4} The sensor data for the set of nodes \( V \) can be represented using a (static) \textit{sensor} function \( s : V \rightarrow D \), where \( D \) is the domain for some environmental variable. In this chapter we assume a Boolean domain \( D = \{0, 1\} \). Any node \( v \) where \( s(v) = 1 \) (i.e., \( \text{that can detect the environmental variable} \) is termed a “hot” node; any node \( v \) where \( s(v) = 0 \) (i.e., \( \text{that cannot detect the environmental variable} \) is termed a “cold” node.

Figure 2 summarizes the formal model of the geosensor network, showing the UDG for a small group of hot (black) and cold (white) nodes.
Figure 2: **Summary of formal model.** Geosensor network $G = (V, E)$, where $V = \{v_1, ..., v_{20}\}$ and $E = \{\{v_1, v_2\}, ...\}$ is the UDG based on communication distance $c$. Hot nodes, where $s(v) = 1$, are shown in black; cold nodes in white.

### 3.2.2 Clusters

The decentralized algorithms introduced in later sections are designed to find clusters, such as those in Figure 1. Here we adopt a simple definition of a cluster as a set of at least $n$ related observations that lie within a circle of radius $r$ spatial region (definition 3.2.5).

**Definition 3.2.5** Given a geosensor network $G = (V, E)$, a locator function $l : V \rightarrow \mathbb{R}^2$, and a sensor function $s : V \rightarrow \{0, 1\}$, an $(nr)$ cluster is defined as a set of nodes $V' \subseteq V$ such that $|V'| \geq n$; there exists some circle $e_r$ of radius $r$ such that for all $v \in V'$, $l(v)$ is spatially contained within $e_r$; and for all $v \in V'$, $s(v) = 1$.

Note that this definition concerns only the static scenarios outlined above, although more sophisticated dynamic clustering definition can also be defined based on the definitions above.

The problem facing a DSDM algorithm is how to detect a cluster of $n$ nodes within a circle of radius $r$, when individual nodes $v \in V$ can only communicate with their immediate neighbors, $v'$ such that $\{v, v'\} \in E$. If we assume the neighborhood of a node is defined by the UDG, then the neighborhood of $v$ depends on the communication radius $c$. Assuming $c$ is substantially larger than $r$, then finding a cluster is relatively straightforward: each node can locally look for a cluster only in its immediate neighborhood. However, when $c$ and $r$ are similar, or $c$ is smaller than $r$, more sophisticated strategies are required.

### 3.3 Compensation strategies

Based on the problem definition above, it is possible to describe four classes of strategies for decentralized spatial data mining as follows.
Figure 3: **Four compensation strategies.** Clockwise from top left: local extrapolation, local absorption, selective collaboration, node mobility.

### 3.3.1 Local extrapolation

In local extrapolation, nodes infer knowledge about patterns that extend their local knowledge range. For example, if $c = \frac{1}{2}r$, a node $v$ can expect to only receive knowledge on approximately $\frac{1^2}{3^2} = \frac{1}{9}$ of the entire pattern extent from its immediate neighbors $nbr(v)$. Thus, if a node detects $\frac{1}{4}$ of the required cluster (i.e., for all $v' \in nbr(v)$, $s(v') = 1$ but $|nbr(v)| \approx \frac{1}{4} n$), it may locally infer that the cluster has been detected. For example, the clusters in Figure 3 consist of 20 hot nodes. In the top left case the node with communication range $c$ has 5 hot neighbors and hence naively assumes that it has detected a cluster.

### 3.3.2 Local absorption

Nodes can reach beyond the limits of their communication ranges by absorbing their neighbors’ knowledge. In other words, nodes propagate their knowledge by locally restricted flooding (Zhao and Guibas 2004). If $c = \frac{1}{2}r$, nodes may reach out to the edges of patterns by relaying knowledge about their neighbors to their other neighbors (termed two-hop communication). If the central node in the example top right in Figure 3 receives knowledge from neighbors up to two hops away, it will be able to locally infer the presence of the clustering pattern. The logical extreme of local absorption is to flood knowledge throughout the network (multi-hop communication). In such a case, every node could possess knowledge about the state of all other nodes in the network.
However, as already discussed, the physical and technical limitations of sensor networks makes such an approach unscalable and impractical. Hence, local absorption must typically be limited to only a few hops.

### 3.3.3 Selective collaboration

A third compensation strategy is to invoke more targeted communication between nodes only if some pre-defined condition is met. Such conditions are similar to certificates used for kinetic data structures (Guibas 2002). In that context, data structures for dynamic systems only update when some local certificate (that is some elementary relations among the objects involved) is violated. In a decentralized data mining context, nodes do not communicate until they have good reason to believe that they might be involved in a pattern. Then nodes then select other nodes (typically close neighbors) to solve collaboratively the task at hand. In Figure 3 (bottom right) the central node has detected a certain number of hot nodes within communication range and established a collaboration with six nearby nodes in order to cover the pattern. Since the selected nodes may not necessarily be in the immediate neighborhood of a node (i.e., \( n' \notin \text{nbr}(n) \)), selective collaboration may require more sophisticated routing protocols to organize communication between remote nodes. Schemes for such routing protocols are legion in the literature, so in this chapter we don’t consider this issue further.

### 3.3.4 Node mobility

In more dynamic situations than considered thus far, an important possibility is for nodes to extend their spatially limited communication range through mobility. As mobile nodes move around the geographic space, they “see” different parts of the geographic area and can potentially communicate with different neighbors.

Given a set \( T \) of discrete, totally ordered times \( \{t_1, \ldots, t_n\} \), we can extend the formal definitions presented in section 3.2 to model mobility. Assuming the number of nodes in the geosensor network is constant, the mobility of nodes can be modeled with a dynamic locator function, \( l : V \times T \rightarrow \mathbb{R}^n \), where for any vertex \( v \in V \) and time \( t \in T \), \( l(v, t) \) maps to the coordinate location of that node at time \( t \). The dynamic neighborhoods can be modeled as a dynamic graph, where the set of edges changes over time. For example, given a maximum communication distance \( c \), the dynamic unit distance graph (UDG) can be defined as \( G(t) = (V, E(t)) \) where \( E(t) = \{(u, v) \in V \times V | 0 < d(l(u, t), l(v, t)) \leq c\} \). Similarly, the changing environmental variables sensed by a node can be modeled as a dynamic sensor function \( s : V \times T \rightarrow \{0, 1\} \).

Node mobility opens at least two options for exploiting mobility for knowledge discovery (Grossglauser and Vetterli 2006, Grossglauser and Tse 2002). First, nodes might “graze” information whilst moving and store it in a constantly updated memory, termed mobility memory. Formally, an individual node \( v \) tracks its sensed values \( s(v, t) \) and potentially its location \( l(v, t) \) over a range of times \( t \), and combines that knowledge in its pattern detection algorithm. Figure 3 (bottom left) illustrates a node passing through a pattern and thereby collecting enough information to reason about the presence of a pattern.
Mobility memory can operate even without any communication between nodes. However, knowledge discovery can clearly be improved by additional enabling nodes to exchange information with their constantly changing neighbors whilst moving, termed mobility diffusion. Formally, an individual node $v$ may communicate information with its neighbors $\{v' \mid \{v, v'\} \in E(t)\}$ over a range of times $t$. In some senses, mobility diffusion can be regarded as inexpensive variant of local absorption, since mobility (rather than multi-hop communication) is used to move information around the system beyond a node’s immediate neighbors at a particular time (Grossglauser and Tse 2002).

Combinations of these individual strategies can be used, and indeed are expected to be more effective than strategies used in isolation. An obvious combination is the use of local extrapolation as a preliminary for other strategies. Local extrapolation can be used to establish a local state of belief about the presence of a pattern. If this state of belief reaches some threshold, it may trigger one of the other more involved methods in order to derive further information.

4 Decentralized spatial clustering algorithms

In this section we present two algorithms implementing two of the above compensation strategies, local extrapolation and local absorption. Although work is ongoing on examples of algorithms in all four categories, in this chapter we restrict the discussion to these two cases because they are relatively easy to grasp and representative of the issues faced in DSDM.

In both algorithms, nodes process locally collected knowledge about their neighborhood in order to develop a state of belief as to whether or not they have detected a cluster. As we like to refer to agents that have detected a pattern as “happy,” our algorithms are termed happiness extrapolation clustering (HEC) and happiness absorption clustering (HAC), with a preliminary base-case algorithm termed naïve clustering.

4.1 Naïve clustering

As discussed above, where the communication range $c$ is substantially greater than the cluster size $r$, it is potentially possible for a node to locally detect a cluster without any need for compensation strategies. Algorithm 1 presents such a naïve base-case algorithm, where each node simply examines those nodes in its immediate neighborhood to determine whether it can locally detect a cluster. The algorithm cycles through every node (line 1.1); checks whether enough hot nodes to form a cluster are within in its neighborhood (line 1.3); and if so whether they lie inside a circle of radius $r$ (line 1.5).

Several points are worth noting about this algorithm. First, the NC algorithm is obviously expected to fail in cases where $c < r$ or even $c \approx r$. Second, in Algorithm 1 several nodes may potentially detect the same cluster. The distinction between cases where a particular node detects a phenomenon, and where some node detects a phenomenon is an important one in DSDM. Normally in DSDM we are interested primarily in the latter situation, where some node detects a phenomenon since individual node behavior is not as important as the overall global network.
Algorithm 1: NC: Naïve Cluster algorithm to check for each node whether it can sense a cluster in its local neighborhood

**Data:** Geosensor network graph $G = (V, E)$; locator function $l : V \rightarrow E$; sensor function $s : V \rightarrow \{1, 0\}$; cluster radius $r$; cluster size $n$

```plaintext
foreach $v \in V$ do
  1.1 $X_v = \{v' \in V | s(v') = 1 \text{ and } v' \in \text{nbr}(v)\};$
  1.2 if $|X_v| \geq n$ then
  1.3 $d = \max_{v_1, v_2 \in \text{nbr}(v)}(\delta(l(v_1), l(v_2)));
  1.4 if $d \geq r$ then
  1.5 Node $v$ has detected a cluster of radius $r$ and size $n$;
```

behavior. In other words, an algorithm can be regarded as successful as long as some node detects a phenomenon, although it may not matter whether any particular node detects it.

Unfortunately, even Algorithm 1 is not guaranteed to detect a cluster in all cases. For larger communication ranges, certain configurations of nodes could result in clusters with a relatively small radius $r$ being missed. A more sophisticated algorithm to solve this would need to perform the computationally intensive process of checking the distances between different permutations of subsets of the entire set of neighbors. For simplicity, and in recognition of the limited processing power of sensor nodes, this more simple algorithm has been preferred here. As already noted, decentralized algorithms are often approximate, in this case potentially missing some clusters (error of omission), although never incorrectly identifying a cluster (error of commission).

It is also important to note that the algorithm does also rely on some quantitative information about each node’s location (location-awareness), or at the very least quantitative information distances between nodes (e.g., using range-finding techniques). As already intimated, in many practical situations for geosensor networks, location-awareness may be unreliable or unavailable, limiting the applicability of such an algorithm.

4.2 Happiness extrapolation clustering (HEC)

Happiness extrapolation clustering (HEC) implements a basic form of local extrapolation for decentralized cluster detection. HEC is based purely on instantaneous and local neighbor counts. Every node in parallel extrapolates its local knowledge and computes its belief in having detected a cluster, or in “being happy” respectively (as illustrated earlier in Figure 1).

In performing the extrapolation an important observation is that it is unlikely that a node will be located exactly in the middle of cluster. Thus, even when $c \approx r$ the naïve clustering algorithm is likely to fail: in such cases although it would be possible for a single node to observe the entire cluster, there is no a priori reason for expecting any node to be so conveniently located. The problem is illustrated in Figure 4, where although $c = r$, there exists no node that can is located in such a way to be able to detect the cluster of radius $r$.

Algorithm 2 addresses this problem using a threshold $t$ (line 2.1) that adjusts a node’s ex-
Figure 4: **Limits of naive clustering.** Nodes randomly distributed within a cluster are unlikely to be in a position to observe a cluster using the naïve cluster algorithm for $c \approx r$. Even though placed centrally, node $v$ misses three hot nodes for $c = r$.

Expectation of what proportion of a cluster size $n$ it should see, given the known ratio between the communication and cluster areas ($c^2 / r^2$). Figure 5 illustrates the threshold $t$ in a graph, plotting the ratio between communication and cluster areas against the expected number of hot neighbors a node should see in a cluster where $n = 10$. In practice $t$ can be empirically determined. For example, for a cluster size of about $n = 10$ a $t$-value of approximately 0.8 provides the required adjustment (i.e., when communication range and cluster radius are the same, a node would expect to see at least 80% of the cluster).

If this adjusted expected number of hot neighbors is greater than the cluster size, then Algorithm 2 (line 2.3) resorts to the naïve cluster algorithm (since we expect some node to be able to observe the entire cluster). Otherwise, Algorithm 2 cycles through each node, checking whether it has enough hot nodes in its neighborhood to justify a belief that it can see a cluster and hence be happy (lines 2.5–2.8).

**Algorithm 2: HEC**: Local extrapolation algorithm to check for each node whether it can locally infer a cluster from its neighborhood.

**Data**: Geosensor network graph $G = (V, E)$; locator function $l : V \rightarrow E$; sensor function $s : V \rightarrow \{1, 0\}$; cluster radius $r$; cluster size $n$

**Set** $x \leftarrow n \times t \times \frac{c^2}{r^2}$;

**if** $x \geq n$ **then**

Use NC algorithm (naïve cluster) to determine whether cluster is detected;

**else**

**foreach** $v \in V$ **do**

$X_v = \{v' \in V | s(v') = 1 \text{ and } v' \in nbr(v)\}$;

**if** $|X_v| \geq x \geq 2$ **then**

Node $v$ has detected a cluster of radius $r$ and size $n$;
Clearly, the smaller the communication range (and so the smaller the number of hot neighbors must be detected for a cluster) the more likely it becomes that a node misidentifies a random constellation of nodes as a cluster. Consequently, for small communication ranges the number of errors of commission (false positives) is expected to increase. The final threshold for HEC algorithms is $n = 2$ (line 2.8), since a “group” of one node provides no rationale to believe there are any other hot nodes nearby.

4.3 Happiness absorption clustering (HAC)

Local absorption aims at extending a node’s limited communication range by absorbing knowledge from its neighbors. In the happiness absorption clustering (HAC) algorithm, this knowledge is again simple neighbor counts, i.e. the number of hot nodes within communication range. This time, however, nodes pass their local counts on to their neighbors, and after a limited number of hops, the aggregated knowledge is analyzed and used to compute expectations about the presence or absence of clusters, and a node’s happiness respectively.

Algorithm 3 begins as for the HEC algorithm, resorting to the naïve cluster algorithm if the communication range is high enough to enable nodes to expect to see entire clusters (line 3.1–3.3). Otherwise, for each node $v$ a new parameter $z_v$ is initialized as that node’s initial happy value (i.e., 1 if they are hot, 0 otherwise, line 3.6).

Next, nodes communicate and aggregate their neighbors’ $z_v$ values for a number of hops (lines 3.8–3.9). The number of hops depends solely on the ratio of communication range $r$ and the cluster radius $p$. Each time the (multi-hop) communication range drops below cluster radius, an extra hop is added in order to make sure that again an area of at least the cluster extent $r^2 \pi$ is used for information collection and for reasoning about presence or absence of a pattern. If, for example, $p > r > \frac{p}{2}$, one additional hop is added in order to cover the whole cluster (see Figure 15).
Algorithm 3: HAC: Local absorption algorithm to check for each node whether it can locally infer a cluster from its multi-hop neighborhood

<table>
<thead>
<tr>
<th>Line</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>Set $x \leftarrow n \cdot t \cdot \frac{c^2}{r^2}$;</td>
</tr>
<tr>
<td>3.2</td>
<td>if $x \geq n$ then</td>
</tr>
<tr>
<td>3.3</td>
<td>Use NC algorithm (naïve cluster) to determine whether cluster is detected;</td>
</tr>
<tr>
<td>3.4</td>
<td>else</td>
</tr>
<tr>
<td>3.5</td>
<td>foreach $v \in V$ do</td>
</tr>
<tr>
<td>3.6</td>
<td>$z_v \leftarrow s(v)$;</td>
</tr>
<tr>
<td>3.7</td>
<td>for $i = 1$ to $\lceil \frac{c}{r} \rceil$ do</td>
</tr>
<tr>
<td>3.8</td>
<td>foreach $v \in V$ do</td>
</tr>
<tr>
<td>3.9</td>
<td>$z_v \leftarrow z_v + \sum_{v' \in \text{nbr}(v)} z_{v'}$;</td>
</tr>
<tr>
<td>3.10</td>
<td>foreach $v \in V$ do</td>
</tr>
<tr>
<td>3.11</td>
<td>if $z_v / d(c, r, n) &gt; n$ then</td>
</tr>
<tr>
<td>3.12</td>
<td>Node $v$ has detected a cluster of radius $r$ and size $n$;</td>
</tr>
</tbody>
</table>

Using this happiness absorption method, nodes within clusters accumulate counts of hot neighbors, then neighbors’ hot neighbors, and so on. Nodes not within clusters accumulate many fewer hot neighbors counts, if any. Because this procedure will count shared neighbor nodes more than once, the final node count needs to be discounted when deciding if a node has actually detected a cluster. For example, Figure 6 shows a two-hop constellation with several overlapping communication ranges. Instead of the actual 20 two-hop hot neighbors, double-counts mean the central node in fact observes 47 hot nodes. Consequently, a heuristic discount function $d$ is used (line 3.11) to allow for the expected number of double counts. The discount function may depend on a number of factors, including the communication range $c$, and the cluster radius $r$ and size $n$. As for the threshold $t$, the discount function can be empirically determined.

5 Experiments

This section describes the results of experiments to compare the performance of the three algorithms, NC, HEC, and HAC, for decentralized detection of node clusters. The experiments were conducted using a popular free and open-source agent-based simulation and modeling toolkit, called Repast. Repast is implemented in several languages and features various libraries for simulation, visualization, and analysis. Each of the three algorithms was implemented in Repast, with sensor nodes modeled as agents (see Figure 7).
5.1 Design

For each set of experiments, 1000 nodes were located in the square simulation space (set to have side length 1 unit). 10 non-overlapping, but otherwise randomly located clusters of nodes were also generated. Each cluster consisted of 10 hot nodes with a cluster radius of 0.05 units. A further 100 hot nodes were also randomly distributed outside the clusters in order to reach a total of 200 hot nodes. Finally, a further 800 cold nodes were randomly distributed in the simulation space. Cold nodes were allowed to be located anywhere, including within existing hot clusters.

A set of experiments was then run, each experiment varying the communication range $r$, starting from $c = 2r$ and decreasing step by step to 0. At each step the performance of the three algorithms was recorded. Performance was measured in terms of errors of omission (clusters that were placed in the simulation but not detected) and errors of commission (non-clusters that were incorrectly classified by a node as clusters) for each of the three algorithms. Errors of omission and commission were recorded against individual nodes (e.g., whether every node in a cluster correctly detected it was part of a cluster or not) as well as against individual clusters (e.g., whether some node in a cluster correctly detected it was part of a cluster). As discussed previously in section 4.1, in the context of a distributed system it is the latter measure that is more important and so this measure is used in the following discussion of results.

5.2 Results

Figure 8 presents the results for the average performance of the NC, HEC, and HAC algorithms over several simulations. The $x$-axis represents the ratio of communication range $c$ to cluster radius $r$, decreasing step-wise from $c = 2r$, through $c = r$, to $c = 0$. The $y$-axis shows error of commission (EOC, false positives) and error of omission (EOO, false negatives) expressed in number of clusters. As a consequence, when algorithms are performing well they will have zero...
Figure 7: **Implementation in Repast.** The framework features a map view (communication ranges as gray circles, clustering nodes connected with edges), error plots (for errors of omission and commission), and a system log window.

or low corresponding values on the $y$-axis, and conversely high values when performing badly.

The figure shows that the NC algorithm performs near-perfectly when the communication range is strictly larger than the cluster radius, in the range of 2–1.25 for the $c/r$ ratio. However, below 1.25 for the $c/r$ ratio, the performance of the NC algorithm degrades rapidly. This result is to be expected since at such high communication radii individual nodes can be expected to be able to detect entire clusters in their immediate neighborhood.

Since the HEC and HAC algorithms revert to the NC algorithm for larger $c/r$ ratios, these algorithms similarly perform well in the 2–1.25 $c/r$ ratio range. However, the HEC algorithm exhibits a clear improvement in performance over the NC algorithm, exhibiting on average less than two errors of omission or commission, in the 1.25–0.75 $c/r$ ratio range. The local extrapolation adopted by HEC helps to extend it’s operating range beyond that of the NC algorithm. Below the 0.75 $c/r$ ratio range, HEC algorithm performance also degrades rapidly, mirroring the fact that with decreasing $c/r$ ratio, there is a greater chance that small groups of 2 or 3 hot nodes can falsely trigger cluster detection. The HAC algorithm exhibits further improvement on the HEC algorithm, finally degrading at beyond about 0.4 $c/r$ ratio mark. The local absorption used in HAC is able to extend the range of the cluster detection into the zone where individual nodes only detect a very small number of neighbors, and so where HEC fails.
Figure 8: **Errors of omission and commission for NC, HEC, and HAC algorithms.**

## 6 Discussion and conclusions

The experiments in the previous section provide specific examples of *decentralized* algorithms for spatial data mining. The three algorithms presented illustrate how increasingly sophisticated DSDM algorithms can be designed to deliver step improvements in performance. For the specific example of clustering, a naïve decentralized algorithm is bettered by a local extrapolation algorithm, which in turn is outperformed by a local absorption algorithm. Current work is also investigating the further improvements that can be gained from using selective collaboration and, in cases where nodes are mobile, mobility memory and diffusion.

Because distributed spatial data mining is often focused on efficient but approximate algorithms (that may not match the solution generated using an exact centralized algorithm (Datta *et al.* 2006a), the performance of DSDM algorithms is primarily measured in terms of the certainty of its outcomes. Hence, we use the errors of omission and commission to assess our algorithms’ performance. The results indicated the range of conditions under which the algorithms generate reliable results, and those where the algorithm’s performance degrades. Different application domains may have different requirements for DSDM algorithm performance. For example, in safety critical applications, like for example volcano monitoring, it may be vital...
never to miss a salient event (e.g., Werner-Allen et al. 2006). In such applications, approximate DSDM algorithms can still be useful if configured to guarantee no errors of omission, since a small number of errors of commission can be filtered out by additional scrutiny (e.g., human expertise).

The primary advantage of tolerating errors of omission and commission is computational. DSDM algorithms are computationally efficient and as a result highly scalable. Two of the three algorithms used in the experiments above (NC and HEC) use only one one-hop communication, while the third (HAC) uses a small number of hops (up to three hops in practice). This dependence on local rather than global knowledge is what gives these algorithms scalability, because their computational complexity depends not on the total number of nodes in the network, rather on the number of neighbors a node has (which is expected to remain constant as size of the network increases, as long as node density remains constant). This contrasts strongly with centralized algorithms, where computational complexity typically increases with the number of observations (equivalent to nodes in the system). In moving from today’s geosensor networks of tens or hundreds of nodes, to the predicted future networks of thousands or millions of nodes, scalability is paramount.

While centralized algorithms will long remain a core topic in knowledge discovery, DSDM represents a new approach to geographic knowledge discovery and, as we have shown, comes with new challenges beyond those posed by centralized algorithms. New technologies that blur the traditional separation between data capture and data processing (like geosensor networks) are driving the exploration of decentralized processing of spatial data. Longer-term, the promise of these techniques is to contribute to the development of what is sometimes termed ambient spatial intelligence: spatial data capture, processing, and actuating capabilities embedded throughout our natural and built environment.

7 Outlook: A DSDM research agenda

To conclude, we identify four main research and development topics in DSDM:

- The development of a library of fundamental DSDM algorithms for decentralized computation of classic spatial data mining tasks, including clustering, spatial outlier detection, co-location mining, and spatial association rule mining;
- The investigation of robust and fault-tolerant methods for implementing DSDM in notoriously error-prone WSN environments;
- The exploration of DSDM in mobile decentralized spatial computing systems, for example in the domain of traffic management or LBS; and
- The exploitation of decentralization as a means for providing geographic knowledge discovery at the same time as enhancing the location privacy of individuals in scenarios where nodes are associated with human users, for example in traffic or LBS applications.
Acknowledgements

Patrick Laube and Matt Duckham’s work is funded by the Australian Research Council (ARC), Discovery grant DP0662906. Patrick Laube’s research is additionally funded by the ARC Research Network on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP). Patrick Laube furthermore thanks organizers and attendees of the GADGET Workshop on Geometric Algorithms and Spatial Data Mining, funded by the Netherlands Organisation for Scientific Research (NWO) under BRICKS/FOCUS grant number 642.065.503, for an inspiring workshop.

References


