Mining Group Movement Patterns for Tracking Moving Objects Efficiently

Hsiao-Ping Tsai, *Member*, *IEEE*, De-Nian Yang, and Ming-Syan Chen, *Fellow*, *IEEE*

Abstract—Existing object tracking applications focus on finding the moving patterns of a single object or all objects. In contrast, we propose a distributed mining algorithm that identifies a group of objects with similar movement patterns. This information is important in some biological research domains, such as the study of animals' social behavior and wildlife migration. The proposed algorithm comprises a local mining phase and a cluster ensembling phase. In the local mining phase, the algorithm finds movement patterns based on local trajectories. Then, based on the derived patterns, we propose a new similarity measure to compute the similarity of moving objects and identify the local group relationships. To address the energy conservation issue in resource-constrained environments, the algorithm only transmits the local grouping results to the sink node for further ensembling. In the cluster ensembling phase, our algorithm combines the local grouping results to derive the group relationships from a global view. We further leverage the mining results to track moving objects efficiently. The results of experiments show that the proposed mining algorithm achieves good grouping quality, and the mining technique helps reduce the energy consumption by reducing the amount of data to be transmitted.

Index Terms—Distributed clustering, similarity measure, object tracking, WSN.

1 Introduction

Recent advances in location-acquisition technologies, such as global positioning systems (GPSs) and wireless sensor networks (WSNs), have fostered many novel applications like object tracking, environmental monitoring, and location-dependent service. These applications generate large amounts of location data, and many approaches focus on compiling the collected data to identify the repeating movement patterns of objects of interest. The objective is to facilitate the analysis of past movements and estimate future movements, as well as support approximate queries on the original data [17], [34], [38], [48], [52].

In object tracking applications, many natural phenomena show that moving objects often exhibit some degree of regularity in their movements. For example, the famous annual wildebeest migration demonstrates that the movement of creatures is temporally and spatially correlated. In addition, biologists have found that many creatures, such as elephants, zebra, whales, and birds, form large social groups when migrating to find food, or for breeding, wintering, or other unknown reasons. These characteristics indicate that the trajectory data of multiple objects may be correlated. Moreover, some research domains, such as the study of animals' social behavior and wildlife migration [41], [44], are more concerned with a group of animals' movement patterns than each individual's. This raises a new challenge of finding moving animals belonging to the same group and identifying their aggregated movement patterns.

Many researchers model the temporal-and-spatial correlations of moving objects as sequential patterns in data mining, and various algorithms have been proposed to discover frequent movement patterns [17], [33], [34], [37], [48], [52]. However, such works only consider the movement characteristics of a single object or all objects. Other works, such as [35], [50], take the euclidean distance to measure the similarity of two entire trajectories, and then derive groups of mobile users based on their movement data. Since objects may be close together in some types of terrain, such as gorges, and widely distributed in less rugged areas, such as grassland, their group relationships are distinct in some areas and vague in others. Instead of applying global clustering on entire trajectories, examining partial trajectories of individual areas shows more opportunities of revealing the local group relationships of moving objects.

Another motivation for discovering the group relationships and movement patterns behind the trajectories of moving objects, such as monkeys or elephants, is to reduce tracking costs, especially in resource-constrained environments like WSNs. In a WSN, a large number of miniature sensor nodes with sensing, computing, and wireless communication capabilities are deployed in remote areas for various applications, such as environmental monitoring or wildlife tracking. As the sensors are generally battery-powered, recharging a large number of them is difficult; therefore, energy conservation is paramount among all the design issues in WSNs [5]. One important characteristic of WSNs is that sensors are deployed close together to ensure complete coverage of the monitored area. As a result, the

- *H.-P. Tsai is with the Department of Electrical Engineering (EE), National Chung Hsing University, No. 250, Kuo Kuang Road, Taichung 402, Taiwan, ROC. E-mail: hptsai@nchu.edu.tw.*
- *D.-N. Yang is with the Institute of Information Science (IIS) and the Research Center of Information Technology Innovation (CITI), Academia Sinica, No. 128, Academia Road, Sec. 2, Nankang, Taipei 11529, Taiwan, ROC. E-mail: dnyang@iis.sinica.edu.tw.*
- *M.-S. Chen is with the Research Center of Information Technology Innovation (CITI), Academia Sinica, No. 128, Academia Road, Sec. 2, Nankang, Taipei 11529, Taiwan, ROC, and the Department of Electrical Engineering (EE), Department of Computer Science and Information Engineer (CSIE), and the Graduate Institute of Communication Engineering (GICE), National Taiwan University, No. 1, Sec. 4, Roosevelt Road, Taipei 10617, Taiwan, ROC. E-mail: mschen@cc.ee.ntu.edu.tw.*

Manuscript received 20 July 2008; revised 1 Mar. 2009; accepted 8 Aug. 2009; published online 11 Nov. 2009.
Recommended for acceptance by L. Wang.
For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number TKDE-2008-07-0372.
Digital Object Identifier no. 10.1109/TKDE.2009.202.

sensing fields of multiple sensors overlap, so there is likely to be some redundancy in readings of nearby sensors or in a series of readings of a single sensor. Most applications incorporate data aggregation techniques to combine data from multiple sources and filter redundant data; and thereby, reduce the amount of data and—by extension—the energy consumption. If we can find moving animals belonging to the same group and identify their aggregated movement patterns, we can utilize the group relationships in data aggregation to reduce long-distance transmissions, or in wake-up scheduling to keep most sensors asleep. To the best of our knowledge, little research has been dedicated to discovering a group of objects with similar movement patterns for data aggregation purposes. Algorithms for discovering the movement patterns of a specific group of objects in a resource-constrained environment have not been proposed, and related design issues have not been discussed in the literature. Moreover, we find that discovering the movement patterns of a group of objects is more difficult than finding the patterns of a single object because we need to identify a group of objects before or after discovering their movement patterns. To address these problems, we first propose a mining framework that can identify a group of moving objects and discover their group movement patterns in a distributed manner. The discovered information is then used in the design of an efficient tracking network. We show that discovering and exploiting the movement patterns of a group of objects can further reduce the transmission costs and thereby conserve energy.

In the first part of the paper, we present our distributed mining algorithm, which is comprised of 1) a local Group Movement Pattern Mining (GMPMine) algorithm that extracts local group information, and 2) a Cluster Ensembling (CE) algorithm that combines and improves the local grouping results. To address the energy conservation issue in WSNs, the algorithm only transmits the local grouping results to the sink node for further ensembling, instead of all the location data about moving objects. In contrast to approaches that perform clustering on entire trajectories at a central server, the proposed algorithm discovers the group relationship in a distributed manner on sensor nodes. The network structure naturally partitions the trajectories of moving objects into segments, which our algorithm uses to identify the objects' local group relationships. Thus, we can discover the group relationships of objects that may be blurred when analyzing an entire trajectory. As a result, we can aggregate the location data of such objects in areas where they have distinct group relationships. Moreover, our algorithm considers the diversity of the number of groups and their sizes in the tracking applications. Since there are inherent variations in the number of groups and their sizes (e.g., elephant herds may contain 8-100 individuals, depending on the environment and family size [1]), it is difficult to predetermine these two parameters. Therefore, we use the HCS algorithm [21] to cluster objects efficiently without prespecifying the number of groups or their sizes.

In the second part of the paper, we leverage the mining results to reduce the number and size of packets in the proposed tracking network, which consistently reports the locations of moving objects. We utilize the discovered group information to combine the location data of a group of objects in data aggregation. By using object movement patterns as the prediction model, we do not need to send the update packets for the objects whose locations are predictable. Moreover, the group information enables us to adaptively adjust the range in which a group of objects is monitored and, thereby, limit the overhead due to flooding within that range. Therefore, the proposed design reduces long-distance transmissions and the amount of data to be transmitted. It also reduces in-network traffic and prevents hot spots around the clusterheads (CHs).

The contribution of this paper is threefold. First, we propose a distributed mining framework to discover group relationships as well as group movement patterns. Second, we propose a new pairwise measure based on pattern similarity to compute the similarity of moving objects. Third, we use the discovered information to track moving objects efficiently. The remainder of this paper is organized as follows: In Section 2, we review related works, provide an overview of our network model and location model, and define the notations used throughout the paper. In Section 3, we describe the design of our distributed mining algorithm. Section 4 considers the design aspects of our tracking network, and Section 5 details our experimental results. Then, in Section 6, we summarize our conclusions.

2 Preliminaries

2.1 Related Work

2.1.1 Movement Pattern Mining

The temporal-and-spatial correlations and the regularity in the trajectory data sets of moving objects are often modeled as sequential patterns for use in data mining. Agrawal and Srikant [4] first defined the sequential pattern mining problem and proposed an Apriori-like algorithm to mine frequent sequential patterns. Han et al. proposed FreeSpan [20], which is an FP-growth-based algorithm that addresses the sequential pattern mining problem by considering the pattern-projection method. For handling the uncertainty in trajectories of mobile objects, Yang and Hu [52] developed a new match measure and proposed TrajPattern to mine sequential patterns from imprecise trajectories. Moreover, a number of research works have been elaborated upon mining traversal patterns for various applications. For example, Chen et al. [13] proposed the FS and SS algorithms for mining path traversal patterns in a Web environment while Peng and Chen [37] proposed an incremental algorithm to mine user moving patterns for data allocation in a mobile computing system. However, sequential patterns or path traversal patterns do not provide sufficient information for location prediction or clustering. The reasons are as follows: First, for sequential pattern mining or path traversal pattern mining extract frequent patterns of all objects, meaningful movement characteristics of individual objects may be ignored. Second, a sequential pattern or a traversal pattern carries no time information between consecutive items, so they cannot provide accurate information for location prediction when time is concerned. Third, sequential patterns are not full representative to individual trajectories because a sequential pattern does not contain the information about the number of times it occurs in each individual trajectory. To discover significant patterns for location prediction, Morzy proposed Apriori-Traj [33] and Traj-PrefixSpan [34] to mine frequent trajectories, where consecutive items of a frequent trajectory are also adjacent in the original trajectory data. Meanwhile,

the approach in [17] extracts T-patterns from spatial temporal data sets to provide concise descriptions of frequent movements. Tseng and Lin [48] proposed the TMP-Mine algorithm for discovering the temporal movement patterns of objects. However, the above Apriori-like or FP-growth-based algorithms suffer from computing efficiency or memory problems, which make them unsuitable for use in resource-constrained environments. In addition, they focus on discovering frequent patterns of all objects.

2.1.2 Trajectory Clustering

Recently, clustering based on objects' movement behavior has attracted more attention. For example, Li et al. [30] employ Moving Microclusters (MMC) to discover and maintain a cluster of moving objects online. Meanwhile, Lee et al. [28] proposed trajectory clustering to discover popular movement paths. Clustering similar trajectory sequences to discover group relationships is closely related to our problem. Wang et al. [50] transform the location sequences into a transaction-like data on users and based on which to obtain a valid group. However, the proposed AGP and VG-growth algorithms are Apriori-like or FP-growth-based algorithms that suffer from high computing cost and memory demand. Nanni and Pedreschi [35] apply a density-based clustering algorithm to the trajectory clustering problem based on the average euclidean distance of two trajectories. However, the above works that discover group information based on the proportion of the time a group of users stay close together or the average euclidean distance of the entire trajectories may not reveal the local group relationships, which are required for many applications.

2.1.3 Similarity Measure

Identifying the similarity (or distance) between two trajectories is essential for clustering. Computing the average euclidean distance of two geometric trajectories is a simple and useful approach. Nevertheless, the geometric coordinates are expensive and not always available. Other approaches, such as EDR, LCSS, and DTW, are widely used to compute the similarity of symbolic sequences [12]. However, the above dynamic programming approaches suffer from scalability problem [19]. Therefore, approximation or summarization techniques are used to represent original data for providing scalability. For example, Guralnik and Karypis [19] represent each data sequence by a vector whose dimensions are the sequential patterns, and use a vector-based K-means algorithm to find the clusters of objects. But there are two issues of this approach. First, as mentioned previously, sequential patterns are unsuitable for our applications. Second, since meaningful patterns are underpruned if the minimum support threshold is not adequate, the importance of a sequential pattern regarding different sequences can be very different. However, when projecting each data sequence into a vector space of sequential patterns, the importance of a sequential pattern regarding to each data sequence is not mentioned. In addition, Yang and Wang [53] employ a probabilistic suffix tree to learn the structural features of sequences and proposed a new similarity measure which computes the similarity of a probabilistic suffix tree and a sequence. Their clustering algorithm iteratively identifies a sequence to a cluster and adjusts the representative probabilistic suffix tree for each cluster. However, the generated clusters may overlap which differentiates their objective from ours. In addition, once the tree or the sequence is changed, recalculating their similarity requires repeating the whole similarity computing process, which is not desirable for resource-constrained or streaming environments.

2.1.4 Distributed Clustering

Distributed clustering is an important research topic. Most of the approaches proposed in the literature focus on seeking a combination of multiple clustering results to achieve better clustering quality, stability, and scalability. For example, Strehl and Ghosh [46] introduced and formulated the clustering ensemble problem to a hypergraph partitioning problem, and proposed CSPA, HGPA, and MCLA to compute the best K-partition of the graph. Ayad et al. [8] presented a probabilistic model to combine cluster ensembles by utilizing information theoretic measures. Fred and Jain [16] combine multiple runs of the K-means algorithm with random initializations and random numbers to obtain the final consensus partition. Fern and Brodley [15] apply random projection to the high dimensional data and cluster the reduced data by using EM for a single run of clustering. The Collective PCA technique [24] proposed by Kargupta et al. is applied to reduce the vector dimension for distributed clustering of high dimensional heterogeneous data. However, since the trajectory data set is composed of sequential data, one of the challenges addressed in our paper is the similarity comparison of location sequences. The data types of the above works [8], [15], [16], [24], [46] are most integer vector or categorical data, and their related issues are thereby different from ours. In addition, previous works that require a predetermined k in their clustering or ensembling algorithms are not suitable for our applications. Besides, although the local grouping results in a vector of integers, each of which represents the mapping between an object and its belonging group, dimension reduction like Collective PCA [24] is unnecessary in our case.

2.2 Hierarchical Network Structure and Location Model

Many researchers believe that a hierarchical architecture provides better coverage and scalability, and also extends the network lifetime of WSNs [47], [56]. In a hierarchical WSN, such as that proposed in [36], the energy, computing, and storage capacity of sensors are heterogeneous. A high-end sophisticated node, such as Intel Stargate [3], is assigned as a CH to perform high complexity tasks; while a resource-constrained node, such as Mica2 mote [2], performs the sensing and low complexity tasks. In this work, we adopt a hierarchical and cluster-based network structure with K layers. As shown in Fig. 1a, the nodes are clustered in each level, and each cluster is a mesh network of fixed size, i.e., each cluster is a set of $n \times n$ sensors. We assume that each sensor in a cluster has a locally unique ID, and denote the sensor IDs by an alphabet Σ. Fig. 1b shows an example of a two-layer network structure, where each cluster contains 16 nodes whose IDs are identified by $\Sigma = \{a, b, \ldots, p\}$.

A cluster of sensors communicate with each other by using multihop routing, and wake up on their duty cycles to carry out a given task [6]. They collaboratively gather or relay remote information to a base station called a sink. Take the tracking application for example. When a sensor

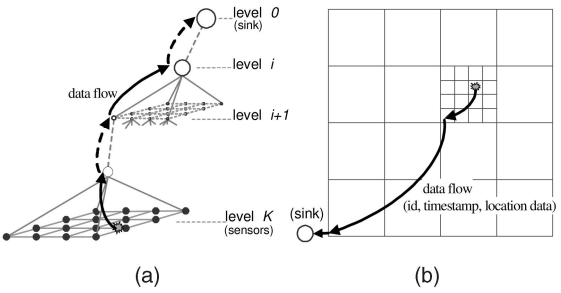


Fig. 1. (a) The hierarchical and cluster-based network structure and the data flow of an update-based tracking network. (b) A flat view of a two-layer tracking network.

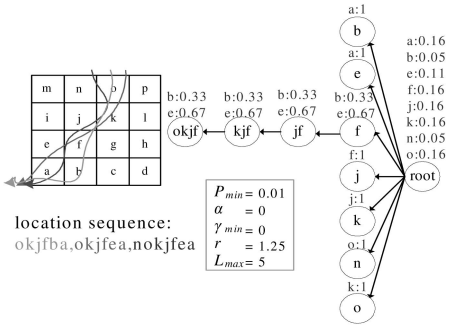


Fig. 2. The movement of an object and its PST.

wakes up and detects an object of interest, it transmits the location data of the object to its CH and then enters the sleep mode. The CH aggregates the data and forwards it to the CH of the upper layer. The process is repeated until the sink node receives the location data. The data flow is as shown in Fig. 1. When a task of discovering the group relationships of objects is assigned, it is unnecessary to transmit all the location data to the sink for postprocessing. In our design, CHs collect the location data for a period and generate location sequence data sets locally. Then, based on the data sets, our mining algorithm tries to discover the group relationships about the objects of interest.

Geometric models and symbolic models are widely used to represent the location of objects [22]. A geometric location denotes precise two-dimension or three-dimension coordinates; while a symbolic location represents an area, such as the sensing area of a sensor or a group of sensors, defined by the applications. Since the accurate geometric location is not easy to obtain, in this work, we employ a symbolic model and take the sensors' IDs as the locations of an object of interest. Sensors are closely deployed to ensure complete coverage of the monitored area, but this causes consistency and redundancy problems. Techniques like the Received Signal Strength (RSS) [29] simply estimate an object's location based on the ID of the sensor with the strongest signal and eliminate unnecessary transmissions. The trajectory of a moving object is thus modeled as an ordered sequence of sensor IDs, i.e., a location sequence denoted by $s = \sigma_0\sigma_1...\sigma_{l-1}$, where $\sigma_i \in \Sigma$ and l is the sequence length.

2.3 Variable Length Markov Model (VMM) and Probabilistic Suffix Tree (PST)

If the movement of an object is regular, the object's next location can be predicted based on its preceding locations. We leverage the temporal-and-spatial correlations of the moving object and use a VMM to model the statistics. Under this model, an object's movement is modeled by the conditional probability distribution over Σ for a given location sequence data set. Specifically, for a location sequence s over Σ and a symbol $\sigma \in \Sigma$, $P(\sigma|s)$ is the conditional probability that σ will follow s. The length of s is floating, which provides the flexibility to adapt to the variable length of movement patterns.

Among many implementations of VMM, PST [40] has the lowest storage requirement [25]. The PST building algorithm[1] extracts significant patterns from a data set, prunes unnecessary nodes during tree construction, and then generates a PST. Each node in the tree is labeled by a string s, which represents a significant pattern with occurrence probability above the minimal threshold P_{min}. Each node carries the conditional empirical probabilities, i.e., $P(\sigma|s)$, for every $\sigma \in \Sigma$, and the maximal length of s is specified by L_{max}. For example, as shown in Fig. 2, $node_{jf}$ is a child of $node_f$ with respect to symbol j. It represents a significant pattern with $s = "jf"$, whose occurrence probability is above 0.01. The conditional empirical probabilities are $P('b'|"jf") = 0.33$, $P('e'|"jf") = 0.67$, and $P(\sigma|"jf") = 0$ for the other $\sigma \in \Sigma$.

The PST algorithm has an excellent capacity for extracting structural information from sequences. Its low complexity, i.e., $O(l)$ in both time and space [7], makes it more attractive to be used in streaming or resource-constrained environments. Compared with algorithms that mine all accurate frequent patterns, the compact tree structure and the controllable size of a PST are particularly useful in resource-constrained environments. For example, if the conditional probabilities of a pattern "ABCD" are similar to those of "BCD," PST will only store "BCD." Moreover, PST is efficient in predicting the occurrence probability of a sequence or predicting the next symbol of a sequence. The occurrence probability of a sequence s regarding to a PST T, denoted by $P^T(s)$, is the prediction of the occurrence probability of s based on T. For example, given a PST T, as shown in Fig. 2, the occurrence probability $P^T("nokjfb")$ is computed as follows:

$$\begin{aligned} P^T("nokjfb") &= P^T("n") \times P^T('o'|"n") \times P^T('k'|"no") \\ &\quad \times P^T('j'|"nok") \times P^T('f'|"nokj") \\ &\quad \times P^T('b'|"nokjf") \\ &= P^T("n") \times P^T('o'|"n") \times P^T('k'|"o") \\ &\quad \times P^T('j'|"k") \times P^T('f'|"j") \times P^T('b'|"okjf") \\ &= 0.05 \times 1 \times 1 \times 1 \times 1 \times 0.33 \\ &= 0.0165. \end{aligned}$$

For a given sequence s and a PST T, our predict_next algorithm[2] outputs the next most likely symbol σ. We explain the algorithm by an example to demonstrate the efficiency of the algorithm. Given a sequence $s = "nokjf"$

1. Given in Appendix 1, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TKDE.2009.202.

2. Given in Appendix 2, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TKDE.2009.202.

TABLE 1
Description of the Notations and Key Symbols

Notation/Symbol	Description	Notation/Symbol	Description
$O = \{o_0, o_1, \ldots, o_{N-1}\}$	A set of N objects	D	A set of thresholds (CE)
$G = \{g_0, g_1, \ldots, g_{m-1}\}$	A grouping result containing m groups of objects	δ	A threshold in the range $[0,1]$
$C = \{G_0, G_2, \ldots, G_{\mathcal{K}-1}\}$	An ensemble of $\mathcal{K}$ local grouping results	$\pi(o_i)$	The identified group ID of object o_i
Σ	A finite alphabet representing IDs of a cluster of sensors	S_{ij}	Jaccard Similarity Coefficient
$s = \sigma_0\sigma_1 \ldots \sigma_{l-1}$	A location sequence of length l, where $\forall\sigma_i \in \Sigma$	TC	The transmission cost
$P(\sigma\|s)$	The empirical conditional probability that σ is right after s	N	The number of objects of interest
T	A PST, T_i is the PST associated with o_i	N_r	The number of random-walk objects
L_{max}	The maximal memory length of a PST	R	The radius of an SG in hops
$P^T(s)$	The predicted occurrence probability of s regarding T	K	Layer number of the network structure
sim_p	The new proposed similarity measure	n	Square root of the size of a sensor cluster
sim_{min}	The minimal similarity threshold (GMPMine)	EB	A specified error bound
$\widetilde{S}$	The union of nodes in two PSTs	GDR	The group dispersion radius

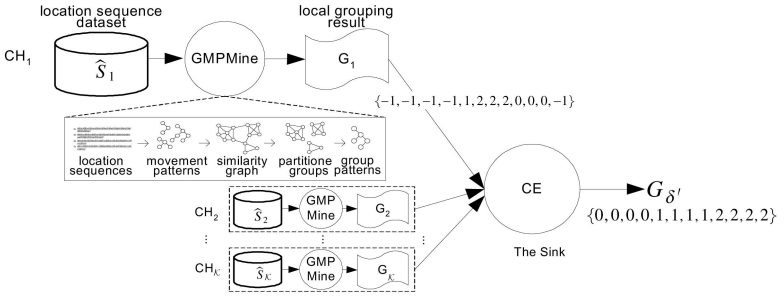


Fig. 3. The framework of our distributed mining algorithm: At the local end, CH_i performs the GMPMine algorithm to generate local grouping result G_i while the sink performs the Cluster Ensembling algorithm to combine the local grouping results into a consensus final result $G_{\delta'}$.

and a PST T, as shown in Fig. 2, the predict_next algorithm traverses T to the deepest node $node_{okjf}$ based on s. The path includes $node_{root}$, $node_f$, $node_{jf}$, $node_{kjf}$, and $node_{okjf}$. Finally, symbol 'e', which has the highest conditional empirical probability in $node_{okjf}$, is returned as the most probable next symbol. Since the algorithm's computational overhead is limited by the height of a PST, it is suitable for sensor nodes. Further details about the PST building algorithm and related discussions about parameter setting can be found in [9], [27]. The notations and key symbols used in the paper are listed in Table 1.

3 Design of the Distributed Mining Algorithm

In this work, we model the movement of an object by a VMM, and use a PST to mine the significant movement patterns. The advantages of PST include its computing and storage efficiency as well as the information it carries. In the tracking application, objects are tracked periodically so that the time interval of consecutive items of a location sequence is implied. The PST building algorithm scans the sequence for significant movement patterns, whose items are constrained to be consecutive in the location sequence. This is also why the computing cost is much lower than sequence pattern mining. Moreover, a PST provides us important information in similarity comparison. For a pattern and a PST, we can predict the occurrence probability of the pattern, which is viewed as the importance of the pattern regarding the PST.

A set of moving objects is regarded as belonging to the same group if they share similar movement patterns. In this section, we first propose a new similarity measure to define the pairwise similarity of moving objects. The advantages of the new proposed similarity measure sim_p include its efficiency and its accuracy. First, sim_p compares the similarity of two objects based on their significant movement patterns instead of their entire location sequences. In [54], a variation of PST, named emission tree, is used to train the patterns in a streaming environment. sim_p can be directly applied to the mature nodes of two emission trees, instead of all nodes. Thus, sim_p can provide efficiency for the applications with evolving and evolutionary similarity relationships.[3] Second, it considers the importance of each movement pattern regarding to each individual object so that it achieves better accuracy in similarity comparison.

With the definition of sim_p,[4] two objects are similar if their similarity score is above a minimal threshold. A set of objects is regarded as a group if each object is similar to at least half the members of the same group. To tackle the problem of discovering groups of moving objects, we propose a distributed mining algorithm comprised of a GMPMine algorithm and a CE algorithm as shown in Fig. 3. The GMPMine algorithm uses a PST to generate the

3. Discussion about continually refining of a PST is given in Appendix 3, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TKDE.2009.202.

4. The definition is given in Section 3.1.

significant movement patterns and computes the pairwise similarity of moving objects by using sim_p. It utilizes the HCS algorithm to cluster the moving objects into non-overlapped groups. To address the energy consumption problem in WSNs, our algorithm only transmits the local grouping result G_i to the sink. For $\mathcal{K}$ clusters, at most $\mathcal{K}$ vectors whose dimension is equal to the number of objects are transmitted. The sink then applies the CE algorithm to combine the local grouping results. The CE algorithm utilizes the Jaccard similarity coefficient to measure the similarity between a pair of objects, and normalized mutual information (NMI) to derive the final ensembling result $G_{\delta'}$.

When the group information of moving objects in a specified subregion is of interest, our design can be easily extended to emphasize the importance of specified areas. It can also combine local grouping results from sensor clusters with heterogeneous tracking configurations, such as different monitoring intervals, or different network structures of sensor clusters, which can reduce the tracking costs. For example, instead of waking up all sensors at the same frequency, a shorter tracking interval is specified for some types of terrain, such as gorges, in the migration season to reduce energy consumption. Rather than deploying the sensors in the same density, they are only highly concentrated in areas of interest in order to reduce deployment costs. In contrast to other clustering algorithms, such as K-means, which partition objects into a predetermined number of groups, we consider the diversity of the number of groups and their sizes in the tracking applications. In addition, we trade off the grouping quality against the computation cost by adjusting the partition parameter of the CE algorithm.

3.1 Similarity Measurement

In this work, we use a PST to mine significant movement patterns of an object, where a significant movement pattern is a subsequence with occurrence probability higher than a minimal threshold. Each node of a PST represents a significant movement pattern and carries its conditional probabilities, and all nodes of a PST provide the precise information about the predicted occurrence probability of a given pattern.

To provide better discrimination accuracy, we propose a new similarity measure sim_p that adequately and skillfully utilizes the information carried by PSTs to measure the similarity of two objects. The design concepts of sim_p are simple and useful. Different from the previous works, such as [19], that equal weights the patterns, we further consider importance and difference of a pattern related to each individual object. The importance of a pattern s is modeled by using the predicted occurrence probability, i.e., $P^T(s)$, while the difference of a pattern is defined over all of the dimensions, i.e., $P^T(\sigma|s)$, $\forall \sigma \in \Sigma$. Based on the two concepts, we define the distance a pattern s associated with two objects o_i and o_j as

$$\begin{aligned} d(s) &= \sqrt{\sum_{\sigma\in\Sigma}(P^{T_i}(s\sigma) - P^{T_j}(s\sigma))^2} \\ &= \sqrt{\sum_{\sigma\in\Sigma}(P^{T_i}(s) \times P^{T_i}(\sigma|s) - P^{T_j}(s) \times P^{T_j}(\sigma|s))^2}, \end{aligned} \quad (1)$$

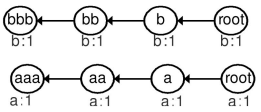


Fig. 4. The maximal value of $\sum_{s\in\widetilde{S}} d(s)$ of the two PSTs is $6+\sqrt{2}$.

where T_i and T_j are their respective PSTs. $d(s)$ is the euclidian distance of products of the importance and difference over Σ related to o_i and o_j. Note that since the similarity of two objects is symmetric in our applications, a symmetric measure, such as euclidean distance, is more desirable. Furthermore, for a pattern $s \in T_i$, $P^{T_i}(s)$ is a significant value because the occurrence probability of s is higher than the minimal support P_{min}. If o_i and o_j share the pattern s, we have $s \in T_i$ and $s \in T_j$, respectively, such that $P^{T_i}(s)$ and $P^{T_j}(s)$ are non-negligible and meaningful in the similarity comparison. Thus, we define the similarity score of o_i and o_j by using all of their significant movement patterns as follows:

$$sim_p(o_i, o_j) = -log\frac{\sum_{s\in\widetilde{S}}\sqrt{\sum_{\sigma\in\Sigma}(P^{T_i}(s\sigma) - P^{T_j}(s\sigma))^2}}{2L_{max}+\sqrt{2}}, \quad (2)$$

where $\widetilde{S}$ denotes the union of significant movement patterns (nodes) of both objects (trees). We sum $d(s)$ for all $s \in \widetilde{S}$ as the distance between two PSTs and normalize it by its maximal value, i.e., $2L_{max}+\sqrt{2}$. For example, the PSTs shown in Fig. 4 are built for two sequences $''aa\ldots a''$ and $''bb\ldots b''$ with $\alpha = 0$, $r = 1$, and $L_{max} = 3$. The distance between two PSTs is $6+\sqrt{2}$, where $\widetilde{S} = \{''a'', ''aa'', ''aaa'', ''b'', ''bb'', ''bbb''\}$. Thereafter, we take the negative log of the distance between two PSTs as the similarity score such that a larger value of the similarity score implies a stronger similar relationship, and vice versa.

The worst-case complexity of computing sim_p is $O(L_{max} \times |\Sigma| \times |\widetilde{S}|)$. Therefore, the complexity of building a PST and scoring the similarity between two PSTs is $O(L_{max} \times |\Sigma| \times |\widetilde{S}| + l)$, where l is the sequence length.

3.2 The GMPMine Algorithm

We now describe the GMPMine algorithm, which identifies groups of objects and determines their movement patterns. Fig. 5 illustrates the GMPMine algorithm, where $\widehat{S}$ represents the location sequence data set and N denotes the number of objects of interest. The minimal similarity threshold (sim_{min}) is the lower limit of the similarity between two objects belonging to the same group. Let $O = \{o_0, o_1, \ldots, o_{N-1}\}$ denote the objects of interest and $\pi(o_i)$ denote the mapping of the group ID and object o_i. The GMPMine algorithm generates the grouping result G and the associated group movement patterns GT. Specifically, G is composed of m disjoint groups of objects over O, denoted by $G = \{g_0, g_1, \ldots, g_{m-1}\}$, where $g_i = \{o_j|\pi(o_j) = i, o_j \in O\}$. The group movement patterns associated g_i is denoted by GT_i, and $GT = \{GT_0, GT_1, \ldots, GT_{m-1}\}$ denotes the group movement patterns for the m groups.

The GMPMine algorithm is comprised of four steps. First, we extract the movement patterns of each object from the location sequence. Second, we construct a similarity

Algorithm: GMPMine
Input: $\hat{S} = \{ s_i | 0 \le i < N \}$, sim_{min}, P_{min}, α, γ_{min}, r, L_{max}, Σ
Output: G, m, GT

0. $G = \emptyset$
1. $m = 0$
2. /*building a PST for each object and pruning noise*/
3. **for** each s_i in $\hat{S}$
4. T_i = PST_Build(s_i, P_{min}, α, γ_{min}, r, L_{max}, Σ)
5. /*constructing a similarity graph on PSTs*/
6. **for** 0 $\le$ i < N-1
7. **for** i+1 $\le$ j < N
8. **if** $\text{sim}_p(T_i, T_j) > sim_{min}$ **then**
9. add_edge(i,j) to $Graph(V, E)$
10. /*extracting highly connected subgraph*/
11. (G,m) = HCS($Graph(V, E)$) // G={$g_i | 0 \le i < m$}
12. /* selecting a group PST GT_i for each group g_i */
13. **for** $0 \le i < m$
14. $S' = \{s_j | o_j \in g_i, 0 \le j < N\}$
15. $T' = \{T_j | o_j \in g_i, 0 \le j < N\}$
16. $GT_i = \arg\max_{T_j} \sum_{s_k \in S'} P_j(s_k)$ $where$ $T_j \in T'$
17. **return** $G = \{g_i | 0 \le i < m\}$, $GT = \{GT_i | 0 \le i < m\}$

Fig. 5. The GMPMine algorithm.

graph in which similar objects are connected by an edge. Third, we extract highly connected components to derive the group information. Fourth, we construct a group PST for each group in order to conserve the memory space. In the next section, we describe the steps in detail.

3.2.1 Building PSTs for All Objects

In this step, we learn the movement patterns and construct a PST for each object. As shown in Lines 3 and 4 of Fig. 5, for the location sequence data set with N location sequences, we compute the movement patterns and generate N PSTs. For example, Fig. 6a shows the trajectories of three groups, each of which contains four objects. The mesh network is a two-layer structure with four clusters, each containing four sensors labeled as $\Sigma = \{a, b, c, d\}$. The location sequence data set of CH_a is shown in Fig. 6b. With the PST parameter setting (0.01, 0, 0.0001, 1.3, 5), we show the PSTs of objects o_4, o_5, o_8, and o_9 in Fig. 7 for example.

3.2.2 Constructing a Similarity Graph Based on PSTs

In this step, we compute the similarity of two objects based on their PSTs by using sim_p. We compute the similarity score for each pair of objects and construct an unweighted, undirected similarity graph $G(V, E)$ (Lines 6-9). A vertex in $G(V, E)$ corresponds to an object, and an edge between two objects indicates that their similarity score is above the predefined threshold sim_{min}. Hence, we transform the problem into a graph partition problem.

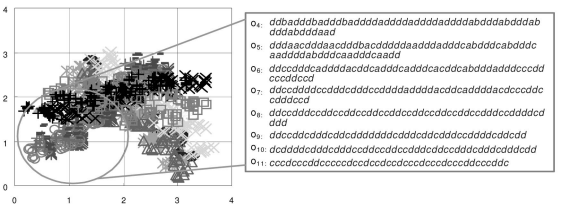


Fig. 6. An example of (a) trajectories of 12 moving objects, and (b) the collected location sequence data set at CH_a.

Let us consider the PSTs of objects o_4, o_5, o_8, and o_9 shown in Fig. 7. According to (2), the similarity score is approximately 1.618 for o_4 and o_5, 1.067 for o_4 and o_9, and 1.832 for o_8 and o_9. With $sim_{min} = 1.2$, we can infer that o_4 and o_5 as well as o_8 and o_9 are similar. Then, we construct the similarity graph, as shown in Fig. 7e.

3.2.3 Extracting Highly Connected Subgraphs

In this step, we partition the similarity graph to generate the grouping result. The properties of the HCS algorithm [21] make it suitable for use in our tracking applications. First, it derives group relationships based on the graph's connectivity without any parameter tuning. Second, the output subgraphs do not need to have the similar number of nodes. Third, the algorithm's polynomial complexity is low, so it is efficient in practice. For the details of HCS, please refer to [21]. As shown in Line 11 of Fig. 5, our algorithm partitions the similarity graph $Graph(V, E)$ and generates the grouping result G. After that, the CH represents the local grouping result by the vector $\{\pi(o_0), \pi(o_1), \pi(o_2), \ldots, \pi(o_{N-1})\}$ and sends it to the sink node. For the example shown in Fig. 7e, the clustering result is shown in Fig. 7f and the vector is $\{-1, -1, -1, -1, 1, 2, 2, 2, 0, 0, 0, -1\}$. Note that $\pi(o_i) = -1$ indicates that the CH does not have enough information to group o_i, so the group relationship of o_i is unknown.

3.2.4 Constructing a Group Probabilistic Suffix Tree

The above steps extract the group information and object movement patterns. In this step, we retain the most representative PST of a group of objects for storage efficiency.

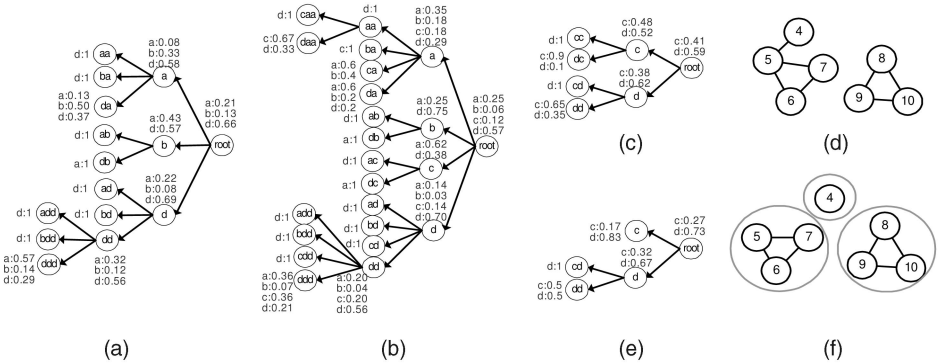


Fig. 7. PSTs of o_4, o_5, o_8, and o_9 over $\Sigma = \{a, b, c, d\}$ and the grouping result at CH_a. (a) PST of o_4. (b) PST of o_5. (c) PST of o_8. (d) PST of o_9. (e) Similarity graph. (f) Highly connected subgraphs.

Let S'/T' denote the set of location sequences/PSTs of the objects in g_i. The formula for selecting the best group PST, GT_i, for g_i is expressed as $GT_i = arg\max_{T_j \in T'} \sum_{s \in S'} P^{T_j}(s)$. Note that the probability $P^T(s)$ will be higher if a sequence s implicitly shares more similar patterns with T. Therefore, we choose the PST with the largest aggregate probability for a group of objects (Lines 13-16 in Fig. 5).

3.3 The CE Algorithm

In the previous sections, each CH collects location data locally and generates group information with the proposed GMPMine algorithm. Since objects may not pass through all the clusters, and the group relationships of objects may vary in different areas, the local grouping results may be inconsistent. For example, if objects walk close together across a canyon, it is reasonable to consider them a group. In contrast, objects scattered in grassland is hardly identified as a group. Furthermore, in the case where a group of objects move across the margin of a sensor cluster, the group relationship is difficult to determine. Therefore, we propose using the CE algorithm to combine multiple local grouping results. The algorithm solves the inconsistency problem and improves the grouping quality.

The ensembling problem involves finding the partition of O that contains the most information about the local grouping results. Let C denote the ensemble of the local grouping results, represented as $C = \{G_0, G_1, \ldots, G_{\mathcal{K}-1}\}$, where $\mathcal{K}$ denotes the ensemble size, i.e., the total number of CHs. The local grouping result G_i obtained from CH_i is a partition of O with m_i disjoint groups, represented as $G_i = \{g_0^i, g_1^i, \ldots, g_{m_i-1}^i\}$. Similar to Strehl and Ghosh [46] and Fred and Jain [16], we utilize NMI to evaluate the grouping quality. The mutual information of G_i and G_j is a measure of the amount of information shared by the two results [14]. It is calculated by

$$MI(G_i, G_j) = \sum_{a=0}^{m_i-1} \sum_{b=0}^{m_j-1} \widehat{P}(a,b) \log \frac{\widehat{P}(a,b)}{\widehat{P}(a) \times \widehat{P}(b)},$$

where $\widehat{P}(a)$ denotes the probability function of G_i, defined as

$$\widehat{P}(a) = \frac{|g_a^i|}{|O|};$$

and $\widehat{P}(a,b)$ is the joint probability distribution function of G_i and G_j, defined as

$$\widehat{P}(a,b) = \frac{|g_a^i \cap g_b^j|}{|O|}.$$

The normalized version of the mutual information of G_i and G_j, denoted by $NMI(G_i, G_j)$, is formulated by

$$NMI(G_i, G_j) = \frac{\sum_{a=0}^{m_i-1} \sum_{b=0}^{m_j-1} \widehat{P}(a,b) \log \frac{\widehat{P}(a,b)}{\widehat{P}(a) \times \widehat{P}(b)}}{\sqrt{H(G_i) \times H(G_j)}}, \quad (3)$$

where $H(G_i)$ is the entropy of G_i, which is a measure of the amount of information in G_i, defined by $H(G_i) = \sum_{a=0}^{m_i-1} -\widehat{P}(a) \times \log \widehat{P}(a)$. Note that a low NMI value indicates that two distributions only have a random association, whereas a higher value indicates that they are mutually informative. For a probable ensembling result G, the summation of NMIs of G and every $G_i \in C$ represents the amount of information that G contains with respect to C. Therefore, the ensembling result G' that contains the most information about C is given by

$$G' = arg\max_{G \in \widetilde{G}} \sum_{i=0}^{\mathcal{K}-1} NMI(G_i, G), \quad (4)$$

where $\widetilde{G}$ denotes all possible ensembling results. However, enumerating every $G \in \widetilde{G}$ in order to find the optimal ensembling result G' is impractical, especially in resource-constrained environments. To overcome this difficulty, we propose the CE algorithm detailed in Fig. 10. The algorithm leverages the information in C to generate the ensembling result $G_{\delta'}$, and trades off the grouping quality against the computation cost by adjusting the partition parameter D, i.e., a set of thresholds with values in the range $[0, 1]$. A finer grained configuration of D achieves a better grouping quality, but the penalty is a higher computation cost.

The proposed CE algorithm is comprised of three steps. First, it utilizes the Jaccard similarity coefficient to measure the similarity of each pair of objects. Second, it partitions the objects for every $\delta \in D$; and third, it employs NMI to optimize the ensembling result. In the following, we describe the three steps in detail.

In the first step, the algorithm measures the similarity of each pair of objects to construct a similarity matrix based on the local grouping results, as shown in Lines 5-7 of Fig. 10. In tracking applications, the trajectories of moving objects span multiple sensor clusters, i.e., moving objects are present in partial sensor clusters and absent from the others. Hence, counting the absence of both objects in a pair makes no meaningful contribution to the similarity measurement. To address this issue, we utilize the Jaccard similarity coefficient [43] to incorporate the opinions of the CHs whose opinions are known, i.e., $\pi(o_i) \neq -1$, when measuring the similarity between a pair of objects. The Jaccard similarity coefficient, which compares the similarity between two binary vectors, is widely used in information retrieval applications where the importance of positive and negative opinions is asymmetric [11]. The metric is more objective and reasonable for our applications than other measures, such as the simple matching coefficient and the overlap coefficient [39]. Let $\pi_k(o_i)$ denote the mapping of the local group ID and o_i obtained from CH_k; and let $I_k(o_i) = 1$ indicate that $\pi_k(o_i) \neq -1$. The group relationship of o_i and o_j in G_k, denoted by $c_k(o_i, o_j)$, represents that objects o_i and o_j belong to the same group in G_k; i.e.,

$$c_k(o_i, o_j) = \begin{cases} 1, & \text{if } \pi_k(o_i) = \pi_k(o_j) \neq -1, \\ 0, & \text{else.} \end{cases}$$

For an ensemble of local grouping results C, we define the Jaccard similarity coefficient of o_i and o_j, denoted by S_{ij}, as the proportion of local grouping results that contain o_i and o_j and belong to the same group, where

$$S_{ij} = \frac{\sum_{k=0}^{\mathcal{K}-1} c_k(o_i, o_j)}{\sum_{k=0}^{\mathcal{K}-1} I_k(o_i) + \sum_{k=0}^{\mathcal{K}-1} I_k(o_j) - \sum_{k=0}^{\mathcal{K}-1} c_k(o_i, o_j)}. \quad (5)$$

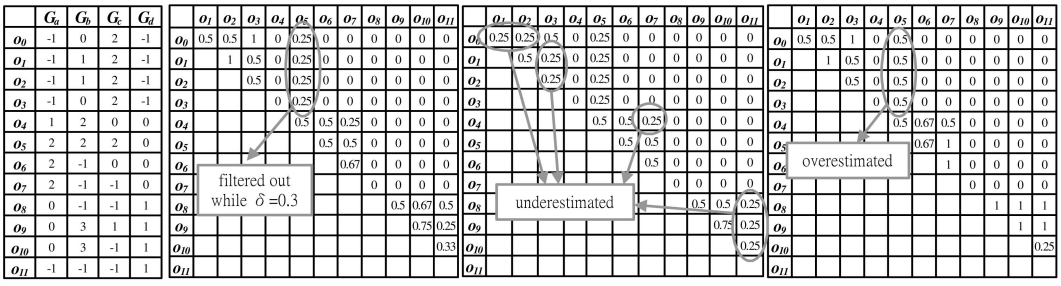
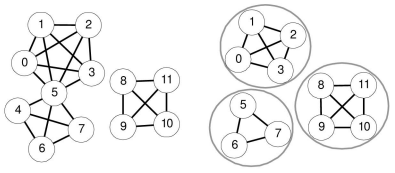


Fig. 8. (a) An example of four local grouping results and three similarity matrices for (b) Jaccard coefficient, (c) simple match coefficient, and (d) overlap coefficient (the simple math coefficient is defined by $S^s_{ij}=\frac{\sum_{k=0}^{\mathcal{K}-1} c_k(o_i,o_j)}{\mathcal{K}}$ and the overlap coefficient is defined by $S^o_{ij}=\frac{\sum_{k=0}^{\mathcal{K}-1} c_k(o_i,o_j)}{\max(\sum_{k=0}^{\mathcal{K}-1} I_k(o_i),\sum_{k=0}^{\mathcal{K}-1} I_k(o_j))}$).



δ	G_δ	$\sum$NMI(G_δ,G_i)
0.1	{{0,1,2,3},{5,6,7},{8,9,10,11}}	2.322
0.2	{{0,1,2,3},{5,6,7},{8,9,10,11}}	2.322
0.3	{{0,1,2,3},{4,5,6,7},{8,9,10,11}}	2.636
0.4	{{0,1,2,3},{4,5,6,7},{8,9,10}}	2.401
0.5	{{0,1,2,3},{4,5,6,7},{8,9,10}}	2.401

(c)

Fig. 9. An example of (a) a similarity graph ($\delta = 0.1$) and (b) its partition result: highly connected subgraphs ($\delta = 0.1$). (c) The ensembling results for a set of thresholds $D = \{0.1, \ldots, 0.5\}$.

Note that the denominator is the number of local grouping results in which o_i or o_j belongs to some group. Using (5), we compute S_{ij} for each pair of objects and construct a similarity matrix SM, where $SM[i,j] = S_{ij}$ and $S_{ij} \in [0,1]$. Next, we give an example to illustrate the advantage of our approach. Fig. 8a shows the inconsistency among the local grouping results $C = \{G_a, G_b, G_c, G_d\}$ while Fig. 8b shows the similarity matrix calculated according to (5). Obviously, the coefficients of objects in the same group are different from that in different groups. In addition, Figs. 8c and 8d show that the simple match coefficient underestimates the objects' correlations and the overlap coefficient overestimates the correlations so that they cannot correctly cluster the objects with a threshold in this example.

In the second step, based on SM, the algorithm generates a partitioning result G_δ, and derives an unweighted, undirected graph for each $\delta \in D$ as follows: For each S_{ij} in SM, if $S_{ij} > \delta$, the algorithm adds an edge between o_i and o_j to the similarity graph $G(V,E)$, as shown in Line 10. It then partitions the graph to generate a partitioning result G_δ by HCS (Line 11). In Fig. 8, we set $D = \{\frac{i}{10} | 1 \le i \le 5\}$ and compute the summation of NMIs of G_δ for every $\delta \in D$. Fig. 9a shows the similarity graph $\delta = 0.1$, and Fig. 9b shows the HCS partitioning result.

In the final step, the algorithm uses NMI to select the ensembling result $G_{\delta'}$. For a set of thresholds D, we rewrite the expression in (4) as follows:

$$G_{\delta'} = arg\max_{\delta \in D} \sum_{i=0}^{\mathcal{K}-1} NMI(G_i, G_\delta).$$

As a result, we can calculate the total information of G_δ with respect to C, i.e., $\sum_{i=0}^{\mathcal{K}-1} NMI(G_i, G_\delta)$, for every δ and choose the G_δ that contains the most information about C as $G_{\delta'}$ (Lines 13-15). For example, Fig. 9c shows the table of the partition results and the summation of their NMIs for every $\delta \in D$. The final ensembling results of the simple match coefficient and overlap coefficient are $\{\{4,5,6,7\},\{8,9,10\}\}$ and $\{\{0,1,2,3\},\{5,6,7\},\{8,9,10,11\}\}$, respectively. In contrast, our algorithm generates the best solution $G_{0.3} = \{\{0,1,2,3\},\{4,5,6,7\},\{8,9,10,11\}\}$, which is identical to the input workload. This demonstrates the effectiveness of using the Jaccard coefficient in our approach.

The sink node uses the CE algorithm to combine the local grouping results. It then assigns a global group ID to each group and sends the group information to the CHs for subsequent collection of the location data.

4 Energy Efficient Object Tracking Sensor Network (OTSN)

WSNs designed for tracking the locations of moving objects are called OTSNs. Conserving energy in OTSNs is more

```
Algorithm: Cluster Ensembling
Input: O = {o0,o1,...oN}, C = {Gi |0 ≤ i<k }, D = {δi |0 ≤ i<d }
Output : Gδ'
0.   init sum[]
1.   init SM[][]
2.   idx = 0
3.   max = 0
4.   /*building similarity matrix by Jaccard*/
5.   for 0 ≤ i<N-1
6.       for i+1 ≤ j<N
7.           SM[i,j] = getSij(C)
8.   /*select the partition with max ∑NMI(Gδ, Gi)*/
9.   for 0 ≤ i<d
10.      Graph(V, E) = Convert2Graph(SM,δi)
11.      Gδi = HCS(Graph(V, E))
12.      sum[i] = ∑_{0≤j<k} NMI(Gj, Gδi)
13.      if sum[i] >max then
14.          max = sum[i]
15.          idx = i
16.  Gδ' = Gδidx
17.  return Gδ'
```

Fig. 10. The Cluster Ensembling Algorithm.

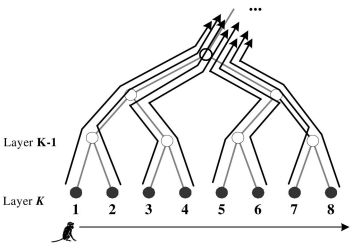


Fig. 11. An example of a conventional update-based OTSN with K layers.

difficult than in WSNs that monitor immobile phenomena, such as humidity, vibrations, or sound, because the target objects are moving. Hence, OTSNs require special consideration and designs to track moving objects efficiently. For example, in [42], [49], [51], [55], the speed and direction of a moving object are estimated and use to reduce the number of transmitted packets or to minimize the number of "awake" sensors in wake-up scheduling. DCTC [57] dynamically configures a convoy tree based on certain trajectory prediction schemes to track a mobile target with high tree coverage and low energy consumption. In STUN [26], an efficient tree is built for tracking objects based on the frequency of objects' movements over a region. For in-network object tracking, Tseng and coworkers [31] further develop several tree structures that consider the physical topology of the sensor network. All the above works exploit the movement characteristics of a single moving object.

To design an energy-efficient OTSN, we leverage the group information and the object movement patterns derived in Section 3. In a conventional update-based OTSN, sensors are assigned a tracking task, i.e., to update the sink with the location data of moving objects at every tracking interval. When a sensor detects an object of interest, it sends an update packet upward to the sink. Since the sensing area of sensors may overlap, a data aggregation step is required to avoid making unnecessary location updates. For example, at best, eight update packets passing through K layers to the sink are required for an object that moves from sensor s_1 to s_8, as shown in Fig. 11. Existing approaches employ techniques like data aggregation or location prediction to reduce the number or sizes of transmitted update packets [31], [42], [49]. In contrast, our tracking algorithm uses the group information discovered by the proposed distributed mining algorithm to further aggregate the location data of a group of moving objects. Figs. 12a and 12b show an aggregation

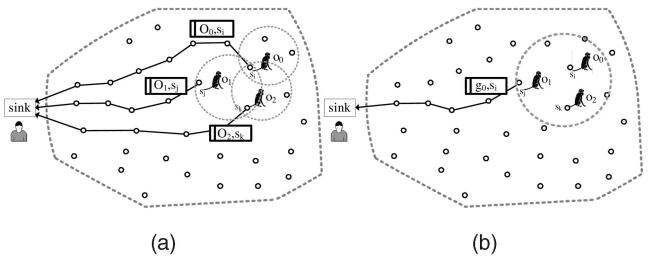


Fig. 12. An example of tracking a group of moving objects. (a) Monitoring multiple objects, respectively. (b) Monitoring multiple objects with group data aggregation.

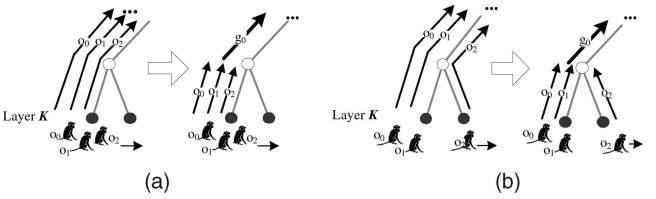


Fig. 13. Examples of group data aggregation with (a) $EB = 0$ and (b) $EB = 1$.

scenario where a group ID, i.e., g_0, replaces three object IDs in the location update, and thus reduces the number of packets as well as the packet sizes. To the best of our knowledge, very few researchers have considered group relationships in location data aggregation. In addition, our OTSN forms Sensor Groups (SGs) distributedly to reduce the load on the CHs and the in-network traffic. Adaptively adjusting the size of an SG for a group of objects allows us to control the overhead of forming the SG. Moreover, we employ location prediction and data aggregation simultaneously. Our distributed mining algorithm generates the PSTs as prediction models to suppress the location updates. In the following sections, we discuss the design considerations in detail. Because of space limitations, we only show the tracking algorithm, and demonstrate our OTSN with an example in Appendix 5, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TKDE.2009.202.

4.1 Group Data Aggregation

In a conventional update-based OTSN, multiple sensors may send readings to the CHs for the same object; thus, the CHs aggregate the update packets to reduce unnecessary long-distance transmissions. When tracking multiple moving objects with the group characteristics, we can query multiple objects and update their location data simultaneously. In our design, the CHs perform group data aggregation as follows: If multiple objects belong to the same group, the CHs combine the location data of all the objects in a single packet and substitute a group ID for the individual object IDs. Specifically, we combine the location data of a group of objects by using a triple (group ID, sensor ID, time stamp). Fig. 13a shows an example where the number and size of packets are reduced by applying group data aggregation.

We define the cost of transmitting a packet from a source sensor to a target sensor by the distance between them, where the distance is the hop count between the two sensors. The transmission cost of an OTSN, denoted by TC, is the average cost of sending the location data of an object from a sensor to the sink, i.e., the hop count per update packet. Let TC_{cu} denote the transmission cost of a conventional update-based OTSN, and let p_g denote the probability that a group of objects will be observed simultaneously. The transmission cost of applying group data aggregation, denoted by TC_g, is expressed by

$$TC_g = \frac{TC_{cu} \times p_g \times m + TC_{cu} \times (1 - p_g) \times N}{N} = TC_{cu} \times \left(1 - \left(1 - \frac{m}{N}\right) \times p_g\right),$$

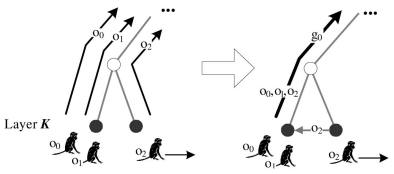


Fig. 14. An example of in-network data aggregation.

where m is the number of groups and N is the number of objects. The saving is $(TC_{cu} - TC_g)/TC_{cu} = (1 - \frac{m}{N}) \times p_g$. For example, if $p_g = 0.6$, $m = 5$, and $N = 50$, the saving will be 54 percent.

Since the accuracy of sensor networks is inherently limited, allowing approximation of sensor readings is a compromise [45]. We regulate the accuracy by an error bound, defined as the maximal hop count between the real and reported locations of an object. This improves energy savings in two ways: it reduces the number of long-distance transmissions, and it increases group data aggregation. Fig. 13b shows an example of group data aggregation with an error bound $EB = 1$. We observe that the CH successfully reduces the number and size of the packets, even when a group of objects is located by different sensors.

4.2 In-Network Data Aggregation

It has been shown that in-network data aggregation in WSNs improves the scalability and reduces long-distance communication demands, and thus saves energy. However, the CHs performing data aggregation become hot spots. Our OTSN distributedly forms a SG comprised of a set of neighboring sensors to detect the location of an object and trace a group of moving objects. The size of an SG is determined by its radius, defined as the hop count between the central sensor and the farthest sensors in the same SG. We adaptively adjust the members of an SG in a distributed manner as objects move. When a sensor detects an object, it sends an invitation message to its nearby neighbors by flooding. On receipt of the message, a neighbor joins the SG and transmits its location data. If a neighbor receives multiple invitations, it joins an SG randomly. The initial sensor then combines the location data of multiple objects in a single packet and forwards it to the CH. This is called in-network data aggregation. Therefore, SGs reduce the burden of CHs and reduce network traffic within a cluster. Fig. 14 shows an example where network traffic is reduced by applying in-network data aggregation. We observe that two packets sent from sensors to the CH are eliminated at the cost of short-distance query and response traffic.

Note that a sensor querying its neighbors to form an SG by flooding generates more network traffic. However, with group information, we can estimate the dispersion range of a group of objects and choose a suitable SG size to limit the cost of forming the SG. For an SG with radius R, the flooding cost and the cost of sending the result to the CH is $d + 4R \times (R + 1)$, where d is the average distance between a sensor and its CH. On the other hand, the cost of collecting location data without aggregation is estimated as $N_i \times d$, where N_i is the group size. If $d + 4R \times (R + 1)$ is less than $N_i \times d$, then the in-network data aggregation is beneficial. Therefore, our tracking algorithm can benefit from in-network data aggregation when both the sizes of groups and the size of sensor clusters are large.

4.3 PST Prediction

In the mining phase, our algorithm generates a PST for each object or group of objects. In the tracking phase, we use the PSTs as prediction models for location prediction. By using the same model for the sink and the CH, our approach reduces energy consumption. Specifically, it enables the CH to avoid sending the updates for predictable locations because the sink can recover the locations via the same prediction model. When the CH fails in the location prediction, an update packet is delivered to the sink to correct the prediction. Fig. 15a shows an example of PST prediction suppressing a location update, which is indicated by the dashed line.

To evaluate the efficiency of PST prediction, we analyze the transmission cost of the update-based OTSN with PST prediction.

Assume K is the number of layers in the network structure and each sensor cluster contains $n \times n$ sensors, the total number of sensors in an update-based OTSN is n^{2K}. In addition, the number of layer-i nodes (clusters or sensors) is n^{2i}. Each sensor has a coordinate (x, y) in a cluster, where $0 \leq x, y < n$. The distance between a sensor and the CH in a cluster is $(i + j) \times d$, where d is the hop count between two adjacent nodes. Note that communications between adjacent nodes of layer-i must pass through all nodes in the lowest layer between them. Thus, d of layer-i is n^{K-i}. When a CH is located in the bottom left-hand corner of a mesh network, the cost of transmitting a packet between the CH and any other sensors within the same cluster can be expressed by

$$D(i) = \frac{\sum_{x=0}^{n-1} \sum_{y=0}^{n-1} (x + y) \times n^{K-i}}{n^2} = n^{K-i} \times (n - 1).$$

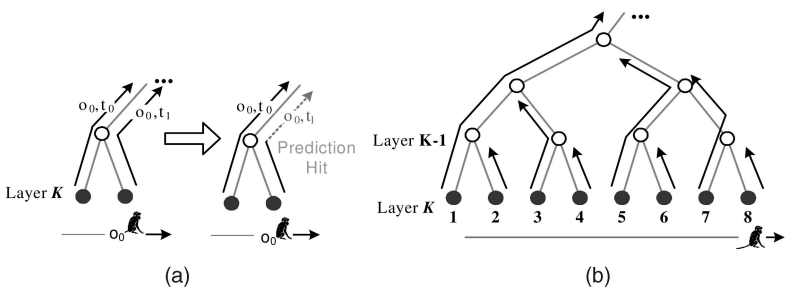


Fig. 15. (a) An example of PST prediction where an update is eliminated. (b) An example of the best case of PST prediction.

In the worst case, where all predictions fail and all update packets pass from the bottom layer to the sink, the transmission cost _(denoted by TC_{worst}) is the same as that of the conventional update-based OTSN. We formulate this scenario as $TC_{worst} = \sum_{i=1}^{K} D(i) = n^K - 1$. In the best case, where all predictions are correct, we only need to update the location of a new object, as shown by the arrows in Fig. 15b. For example, as a new object enters the range of s_1 and the CHs on the path to the sink, a packet passing through K layers is required to update the location sequence data set at those CHs. When the object moves from sensor s_1 to s_2, the latter only updates the location of the object to its CH in layer $K-1$, since the CH correctly predicts the location and the sink recovers the location via the same prediction model. In the best case, the transmission cost, denoted by TC_{best}, contains the product of the cost and the number of arrows between layer-$i-1$ and layer-i for $1 \leq i \leq K$. We formulate TC_{best} as follows:

$$\begin{aligned} TC_{best} &= \frac{\sum_{i=1}^{K} D(i) \times n^{2i}}{n^{2K}} \\ &= \frac{\sum_{i=1}^{K} n^{K-i} \times (n-1) \times n^{2i}}{n^{2K}} = n - \frac{1}{n^{K-1}}. \end{aligned} \quad (6)$$

Equation (6) shows that a network structure with more layers results in a lower TC_{best} cost. However, the best case does not always hold. Besides, if a network structure has more layers, there are probably more predictions; thus, the overall prediction hit rate will be lower. Therefore, we need to conduct more experiments to study the impact of the network structure.

5 Experiments

We implement an event driven simulator in C++ with SIM [10] to evaluate the performance of our design. Since it is difficult to obtain real location data, this study is based on synthetic data only. In the experiments, we use the Reference Point Group Mobility Model [23] to synthetically generate location data, i.e., the coordinates (x, y), for a group of objects. The location-dependent mobility model [18] is used to simulate the roaming behavior of a group leader. The other member objects are followers that are uniformly distributed within a specified group dispersion radius (GDR) of the leader, where the GDR is the maximal hop count between the followers and their leader. We utilize the GDR to control the dispersion degree of the objects. A smaller GDR implies stronger group relationships, i.e., objects are closer together. In addition, we control the movement range of a group of objects by using the movement distance (d), which is the linear distance between the starting point and the furthest point reached by the leader object. A longer distance implies that objects move across more sensor clusters. To ensure our simulation reflects the real-world scenarios, we input objects with a random-walk model. In the following sections, we first study the effectiveness of our distributed mining algorithm, and then demonstrate the efficiency of our OTSN.

5.1 Effectiveness of the Distributed Mining Algorithm

To study the effectiveness of our distributed mining algorithm, we conduct six experiments as follows: First, we inspect the impacts of the group number (m), the SG radius (R), and the movement distance (d) on the grouping quality.[5] Then, we study the influence of the location-dependent GDR on the grouping quality and conduct a case study to show that comparing the similarity on the entire trajectories by using average euclidean distance may not reveal the local group relationships. Finally, we investigate the discrimination ability of the new proposed similarity measure sim_p.

In the following experiments, we assume a 16×16 mesh network of a two-layer structure, i.e., it is composed of 16 clusters, each of which has 16 sensors. There are m groups of objects and N_r random-walk objects walking in the mesh network, where N_r is equal to the total number of objects of the m groups. The size of each group is uniformly distributed between 5 and 15. The speed of the objects is 1 sensor per time unit, and the tracking interval is 0.5 time units. The default values of EB, L_{max}, d, and R are 0, 6, 12, and 0, respectively. We set the PST parameters $(P_{min}, \alpha, \gamma_{min}, r) = (0.01, 0, 0.0001, 1.2)$ empirically. To evaluate the grouping quality, we take the NMI of the ensembling result $G_{\delta'}$ and the input workload G_{input} as the external evaluation metric. Note that a higher NMI value indicates better grouping quality.[6]

In the first experiment, we compare the grouping quality and the transmission costs of our distributed approach (denoted by DGMP) with a vector-based K-means approach (denoted by VK)[7] [19]. The GDR varies from 0.1 to 1.0 while m is uniformly selected between a and b, where (a, b) are set to $(10, 20), \ldots,$ and $(40, 50)$. The parameter k representing the specified cluster number of the K-means algorithm is set to $\frac{a+b}{2}$. Fig. 16a shows a decreasing trend in the NMI values of DGMP and VK as the GDR increases. When the GDR is less than 0.4, our approach clusters the patterned objects exactly and filters out most random-walk objects. The grouping quality decreases to 0.8 as the GDR approaches 1. In contrast, the NMI of VK is less than 0.7 even when the GDR is small. This is because the clustering result of the K-means algorithm is sensitive to the seed selection and the parameter k. Note that the number of unfiltered random-walk objects increases with N_r. These objects often do not belong to any group, or they belong to small groups with two or three members that can be removed by postprocessing. In real-world applications, there may be some unwanted objects that are viewed as outliers, e.g., objects with random-walk movements. To remove a type of outliers, we first train a PST of a sample object of this type and then remove the objects that are similar to it. Through this approach, we prune the similarity graph and thereby reduce the clustering cost.

When the grouping results of DGMP and VK are utilized in our group data aggregation technique to reduce the

5. Due to the page limit, we discuss the impact of L_{max} in Appendix 6, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TKDE.2009.202.

6. For example, given $G_{input} = \{\{0, 1, \ldots, 4\}, \{5, 6, \ldots, 9\}, \{10, 11, \ldots, 14\}, \{15, 16, \ldots, 19\}, \{20, 21, \ldots, 24\}\}$, the NMI of the result {{1, 2, 3}, {6, 8, 9}, {21, 22, 23, 24}}, G_{input} is 0.56; and the NMI of the result {{0, 1, 2, 3, 4}, {5, 7, 9}, {10, 11, 12, 13, 14}, {15, 17, 18, 19}, {20, 23, 24}} and G_{input} is 0.8.

7. Instead of using sequential patterns, we generate a vector for each object by projecting its sequence into the feature domain of the significant movement patterns of all objects.

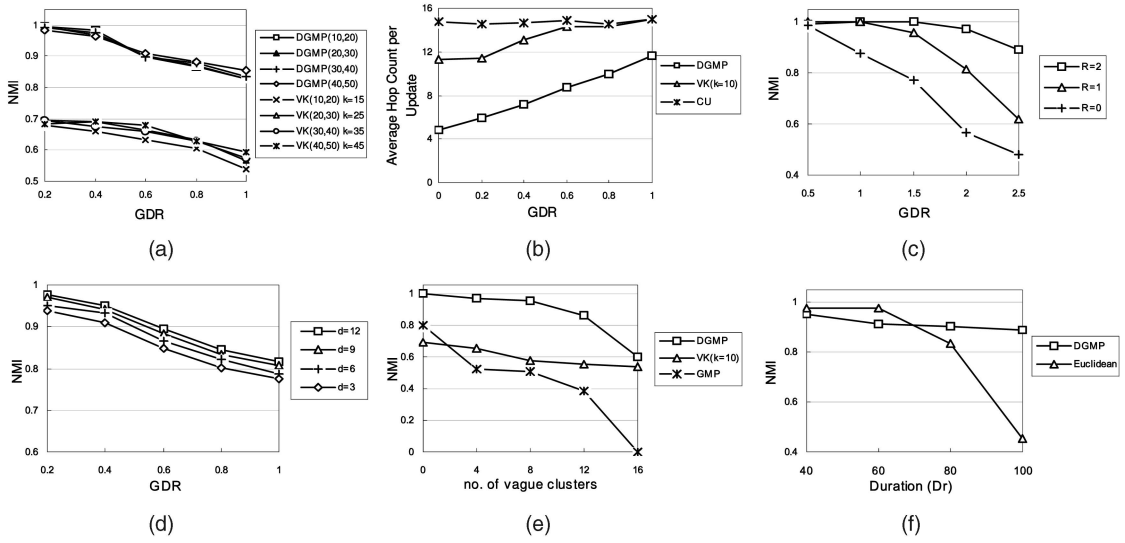


Fig. 16. The simulation results of the distributed mining algorithm. (a) Impact of the number of groups. (b) Comparison of DGMP and VK on TC. (c) Impact of the SG radius ($m = 10$). (d) Impact of the moving distance ($m = 10$). (e) Impact of the location-dependent GDR ($m = 10$). (f) Comparison of DGMP and euclidean+HCS ($m = 10$, $GDR = 0.5$).

transmission cost, Fig. 16b shows that the average hop count per update of DGMP is lower than VK. The is because DGMP provides accurate group information so that more update packets are aggregated. Compared with the case with no group data aggregation (denoted by CU), about 50 percent of the transmission cost of DGMP is reduced when the GDR is below 0.5.

In the second experiment, since objects may be more widely distributed, we investigate the influence of the size of SGs (R) on the grouping quality. For a specified SG radius, we report the ID of the central sensor of the SG as the location of each captured object. Fig. 16c shows that the NMI increases with R, especially when the GDR is large. This experiment shows that if objects are more widely distributed, we can adaptively specify a larger size of SGs to improve the clustering quality.

In the third experiment, we study the impact of moving distance (d). When objects travel over a larger range, they are likely to move across more clusters; hence, more CHs report grouping results. In other words, more CHs participate in deciding the group relationships. Fig. 16d shows that the NMI increases when the distance is larger. The result shows that the CE algorithm effectively improves the grouping quality.

In the fourth experiment, we investigate the influence of the location-dependent GDR, i.e., the trajectories with varying GDR in regions, and compare with VK and an approach with only the GMPMine algorithm (denoted by GMP). In this experiment, clusters are divided into two types: a vague cluster with $GDR = 2.0$ represents a region where a group of objects locate widely while a distinct cluster with $GDR = 0.5$ represents a region where objects stay closely. We select the vague clusters randomly and vary the number of vague clusters from 0 to 16. Fig. 16e shows that when the number of vague clusters increases, the curve of DGMP degrades smoothly while that of GMP degrades significantly. Since DGMP uses the local grouping results from distinct clusters to discover the group relationships, it provides better stability than VK and GMP. On the other hand, VK is not sensitive to the number of vague clusters. This is because VK compels to cluster each object while DGMP and GMP leave alone those objects without distinct group relationships.

In the fifth experiment, we conduct a case study to compare our approach with a euclidean approach that takes the average euclidean distance of entire trajectories as the similarity measure and applies HCS [21] to cluster objects. Assume that moving objects migrate through a passage with coordinates $(4, 4, 12, 8)$ between two grasslands with coordinates $(0, 0, 4, 12)$ and $(12, 4, 16, 16)$, respectively. The movements of objects in the grasslands are modeled by random walk while the trajectories in the migration passage are modeled by the location-dependent mobility model. We vary the duration (Dr) that objects stay in grasslands. Fig. 16f shows that the NMI of euclidean degrades as Dr increases. This is because groups of objects stay closely in the passage and a group of objects spread widely at grassland so that the average euclidean distance of trajectories of two objects in the same group approaches that of two objects in different groups as Dr increases. In contrast, although the group relationships in the grasslands are vague, our approach uses on the local grouping results in the migration passage to discover the group relationships. Therefore, our approach is able to achieve better grouping quality in this case.

Last, we study the discrimination ability of sim_p and compare with sim_s [53]. sim_s is a similarity measure between a sequence s and a PST T, defined as

$$sim_s(s, T) = \prod_{i=0}^{l-1} \frac{P^T(\sigma_i|\sigma_0 \ldots \sigma_{i-1})}{P^T(\sigma_i)},$$

where l is length of s. To compare the similarity of two objects by using sim_s, we take the sequence of one object and the PST of the other as the input. Assume there are 10 groups of objects walking in a one-layer 16×16 mesh network. We compute the sim_p and sim_s of the objects in the same group as well as that of the objects in the different groups. Since

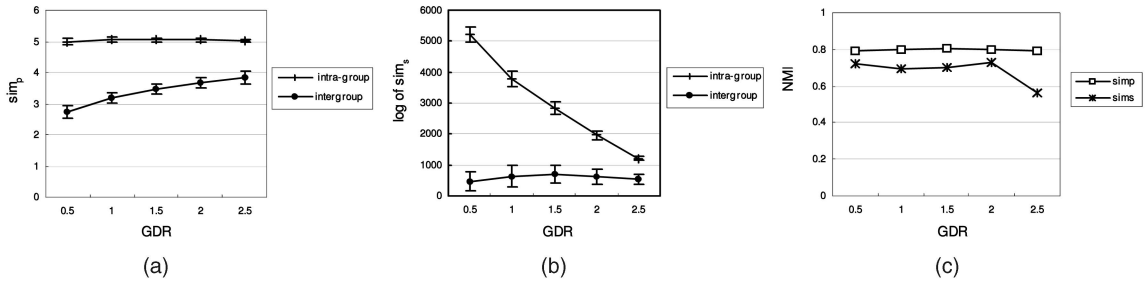


Fig. 17. Comparison of similarity measures sim_p and sim_s. (a) Discrimination ability of sim_p. (b) Discrimination ability of sim_s. (c) Comparison of GMP with sim_p and with sim_s.

sim_s overflows quickly, we use *log* of sim_s instead. Figs. 17a and 17b show that sim_p and sim_s effectively differentiate objects in the same group (intragroup) and objects in different group (intergroup), where the y-error bars represent the standard deviation. As the GDR increases, the curve of sim_s of intragroup approaches that intergroup quickly while the curve of sim_p of intergroup approaches the curve of intragroup smoothly. We use sim_p and sim_s with minimal threshold 4.5 and 1,500, respectively, in our GMPMine algorithm to cluster objects. Fig. 17c shows that sim_p achieves higher NMI than sim_s.

5.2 Efficiency of the Proposed OTSN

To evaluate the efficiency of the proposed OTSN, we investigate the impacts of the network structure, accuracy error bound (EB), and SG radius (R) as GDR varies. To the best of our knowledge, few researchers have considered group relationships in location data aggregation. Therefore, we only compare our design with a conventional update-based OTSN with a naive data aggregation technique (denoted by CU). In the following experiments, we assume that there are five groups of objects, each of which contains five objects, walking in a mesh network composed of 65,536 sensors.

First, we study impact of the network structure on the transmission cost. The sensors are configured in three network structures, i.e., $K = 2$, 4, and 8. Fig. 18a shows that when the group relationships of moving objects exist, the proposed OTSN outperforms CU. Compared with CU, at least 42 percent of the TC is reserved when $K = 2$ and the GDR is less than 1. In addition, the curve of $K = 2$ is close to that of $K = 4$ and both $K = 2$ and $K = 4$ outperform $K = 8$. Fig. 18b shows the prediction hit rates of the three network structures degrade as GDR increases. Since the leader object's movement path together with the GDR sets up a spacious area where the member objects are randomly distributed, a larger GDR implies that the location sequences have higher entropy, which degrades both the prediction hit rate and the transmission cost. We find the prediction hit rate of $K = 2$ is higher than that of $K = 4$; however, the TCs of $K = 2$ and $K = 4$ are similar. This is because that the larger cluster size of $K = 2$ generates more in-network traffic and thus counteracts the advantage of its higher prediction rate.

As mentioned earlier, approximating the readings of sensors is a compromise to reduce the number of long-distance transmissions and increase group data aggregation. In the second experiment, we study the impact of the SG radius on the transmission cost. Fig. 18c shows that by tolerating a lower accuracy with $EB = 1$, 44 percent of the TC is reduced as $GDR = 2.5$. (Note that in this experiment, we set the SG radius accordingly, i.e., $R = EB$.)

Although in-network data aggregation makes a smaller contribution than PST prediction and group data aggregation, it is worth investigating because, in some applications, in-network traffic dominates the total network traffic. For example, in a warning system where a warning message is only necessary while an object is outside a specified area, the in-network traffic generated to report the location of the object is greater than the internetwork traffic. Hence, we study the effectiveness of in-network data aggregation in the last experiment. Fig. 18d shows that in-network data aggregation helps reduce the in-network traffic by about 20 percent when $GDR = 2.5$. Moreover, when the GDR is large, adaptively choosing a larger radius to form a larger SG can reduce the in-network traffic by 44 percent.

6 Conclusions

In this work, we exploit the characteristics of group movements to discover the information about groups of moving objects in an OTSN. In contrast to the centralized

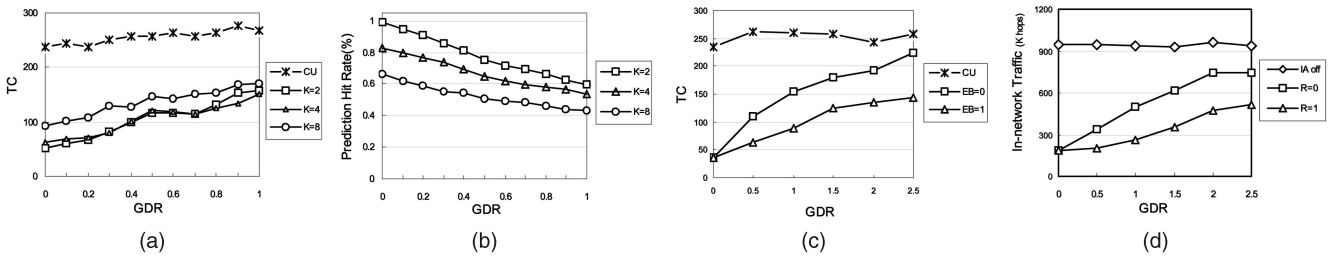


Fig. 18. (a) Transmission cost ($\mathrm{EB} = 0$) and (b) prediction hit rate ($\mathrm{EB} = 0$) of our OTSN for the three structures ($m = 5$, $n_r = 0$). (c) The impact of accuracy ($\mathrm{K} = 2$) and (d) the transmission cost of in-network data aggregation ($\mathrm{EB} = 0$, $\mathrm{K} = 2$).

mining technique, we mine the group information in a distributed manner. We propose a novel mining algorithm, which consists of a local GMPMine algorithm and a CE algorithm, to discover group information. Our algorithm mines object movement patterns as well as group information and the estimated group dispersion radius. Other than clustering trajectories, we can apply the distributed clustering approach to heterogeneous and distributed sequential data sets, such as web logs or gene sequence. Using the mined object movement patterns and the group information, we design an energy-efficient OTSN. The contribution of our approach is threefold: 1) it reduces energy consumption by allowing CHs to avoid sending the prediction-hit locations, because the locations can be recovered by the sink via the same prediction model; 2) it leverages group information in data aggregation to eliminate redundant update traffic; and 3) it sets the size of an SG adaptively to limit the amount of flooding traffic. Our experimental results show that the proposed mining technique achieves good grouping quality. Furthermore, the proposed OTSN with PST prediction, group data aggregation, and in-network data aggregation significantly reduces energy consumption in terms of the transmission cost, especially in the case where moving objects have distinct group relationships.

ACKNOWLEDGMENTS

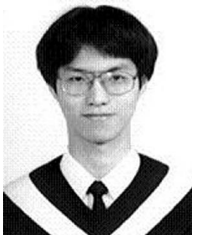
The work was supported in part by the National Science Council of Taiwan, R.O.C., under Contracts NSC99-2218-E-005-002 and NSC99-2219-E-001-001.

REFERENCES

[1] African Elephant, http://www.defenders.org/wildlife_and_habitat/wildlife/elephant.php, 2010.
[2] Mica2 Sensor Board, http://www.xbow.com, 2010.
[3] Stargate: A Platform X Project, http://platformx.sourceforge.net, 2010.
[4] R. Agrawal and R. Srikant, "Mining Sequential Patterns," *Proc. 11th Int'l Conf. Data Eng.*, pp. 3-14, 1995.
[5] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless Sensor Networks: A Survey," *Computer Networks*, vol. 38, no. 4, pp. 393-422, 2002.
[6] J.N. Al Karaki and A.E. Kamal, "Routing Techniques in Wireless Sensor Networks: A Survey," *IEEE Wireless Comm.*, vol. 11, no. 6, pp. 6-28, Dec. 2004.
[7] A. Apostolico and G. Bejerano, "Optimal Amnesic Probabilistic Automata or How to Learn and Classify Proteins in Linear Time and Space," *Proc. Fourth Ann. Int'l Conf. Computational Molecular Biology*, pp. 25-32, 2000.
[8] H. Ayad, O.A. Basir, and M. Kamel, "A Probabilistic Model Using Information Theoretic Measures for Cluster Ensembles," *Proc. Fifth Int'l Workshop Multiple Classifier Systems*, pp. 144-153, June 2004.
[9] G. Bejerano and G. Yona, "Variations on Probabilistic Suffix Trees: Statistical Modeling and the Prediction of Protein Families," *Bioinformatics*, vol. 17, no. 1, pp. 23-43, 2001.
[10] D. Bolier, "SIM: A C++ Library for Discrete Event Simulation," http://www.cs.vu.nl/eliens/sim, Oct. 1995.
[11] D. Bollegala, Y. Matsuo, and M. Ishizuka, "Measuring Semantic Similarity between Words Using Web Search Engines," *Proc. 16th Int'l World Wide Web Conf.*, pp. 757-766, 2007.
[12] L. Chen, M. Tamer Özsu, and V. Oria, "Robust and Fast Similarity Search for Moving Object Trajectories," *Proc. ACM SIGMOD*, pp. 491-502, 2005.
[13] M.-S. Chen, J.S. Park, and P.S. Yu, "Efficient Data Mining for Path Traversal Patterns," *Knowledge and Data Eng.*, vol. 10, no. 2, pp. 209-221, 1998.
[14] T.M. Cover and J.A. Thomas, *Elements of Information Theory*. Wiley Interscience, Aug. 1991.
[15] X.Z. Fern and C.E. Brodley, "Random Projection for High Dimensional Data Clustering: A Cluster Ensemble Approach," *Proc. 20th Int'l Conf. Machine Learning*, pp. 1186-1193, June 2003.
[16] A.L.N. Fred and A.K. Jain, "Combining Multiple Clusterings Using Evidence Accumulation," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 27, no. 6, pp. 835-850, June 2005.
[17] F. Giannotti, M. Nanni, F. Pinelli, and D. Pedreschi, "Trajectory Pattern Mining," *Proc. 13th ACM SIGKDD*, pp. 330-339, 2007.
[18] B. Gloss, M. Scharf, and D. Neubauer, "Location-Dependent Parameterization of a Random Direction Mobility Model," *Proc. IEEE 63rd Conf. Vehicular Technology*, vol. 3, pp. 1068-1072, 2006.
[19] V. Guralnik and G. Karypis, "A Scalable Algorithm for Clustering Sequential Data," *Proc. First IEEE Int'l Conf. Data Mining*, pp. 179-186, 2001.
[20] J. Han, J. Pei, B. Mortazavi-Asl, Q. Chen, U. Dayal, and M. Hsu, "Freespan: Frequent Pattern-Projected Sequential Pattern Mining," *Proc. Sixth ACM SIGKDD*, pp. 355-359, 2000.
[21] E. Hartuv and R. Shamir, "A Clustering Algorithm Based on Graph Connectivity," *Information Processing Letters*, vol. 76, nos. 4-6, pp. 175-181, 2000.
[22] J. Hightower and G. Borriello, "Location Systems for Ubiquitous Computing," *Computer*, vol. 34, no. 8, pp. 57-66, Aug. 2001.
[23] X. Hong, M. Gerla, G. Pei, and C. Chiang, "A Group Mobility Model for Ad Hoc Wireless Networks," *Proc. Ninth ACM/IEEE Int'l Symp. Modeling, Analysis and Simulation of Wireless and Mobile Systems*, pp. 53-60, Aug. 1999.
[24] H. Kargupta, W. Huang, K. Sivakumar, and E.L. Johnson, "Distributed Clustering Using Collective Principal Component Analysis," *Knowledge and Information System*, vol. 3, pp. 422-448, 2001.
[25] D. Katsaros and Y. Manolopoulos, "A Suffix Tree Based Prediction Scheme for Pervasive Computing Environments," *Proc. Panhellenic Conf. Informatics*, pp. 267-277, Nov. 2005.
[26] H.T. Kung and D. Vlah, "Efficient Location Tracking Using Sensor Networks," *Proc. Conf. IEEE Wireless Comm. and Networking*, vol. 3, pp. 1954-1961, Mar. 2003.
[27] C. Largeron-Leténo, "Prediction Suffix Trees for Supervised Classification of Sequences," *Pattern Recognition Letters*, vol. 24, no. 16, pp. 3153-3164, 2003.
[28] J.-G. Lee, J. Han, and K.-Y. Whang, "Trajectory Clustering: A Partition-and-Group Framework," *Proc. ACM SIGMOD*, pp. 593-604, 2007.
[29] D. Li, K.D. Wong, Y.H. Hu, and A.M. Sayeed, "Detection, Classification, and Tracking of Targets," *IEEE Signal Processing Magazine*, vol. 19, no. 2, pp. 17-30, Mar. 2002.
[30] Y. Li, J. Han, and J. Yang, "Clustering Moving Objects," *Proc. 10th ACM SIGKDD*, pp. 617-622, 2004.
[31] C.-Y. Lin, W.-C. Peng, and Y.-C. Tseng, "Efficient In-Network Moving Object Tracking in Wireless Sensor Networks," *IEEE Trans. Mobile Computing*, vol. 5, no. 8, pp. 1044-1056, Aug. 2006.
[32] G. Mazeroff, J. Gregor, M. Thomason, and R. Ford, "Probabilistic Suffix Models for API Sequence Analysis of Windows XP Applications," *Pattern Recognition*, vol. 41, no. 1, pp. 90-101, 2008.
[33] M. Morzy, "Prediction of Moving Object Location Based on Frequent Trajectories," *Proc. 21st Int'l Symp. Computer and Information Sciences*, pp. 583-592, Nov. 2006.
[34] M. Morzy, "Mining Frequent Trajectories of Moving Objects for Location Prediction," *Proc. Fifth Int'l Conf. Machine Learning and Data Mining in Pattern Recognition*, pp. 667-680, July 2007.
[35] M. Nanni and D. Pedreschi, "Time-Focused Clustering of Trajectories of Moving Objects," *J. Intelligent Information Systems*, vol. 27, no. 3, pp. 267-289, 2006.
[36] S. Pandey, S. Dong, P. Agrawal, and K. Sivalingam, "A Hybrid Approach to Optimize Node Placements in Hierarchical Heterogeneous Networks," *Proc. IEEE Conf. Wireless Comm. and Networking Conf.*, pp. 3918-3923, Mar. 2007.
[37] W.-C. Peng and M.-S. Chen, "Developing Data Allocation Schemes by Incremental Mining of User Moving Patterns in a Mobile Computing System," *IEEE Trans. Knowledge and Data Eng.*, vol. 15, no. 1, pp. 70-85, Jan./Feb. 2003.
[38] W.-C. Peng, Y.-Z. Ko, and W.-C. Lee, "On Mining Moving Patterns for Object Tracking Sensor Networks," *Proc. Seventh Int'l Conf. Mobile Data Management*, p. 41, 2006.
[39] C.J. Van Rijsbergen, *Information Retrieval*. Springer, 1979.

[40] D. Ron, Y. Singer, and N. Tishby, "Learning Probabilistic Automata with Variable Memory Length," *Proc. Seventh Ann. Conf. Computational Learning Theory*, July 1994.
[41] C. Roux and R.T.F. Bernard, "Home Range Size, Spatial Distribution and Habitat Use of Elephants in Two Enclosed Game Reserves in the Eastern Cape Province, South Africa," *African J. Ecology*, Oct. 2007.
[42] S. Santini and K. Romer, "An Adaptive Strategy for Quality-Based Data Reduction in Wireless Sensor Networks," *Proc. Third Int'l Conf. Networked Sensing Systems*, pp. 29-36, June 2006.
[43] G. Saporta and G. Youness, "Comparing Two Partitions: Some Proposals and Experiments," *Proc. Computational Statistics*, Aug. 2002.
[44] G. Shannon, B. Page, K. Duffy, and R. Slotow, "African Elephant Home Range and Habitat Selection in Pongola Game Reserve, South Africa," *African Zoology*, vol. 41, no. 1, pp. 37-44, Apr. 2006.
[45] A. Silberstein, "Push and Pull in Sensor Network Query Processing," *Proc. Southeast Workshop Data and Information Management*, Mar. 2006.
[46] A. Strehl and J. Ghosh, "Cluster Ensembles—A Knowledge Reuse Framework for Combining Partitionings," *Proc. Conf. Artificial Intelligence*, pp. 93-98, July 2002.
[47] J. Tang, B. Hao, and A. Sen, "Relay Node Placement in Large Scale Wireless Sensor Networks," *J. Computer Comm.*, special issue on sensor networks, vol. 29, no. 4, pp. 490-501, 2006.
[48] V.S. Tseng and K.W. Lin, "Energy Efficient Strategies for Object Tracking in Sensor Networks: A Data Mining Approach," *J. Systems and Software*, vol. 80, no. 10, pp. 1678-1698, 2007.
[49] G. Wang, H. Wang, J. Cao, and M. Guo, "Energy-Efficient Dual Prediction-Based Data Gathering for Environmental Monitoring Applications," *Proc. IEEE Wireless Comm. and Networking Conf.*, Mar. 2007.
[50] Y. Wang, E.-P. Lim, and S.-Y. Hwang, "Efficient Mining of Group Patterns from User Movement Data," *Data Knowledge Eng.*, vol. 57, no. 3, pp. 240-282, 2006.
[51] Y. Xu, J. Winter, and W.-C. Lee, "Dual Prediction-Based Reporting for Object Tracking Sensor Networks," *Proc. First Ann. Int'l Conf. Mobile and Ubiquitous Systems: Networking and Services*, pp. 154-163, Aug. 2004.
[52] J. Yang and M. Hu, "Trajpattern: Mining Sequential Patterns from Imprecise Trajectories of Mobile Objects," *Proc. 10th Int'l Conf. Extending Database Technology*, pp. 664-681, Mar. 2006.
[53] J. Yang and W. Wang, "CLUSEQ: Efficient and Effective Sequence Clustering," *Proc. 19th Int'l Conf. Data Eng.*, pp. 101-112, Mar. 2003.
[54] J. Yang and W. Wang, "Agile: A General Approach to Detect Transitions in Evolving Data Streams," *Proc. Fourth IEEE Int'l Conf. Data Mining*, pp. 559-562, 2004.
[55] J. Yick, B. Mukherjee, and D. Ghosal, "Analysis of a Prediction-Based Mobility Adaptive Tracking Algorithm," *Proc. Second Int'l Conf. Broadband Networks*, pp. 753-760, Oct. 2005.
[56] M. Younis and K. Akkaya, "Strategies and Techniques for Node Placement in Wireless Sensor Networks: A Survey," *Ad Hoc Networks*, vol. 6, no. 4, pp. 621-655, 2008.
[57] W. Zhang and G. Cao, "DCTC: Dynamic Convoy Tree-Based Collaboration for Target Tracking in Sensor Networks," *IEEE Trans. Wireless Comm.*, vol. 3, no. 5, pp. 1689-1701, Sept. 2004.

Hsiao-Ping Tsai received the BS and MS degrees in computer science and information engineering from National Chiao Tung University, Hsinchu, Taiwan, in 1996 and 1998, respectively, and the PhD degree in electrical engineering from National Taiwan University, Taipei, Taiwan, in January 2009. She is now an assistant professor in the Department of Electrical Engineering, National Chung Hsing University. Her research interests include data mining, sensor data management, object tracking, mobile computing, and wireless data broadcasting. She is a member of the IEEE.



De-Nian Yang received the BS and PhD degrees from the Department of Electrical Engineering, National Taiwan University, Taipei, Taiwan, in 1999 and 2004, respectively. From 1999 to 2004, he joined the Internet Research Lab with advisor Prof. Wanjiun Liao. Afterward, he joined the Network Database Lab with advisor Prof. Ming-Syan Chen as a postdoctoral fellow for the military services. He is now an assistant research fellow in the Institute of Information Science (IIS) and the Research Center of Information Technology Innovation (CITI), Academia Sinica.

Ming-Syan Chen received the BS degree in electrical engineering from the National Taiwan University, Taipei, Taiwan, and the MS and PhD degrees in computer, information and control engineering from the University of Michigan, Ann Arbor, in 1985 and 1988, respectively. He is now a distinguished research fellow and the director of the Research Center of Information Technology Innovation (CITI) in the Academia Sinica, Taiwan, and is also a distinguished professor jointly appointed by the EE Department, the CSIE Department, and the Graduate Institute of Communication Engineering (GICE) at the National Taiwan University. He was a research staff member at IBM Thomas J. Watson Research Center, Yorktown Heights, New York, from 1988 to 1996, the director of GICE from 2003 to 2006, and also the president/CEO of the Institute for Information Industry (III), which is one of the largest organizations for information technology in Taiwan, from 2007 to 2008. His research interests include databases, data mining, mobile computing systems, and multimedia networking, and he has published more than 290 papers in his research areas. In addition to serving as program chairs/vice-chairs and keynote/tutorial speakers in many international conferences, he was an associate editor of the *IEEE Transactions on Knowledge and Data Engineering*, the *VLDB Journal*, *Knowledge and Information Systems (KAIS)*, and also *The Journal of Information Systems Education (JISE)*, and is currently the editor-in-chief of the *International Journal of Electrical Engineering (IJEE)*. He holds, or has applied for, 18 US patents and 7 ROC patents in his research areas. He is a recipient of the National Science Council (NSC) Distinguished Research Award, the Pan Wen Yuan Distinguished Research Award, the Teco Award, the Honorary Medal of Information, and the K.-T. Li Research Breakthrough Award for his research work, and also the Outstanding Innovation Award from IBM Corporate for his contribution to a major database product. He also received numerous awards for his research, teaching, inventions, and patent applications. He is a fellow of the ACM and the IEEE.